MASTER OF COMPUTER APPLICATION

MCA-42

MOBILE APPLICATION DEVELOPMENT

Directorate of Distance Education
Guru Jambheshwar University of Science & Technology, Hisar – 125001

CONTENTS

SUBJECT: MOBILE APPLICATION DEVELOPEMENT	
COURSE CODE: MCA-42	AUTHOR: Dr. Sunil Kumar Verma
LESSON NO. 1	

Android Operating Systems

STRUCTURE

LEARNING OBJECTIVE

The student will get knowledge about the small size operating system used for android devices. This chapter will give introduction and architectural details of each operating used for mobile devices. You will also learn about the process used inside android devices. In last section you will get update challenges of the android platform.

Chapter 1. Android Framework

1.1. Introduction

In real life mobile phones are most useful and common devices for communication and to fulfil daily need among the population. The mobile device needs some operating system to interact with hardware and run some is application on this platform. The lots of services are running by the mobile devices as voice calling, message service, camera features, internet facilities etc.

During starting era of the smart mobile devices, operating systems were very simple to communicate and having some basic and limited set of features. However, the latest mobile devices are laced with many advance and smart features which almost look like a computer. These features are like high resolution screen, graphics processing unit, large memory space, multitasking, high quality camera, and compatibility with different types of external devices.

1.2. Architectures of Mobile OS

In the mobile technology's world, lots of small operating system invented one after other to equip the current requirement of users. A mobile OS is an operating software used for smartphone, PDA, tablets and other mobiles.

The OS used for mobile is a combination of features of personal computer and handheld devices. Modern mobile devices include a set of features: Touchscreen, Bluetooth, Wi-Fi, GPS for navigation, Camera (for video & photo), Voice assistant, Recorder, Music player, Screen cast, Hotspot and NFC. Based on these features the designing of operating is dependent on different types of architecture. The architectures are explained one by one in the following paragraphs.

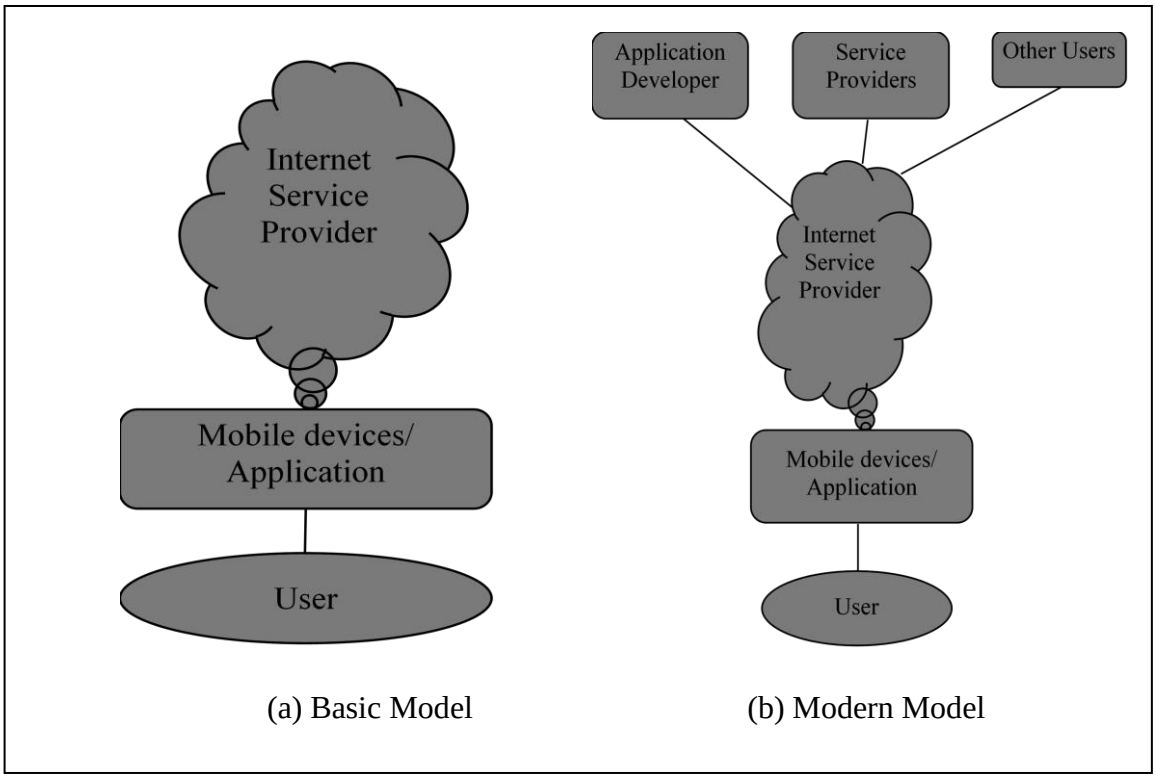


Figure 1.1 A communication model of Android devices.

1.3. Palm Operating System

This operating system was initially developed under the direction of Jeff Hawkins at Palm Computing for touch screen and PDA devices. It was planned with minimum set of features to manage personal information and later it comprises with the intention of maximum usage of process with maximum battery life to use as a smartphone. It uses a GUI based interface and memory management units is used for database, public variable and directories. Some utility features like calendar, event and time are available under system management. Licensed version of Palm OS is named as Garnet OS.

This OS has been implemented on a wide range of mobile devices including handheld gaming consoles, smartphones, wrist watches, barcode readers and GPS devices. Palm OS firstly run on Motorola and Dragon Ball processors later it was compatible with ARM processors.

Latest Palm OS came with single tasking environment, monochrome or color screen size with 480x320 pixels, Handwriting recognition kit is Graffiti2, HotSync technology for data synchronization with PC, Sound playback and record capabilities, screen locking, TCP/IP network access Serial port, infrared, Bluetooth and Wi-Fi connections, Expansion memory card support.

1.4. Palm OS Architecture

The architecture of Palm OS is divided into three main layers from top to bottom: Application, Operating system with services and Hardware. The service is known as Application Programming Interface (API) help to developer to understand the notation for accessing hardware and other layers. In Palm OS the applications have direct control on the processor and functionality instead of APIs. It also increases the compatibility as well as security issues with modern components. A Palm OS architecture is shown below.

1.5. Symbian OS

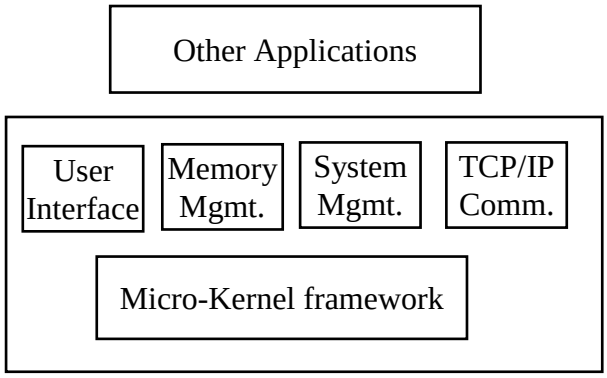


Figure 1.2 Symbian OS services.

Symbian OS is one of the most successful OS from the launching and till the discontinuation. It includes most of the features as per current needs. In 1998 a company Psion is renamed as Symbian Ltd. in collaboration of different popular companies like Nokia, Ericsson, Motorola etc. to provide a common and advance OS for consumers. Interactive user interface is provided by this OS through

AVKON toolkit and provide a keyboard-based activity. Later Symbian shifted from Qt framework to touchscreen mobile. Symbian phones came with the Opera browser which was later replaced by a built-in browser of Symbian OS based on its own web kit named as "Web Browser for S60". The overall development of this OS is based on S60 framework that provided high speed and best interface. For application development a Qt kit with C++ programming languages is used.

Later it can also possible to build app with Java, Python, and Adobe Flash Lite. Symbian OS offered protection against malware and includes the reliable security certificates. Symbian OS is dependent on Nokia.

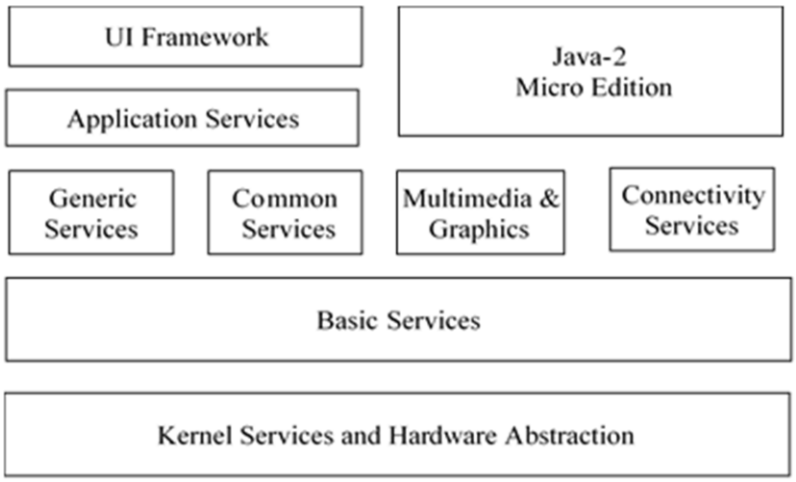


Figure 1.3 Symbian OS Architecture.

1.6. Symbian OS Architecture

The architecture layers of Symbian OS are described one by one:

UI Framework

It is also a topmost layer of Symbian OS framework. currently three custom user interfaces are available: MOAP, S60 and UIQ.

Application Services

The top most layer of Symbian OS architecture provides framework libraries for constructing a graphical user interface. This layer has a hierarchy of class for user interface and concrete widget classes controls components. Some GUI libraries like S60 and UIQ have been implemented and used to extend the functionality of framework layer**.**

J2ME

J2ME stands for Java 2 Micro Edition. it interacts with multiple Symbian system layers. It contains java virtual machine, MIDP and CLDC libraries and plugins for interaction with native layers of OS.

Intermediate Service Layer

Service layer is acting as a middle layer between the base services and application services layer at an upper level. The services of this can be considered in three broad sections:

- ✓ Generic services of OS like task scheduler.
- ✓ Communications services for example calling, network services, short-link services.
- ✓ Multimedia services includes font server, windows server and multimedia framework.
- ✓ Connectivity services includes file browsing feature with PC and software installation.

Base Services Layer

This layer encapsulates frameworks, libraries, servers that are designed on the kernel layer to provides the facility to upper level of operating system services like programming library, file server, persistence data model and cryptography library.

Kernel Services & Hardware Abstraction

This lowest layer OS contains the kernel and its components to make interaction with underlying hardware. It involves various digital components like logical and physical device drivers, mutexes, timers, scheduler and interrupt handler etc. For Symbian OS portability at new hardware a kernel layer is customized.

1.7. Android Operating System

Currently, android is the most popular mobile operating system, with billions of devices actively available in market. It gives chance for unified approach to develop application for mobile devices. This means this development gives power to android with endless possibilities. The application is compatible with other android based system like android powered TV sets or android car systems. This is why it makes our life to make easy every day process or anything whatever is in mind can be implemented in app for any domain that they might run.

Android OS is currently developed by Google by using the Linux kernel and primarily designed for touchscreen mobile devices such as smartphones and tablets. Android is an open-source operating system named Android. Google has designed and shared the code for all users with the low-level "stuff", the needed middleware to power and use an electronic device who wants to write code their own code to make OS from it.

Android architecture is a collection of various components to support any android device needs.

- Android software is an open-source Linux Kernel.
- It is having collection of C/C++ libraries.
- These are used through an application framework service.

Among all these components Linux Kernel provides main functionality of operating system. OS functions for smartphones and Dalvik Virtual Machine (DVM) provide a platform for running an android application.

1.8. Android Architecture

The main components of android architecture are following: -

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

Applications

The top most layer of the Android architecture is the applications layer. Few standard or popular applications are pre- installed with every new device such as SMS social app, caller unit, web browser, contact manager etc. Third party apps can be installed by replacing or new with the existing app.

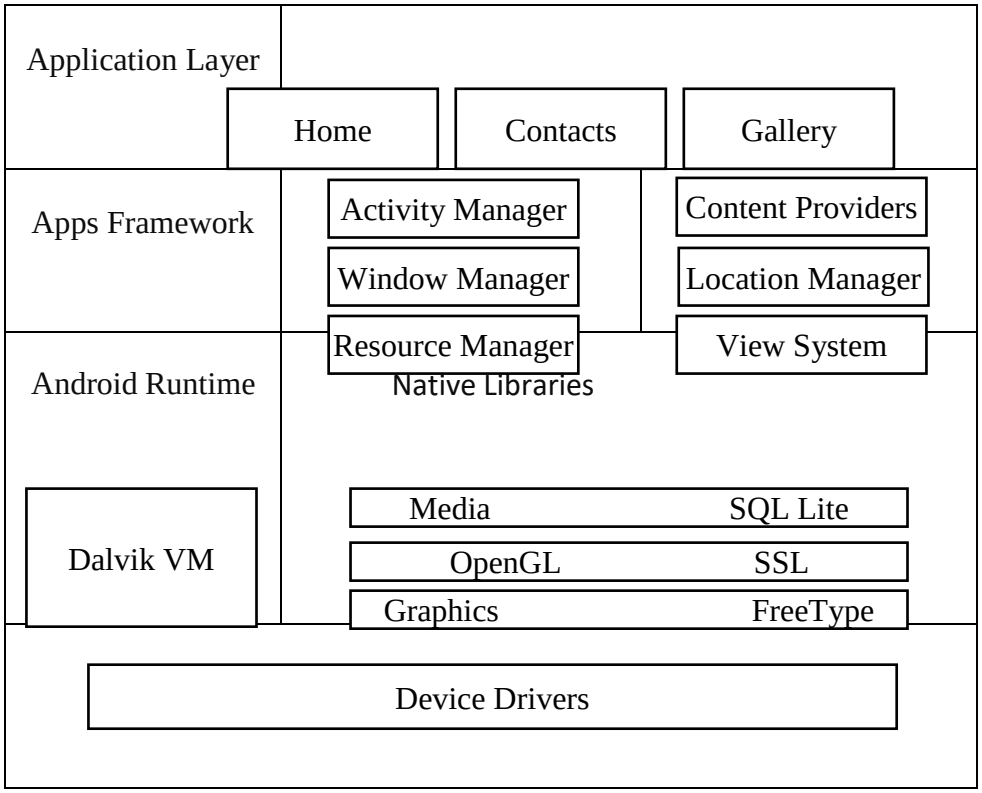


Figure 1.4 Android OS Architecture.

Application Framework

This topmost framework layer provides higher level services or large number of useful APIs to applications using java classes. Most of the Apps developers are can access these services in their newly developed applications.

Main blocks of Application framework are:

Activity Manager: The life cycle of applications is managed by this section.

Content Providers: Sharing of data and managing the accessibility of data from different applications is performed by this block.

Telephony Manager: this manages all voice call related functionalities.

Location Manager: Location details of device can be seen using GPS.

Resource Manager: It manage the various types of kernel layer resources used in various application.

Android Runtime

Android runtime layer is consisting of core java libraries and Dalvik virtual machine. Dalvik virtual machine is just like JVM of java which help in running the applications on android device. The Dalvik VM also enables every android apps as a separate space to run in its own process of DVM. Dalvik VM supports in creating multiple instances of virtual machine to provide security, threading support, isolation and memory management. As JVM is process based while Dalvik VM is register base. Dalvik VM runs the. dex source files generated from .class file by dx tool of android SDK. DVM is fully optimized to work in the environment of low processing power and memory environments.

Platform Libraries

At the top of Linux kernel layer, libraries layer available. This layer makes the device enables to handle various types of data. Data storage and retrieval pattern is specific to hardware. All the layer that we discussed in OS architecture are written in C/C++ language and called through java interface.

Few important basic libraries are:

Surface Manager: it is used to manage the device display.

SQLite: SQLite is used to make connectivity with the database for data storage in android. This is a relational database and available to all applications for data.

Web Kit: A lower-level engine used for display HTML based content

Media framework: The features provided like playbacks and recording of audio, video and picture formats.

Free type: bitmap and font type Rendering features

openGL/ES: used for rendering 2D or 3D graphics content to screen

libc: This provides system related libraries.

Linux Kernel

Linux Kernel layer is at the bottom of technology stack. All processing layer of this OS is based this layer with some modifications done by the Google. Like it provides the following functionalities as general: memory management, Process Management, Device management (keypad, camera, I/O,

display etc.). Android system communicates with the hardware with this layer of the device. Device driver are also available on this layer. Advance feature of this kernel layer is to manage virtual memory, power management, network and drivers.

1.9. BlackBerry OS

BlackBerry UEM (unified endpoint management) is a solution that provides a comprehensive multiplatform device, application and content management based on integrated security and connectivity. This UEM help to manage other mobile devices. It also helps to manage the data of mobile devices of an organization for secure transmission, secure mailing and content transferring.

BlackBerry Infrastructure is a system of global private data network which is distributed across multiple regions. It secures data which is ready for transmission between organizations and millions of users around the world. The transfer of data between BlackBerry devices and end-user devices is managed efficiently.

BlackBerry Infrastructure system registers the user details for new device activation and validates licensing information for BlackBerry UEM. Dynamics NOC is a service of Blackberry devices known as network operations centre that serves a secure communication between BlackBerry apps and BlackBerry UEM and the BlackBerry EMS (Enterprise Mobility Server). UEM always create a direct connection to the BlackBerry Infrastructure over ports 3101 and 443 for any notification and update. So sometime when there is an error, we may find these port error in blackberry mobile devices.

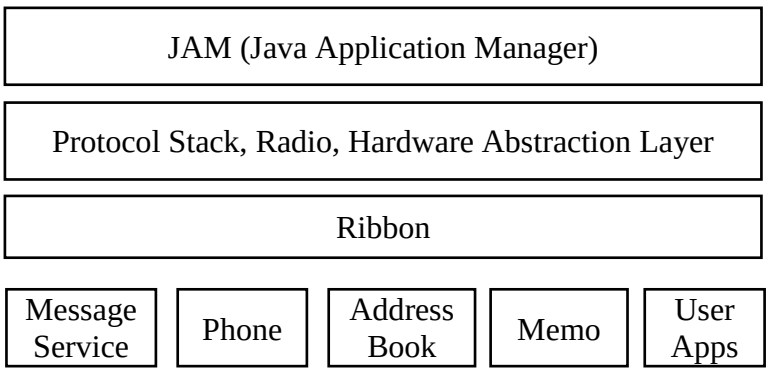


Figure 1.5 BlackBerry OS Architecture.

1.10. BlackBerry OS Architecture

The four layer of blackberry architecture are explained below:

Java Application Manager

This layer help to execute applications which has been downloaded from the network dynamically. After downloading what actions, we performed on application like installing, launching, inspecting and uninstalling all are the part of process as application management. In mobile OS management of application is going under different situation as it has limited resources as it may lack even basic features such as a built-in file system.

HAL and Radio

This hardware abstraction layer is working for software defined radio (SDR) framework. A HAL software device in a processor of a given SDR framework processes data and exchange messages between waveform (FPGA) of hardware/software components. The custom and core modules individually help in reduction of overall cost of developing HALs for software radios at various platform.

Ribbon

This layer help to execute applications which has been downloaded from the network dynamically. After downloading what actions, we performed on application like installing, launching, inspecting and uninstalling all are the part of process as application management. In mobile OS management of application is going under different situation as it has limited resources as it may lack even basic features such as a built-in file system. Ribbon is offering an application launcher for blackberry application. Each apps are recognised thought he icon and OS send a message to the app after it launch successfully.

Lower layer services

Following are the basic API services:

Messaging: This selection of later provides the functionalities to going through the all step included to send a message.

Phone: this API providing a feature to dial out services.

Address book: contain contact database and manage the user information.

Memo: API is used to activate the emulator to verifying the app that is under designing.

User apps: this API is managing all app requesting to use hardware resource or to fulfil their task.

Database: this API help the developer to handle data for new application designing. The API manages the flash file system and memory in use.

1.11. Firefox OS

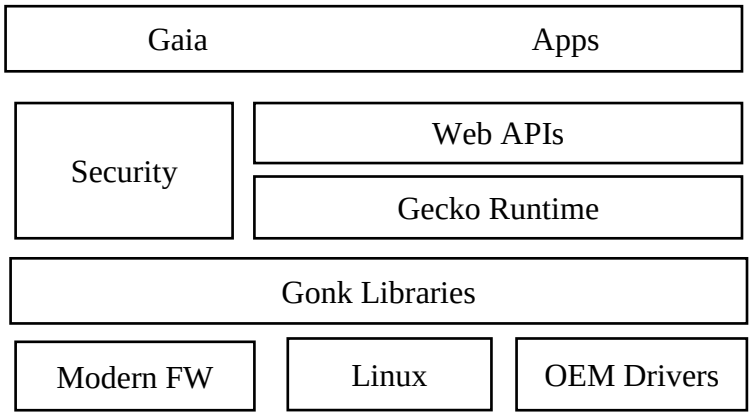
Firefox OS is available in market and enable users to use lower-cost devices which offer a rich set of built in features with social integration of Facebook and Twitter. The first mobile phone devices are Intex Cloud FX and Spice Fire One Mi FX-1 available in Asia. The positive feedback of customer motivates the developer to design an interactive Firefox OS. With the help of Intex and Spice companies, Firefox OS has decided to redefined the low cost-based smartphones for entry-level smartphone

The first launch of Intex Cloud FX mobile based on Firefox OS attained a new era of the Indian smartphone market and proud to be the first Indian company to understand and deliver on market needs. Firefox OS is also Linux kernel-based OS designed for smartphones and supporting devices.

This OS is designed to by the open source community of the Mozilla browser. The OS designing and development process is entirely based on open standards and non-proprietary system. One interesting thing is that it is a browser-based operating system developed with C++, CSS, HTML5 and JavaScript. A web-based user interface runs the Firefox OS stack, which includes the layer known as Gonk. Gecko is a layout engine read and renders it for the user. The system is designed to work with HTML5 apps.

The first variant of Firefox OS was released in 2013 for low-end phones to make accessible for people. Moreover, developers addon advance features to make this more competitive with existing OS such as Android and iOS. Then Alcatel, Huawei, LG, Nexus and ZTE vendors have involved the Firefox OS devices.

Figure 1.6 Firefox OS Architecture



1.12. Firefox OS Architecture

All the four sections of layer are described in this section.

Gonk Layer

Gonk layer is consist of Linux kernel and software libraries. It also includes user space as hardware abstraction layer. Gonk layer is based on a simple Linux distribution. It includes mechanisms of Android system. Mozilla developers stretched the layer services to assimilate the services at the Firefox OS architecture. Gonk layer implies a fusion of various open source software, hardware and OEM dependent components. It uses libraries for android like GPS, camera, etc. and open source projects like libusb, bluez etc. Radio Interface Layer communicates with modem hardware in the mobile phone. Gecko also communicates with the media server with android RPC stack.

Gecko Runtime

It is assumed as web browser engine of Firefox OS. Various applications are using this as a service and developed by Mozilla Corp. Its target to help in internet standards and opening of web pages of different applications. Gecko provides application interfaces with rich programming API. The following Gecko standards are: CSS, DOM, HTML4 (HTML5 partially), JavaScript, MathML, RDF, XForms, XHTML, XML, XSLT and Xpath.

Gaia

Gaia is a topmost layer for the user interface of Firefox OS. It is indicating what we actually see in the OS. All the front screen visibility like app launcher and all the default applications such as camera, dialer, and other settings is a component of Gaia. All these are completely designed using HTML, CSS and JavaScript. This is the final and user end layer of Firefox OS. Gaia provides the WebAPIs for app developers to interact with Gecko layer and the underlying hardware layer. The Firefox OS is openly available as open source on GitHub. it gives full access to the developer to configure the OS as per their personal needs

Firefox OS applications

The third-party applications may be download (Firefox Marketplace) and install in the Firefox OS which will run on Gaia.

Mobile hardware

This layer refers to the various hardware components present in a mobile like battery, camera, GPS, Bluetooth, sensors etc. It is important to interact with the operating system with the device and environment. A good interaction is assumed as the part of best feature phones.

1.13. iPhone OS

iPhone OS is also a smart phone OS developed by Apple and specially distributed for Apple hardware components. This OS is currently supporting many mobile companies like iPad, iPhone and iPod touch etc. It is considered as second OS as mobile operating system in the world, after Android. Every variant of android is found fully in competition due to less cost. But iOS stand at second position. After sometime apple also shown the collection of iOS application in Apple store.

The iOS user interface is based on the concept of direct manipulation, having number of features: multi-touch gestures (swipe, tap, pinch, and reverse pinch), Interface control elements consist of sliders, switches, and buttons, Internal accelerometers (respond to shaking the device) rotating from portrait to landscape mode. iOS shares some frameworks such as Core Foundation and Foundation Kit with OS X. Its UI toolkit provides the Cocoa Touch rather than OS X's Cocoa for AppKit framework. It is therefore not compatible with OS X for applications. Unix based shell access is not provided to users and are restricted for application. The final versions of iOS are released annually.

The iOS is constructed with four major abstraction layers: Core OS layer, Core Services layer, Media layer, and Cocoa Touch layer.

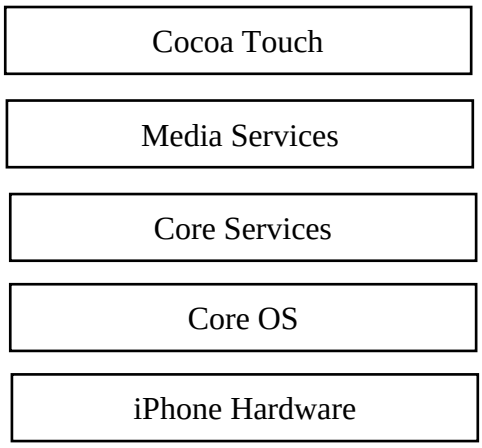


Figure 1.7 iOS Architecture.

1.14. iOS Architecture

The iOS architecture contains layers between the top and lower layer so that they do not make communication directly. The features at the lower layer is to provide basic services and at the higher layers to serve as user interface and graphics.

The layered architecture explained as follows −

Core OS

The low-level technologies features are available at Core OS layer. These techniques are Bluetooth, External Accessory support, accelerate service, security services, local authorization etc.

Core Services

There is collection of many techniques available are core services layer. Name of few are: Cloud kit for moving data between App and iCloud. Core Foundation offers data management and service to the iOS apps. Core Data is a one section the MVC model to handle data. Address Book give an access to the contacts of user.

Core Motion is used handle motion-based data on the mobile. Health kit collection health related information of the user. Core Location save information of various App like location and heading information.

Media

The media layer enables the features like graphics, audio and video technology of the system. different subsets are used: UI kit help in designing images and animation to view the content. Core Graphics supports vector and rendering of images and work as a native engine for drawing. AV kit offer very flexible interface to the user for video presentation.

Cocoa Touch

The cocoa touch also provides number of sub components of different services. Event kit offer standard system interfaces for viewing and set/reset calendar-based events. Game kit is used to share game related information online through game center. Map kit provides map application with scrollable features a scrollable map which can be included into the app user interface.

1.15. Window OS

Window OS version for mobile is based on Window CE kernel. Initially, Pocket PC 2000 was appeared as Mobile OS. It a set of basic applications that uses APIs of Microsoft windows. The design of this OS is almost similar to Desktop windows.

Nokia is using the Window Phone as OS in Mobile. There are different types of variant like Window Phone Mango, Tango and simple. Recent version of Window OS includes these features DivX/XviD video playback support, NFC support, Bluetooth file transfers, Kid's Corner, multi-core processor and multiple screen resolution support, removable memory card support and USB Mass storage mode.

The replacing of WinCE-based architecture to a WinNT kernel, allowing easy applications porting from the desktop Windows 8 to the mobile one, mainly benefiting the tablets running the new Windows 8 RT. Windows Phones has two distinct modes first is Windows Mobile Standard and second is Windows Mobile Professional.

Windows CE directly provide support to industrial controllers, Internet appliances, point of sale terminals, cameras, cable set-top boxes and communications hubs. For imbedded system we can customize this Window CE developers can build customized Windows CE operating systems as well as components for embedded systems.

Few sections of the Windows CE are offered in source code so that later it can be used to enabling hardware vendors to modify as per the configuration of their hardware.

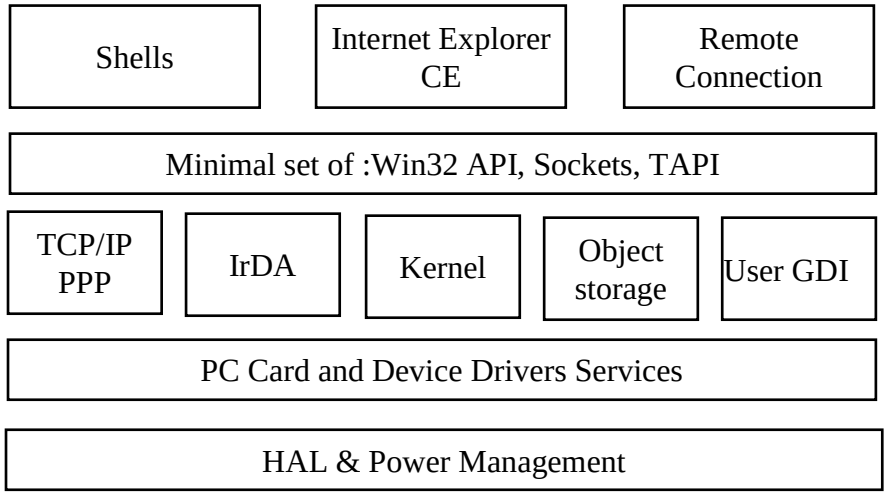


Figure 1.8 Window CE OS Architecture.

1.16. Window CE Architecture

The architecture is set of various layers or components such as Hardware abstraction layer & power management, device drivers, PC card services, Kernel Windows CE, User & GDI, databases and file systems, IrDA and TCP/IP protocols, APIs, remote connectivity, shell and internet explorer. Other embedded systems not included all of the elements discussed above. The detail descriptions of the components are:

HAL and Power Management

The HAL allows to design an embedded system to run Windows CE on hardware platform. HAL provides power management functions for hardware components. Windows CE does not have memory mapping and interrupt process as in PC. A minimal interrupt routine is implemented to run operating system in limited hardware configuration. Windows CE power functions allowing instant on/off feature for Windows CE devices if the device has non-volatile RAM.

Device Drivers and PC Card Services

Windows CE can support two types of device drivers: built-in drivers for permanently existing devices and run-time installable drivers for devices that are externally attached as plug-in. Windows CE by default support the following devices such as mouse, touch panels, ports (serial/parallel), modems, displays, PC Card, audio, speakers and keyboards.

Kernel

The Windows CE kernel is specially designed for smart devices. It also implements the various processes: Win32, thread, and virtual memory-based model. As in Windows NT it also supports a pre-emptive, priority-based scheduling, synchronization primitives. moreover, it including events, mutexes and semaphores. Kernel executes the program in RAM or ROM and uses demand paging concept. The kernel layer also accesses the interrupt service routine as well as low thread latency for a process of taking time less than 100 microsecond. Windows CE kernel permits for many types of real-time systems.

USER and GDI

The USER and GDI are two separate components to provides the facilities and functionalities for user interface. A grayscale display system is acceptable by GDI. Windows CE uses the same Win32 features for USER layer like event management, overlapping windows, user interface controls, dialog boxes, interposes communication etc. USER also supply the features of globalization as UNICODE character interpretation and locale NLS APIs. GDI and USER are very flexible to adjust the various functionalities.

Object Stores

Object Store is component of intermediate layer that provides persistent data and storage management. Persistent type of data contained in non-volatile memory such as flash memory and battery-backed RAM. Embedded system designer can set the amount of RAM used for object storage at runtime. The object store structure is built-up from three classes: file systems, registry and databases. Windows CE uses three types of file systems for mobile OS: ROM-based, RAM-based, FAT file system for storage drives, flash memory and SRAM. Win32 service allows to access these file systems in Window CE OS.

TCP/IP, PPP, and IrDA

This communications protocols provides connectivity to desktop PCs through Internet, and Windows CE devices. The protocol stack used for connections to the standard Internet protocol TCP/IP which is coupled with PPP. The TCP/IP and PPP protocols are used when direct cable connection is there. Windows CE layer stack also embed the standard infrared stack named IrDA.

APIs

Windows CE OS imports a subset API from the Win32 of PC. This offer most used Win32 APIs which is around 500. Many majorly contributed applications are designed for Windows CE using this subset. The major apps are pocket Office and PIM applications for the handheld PC. This subset of API is not strictly limited it allows the number of services like communications protocols APIs, Windows Sockets, TAPI, and Uni-modem, Sockets API is interface for TCP/IP and IrDA communication. TAPI and Uni-modem offer dialing features by modem-based applications.

Remote Connectivity

Windows CE enable the feature to connect remote access API (RAPI) to Windows PC over the Sockets. This service help to update the object stored in Window CE/PC both ways. different task can be performed through remote connectivity as databases updating, viewing registry and modification over the connection. TAPI functions also help in dial out can be completed remotely over a phone line.

Internet Explorer for Windows CE

Internet explorer is used to access the internet services for the Windows CE operating system. Win CE internet explorer is just similar to win-PC internet explorer that is used to explore the information from the Internet through various types of URLs and FTPs.

1.17. MIPS and ARM Processor

Computer processors were investigated in the 1937. There are few processor manufacturer industries are AMD, IBM, Intel, Motorola, Qualcomm, Samsung etc. A small size silicon-based chip is used to make the processor. It is fixed inside the device to perform the operation within few Nano seconds and processing is measured in the term of megahertz. The processor is based on four primary

segments of instruction like fetch, decode execute and write back to the memory. In every processing unit like mobiles, laptops, computers, washing machines are using processors.

Definition of processor

The processor a set of instruction that process the specific task to a particular computer. The processor is consisting of Arithmetic Logic Unit and Control Unit where ALU performs mathematical operations such as additions, multiplications, subtractions, divisions etc. and the control unit manages the operation of the instructions. The processor also interacts with the other components like i/o and memory devices.

Types of Processors

Various types of general-purpose processors are used like embedded, microcontroller, microprocessor, DPS and digital processor.

For mobile phone two types of process are mainly used MIPS and ARM.

- MIPS architecture-based processor is used in making smart phones, embedded systems (router and gateway), supercomputer and Sony PlayStation.
- Both MIPS and ARM are having their own different instruction set architectures from the family of RISC Architecture.
- The instruction sets for both architectures are fixed and same size whereas ARM consist of 16 registers while MIPS consist of 32 registers.
- ARM working with high efficiency than MIPS as ARM processors support 64-bit data buses among the core of processor and the caches.
- ARM provides only general-purpose registers for arithmetic operations while MIPS provides two separate registers to hold the results of multiply operation.

- ARM architecture was invented in 1985 while MIPS was in 1981 both are RISC instruction set architecture and ARM have Memory addressing: The ARM and MIPS architectures use byte addressing to access operands of memory.

- Addressing modes: MIPS has three addressing modes: register, immediate and displacement while MIPS uses three and three different variants of displacement type addressing as per x86 Architecture. Latest architecture is supporting multiple addressing mode.

1.18. Challenges of the mobile platform

Nowadays various industries are realizing the benefits of well-developed App in their business. Application is become a best option to increase business due to the mobile in every hand. there is a high demand of mobile app development. Several app development companies offer number of features in Apps. During the designing of apps there are various issues and challenges depending on the developer experience and companies.

Therefore, in the growing period of application, for developers it is important to know what services is best or not. The following are the challenges:

Interactive apps

An interactive app may interact the customer in a great way. App with feature like built-in sensors and easier data processing.

Involving end user

There should be suggestion of end user for the designing of app. It might help better in popularization when launching app.

Keeping App simple

The design of app should be quite simple that can be easily understand by the user.

this can be attained by implementing a simple instruction, tutorial and structural guideline.

Power consumption and performance

App should use the minimum resources, release resources after user in idle stage. If your app using the required set of services then it might help in less consumption of power.

Marketing the app

Overall success of app is depending on the marketing. as we know the 80-90% marketing and 10% efforts are applied on the development of App

Selection of development technology

Selection of an appropriate development technology is the main key for the success of app. The need of app tells better weather it is developed for native, hybrid, or a cross-platform.

Developing a world-class application

In market there are hundreds of apps available. We need to develop an app that stands out uniquely. However, if we want to design an app for our customer, there should be intuitive design, required features and graphics for ultimate experience of users.

Cross-platform apps

To design an app and checking on single operating systems or mobile phone is not only final version for developing apps. This is because of various model and devices are available in market. To keep this in mind developers should adopt app for cross platform running.

Features of the App

Innovative design always helps to increase the visitors or users. The designing of app must be focused on the user-friendly features, recognizable designs, color combination to attract the attention of user.

Security

Security issues are most important for developers and are new real challenges in mobile app development. Apps should be free from any direct access and free from malwares.

Overall, each application needs a lot of attention during the development process. You should make sure that you are developing superior apps for huge success and better solutions.

1.19. Summary

This chapter discussed the various types of operating system used for small mobile and tablet devices. Android OS is explained with their architecture. The background study of the android framework. It also covered the Hardware abstraction layer used for OS architecture and processor. Section 1.18 discussed the various challenges of android application designing.

1.20. Keywords

Android OS - An open-source operating system developed by Google for mobile devices such as smartphones and tablets.

Java API - A set of classes and interfaces that allow developers to create applications for the Android platform.

Linux Kernel - The core of the Android operating system, which provides basic system functionality and manages hardware resources.

SDK - Software Development Kit, a set of tools used by developers to create applications for the Android platform.

APK - Android Package Kit, a file format used to distribute and install applications on the Android platform.

AOSP - Android Open Source Project, an open-source project that provides the source code for the Android operating system.

ART - Android Runtime, a new runtime environment introduced in Android 4.4 (KitKat) that replaces the Dalvik Virtual Machine.

Binder - A system service that allows applications to communicate with each other on the Android platform.

Intents - A messaging system used by applications to communicate with each other on the Android platform.

Widgets - Small applications that can be placed on the home screen of an Android device.

Fragmentation - The phenomenon of different versions of the Android operating system being used on different devices.

Security Patch - A software update released by Google to fix security vulnerabilities in the Android operating system.

1.21. Check your Progress

1. Android is a_____.
2. Android is mainly developed for_____.
3. Android is developed by_____.

4. Latest version of Android is_____
5. _____ enables to store the data in a structured manner in and void.
6. Android logo is named as _____.
7. API stands for_____
8. Android is based on _____ language
9. APK stands for_____.
10. _____ virtual machine is used by the Android operating system.

1.22. Self-assessment questions

1. Write a short note on:
   a) Buffering and spooling
   b) System calls
   c) System programs
2. What is operating system? Discuss various services provided by operating system.
3. What do you mean by multi-programming, multi-user and multi-processor OS?
4. Differentiate between iPhone OS and Android OS.
5. What are the different Challenges in Android application development?
6. Compare MIPS and ARM processor.
7. Explain COCOA touch.
8. Describe Gonk layer. Also explain Mozilla Gecko runtime.
9. Discuss Android framework libraries with their classes.
10. What is J2ME? Explain it major components.
11. What is the latest version of the Android operating system?
12. What are the main features of Android?
13. How can I customize my Android device?
14. What are the security features of Android?

1.23. Answers (Section 1.20)

1. operating system
2. mobile devices

3. google
4. Android 13
5. SQ lite
6. Bugdroid
7. application programming interface
8. .java
9. android package kit
10. Dalvik virtual machine

1.24. References/ Suggested reading

1. Android Programming for Beginners, John Horton,2nd edition, .Packt Publishing
2. Android Programming with Kotlin for Beginners, John Horton, 1st edition, Packt.
3. Head-First Kotlin, Dawn Griffiths, 1st edition, O'Reilly,
4. Android App Development, Michael Burton, 3rd edition

SUBJECT: MOBILE APPLICATION DEVELOPEMENT

COURSE CODE: MCA-42	AUTHOR: Dr. Sunil Kumar Verma
LESSON NO. 2	

Android Framework

STRUCTURE

LEARNING OBJECTIVE

Student will learn about Java Virtual Machine, Delvik Virtual Machine and its core libraries. Then student read elements of AndroidManifest.xml, Software stack, Linux kernel, security of Framework, Android runtime and core building blocks used to design an Android application. In last section ends with basic example of android application.

Chapter 2. Android Framework

2.1. Introduction

In java programming language, JVM is a high-performance framework and provides better memory management. However, we need to optimize the JVM to make it supportive for handheld and low power devices. The Delvik Virtual machine (DVM) is designed to provide the environment for android operating system to run on mobile devices

The design of this DVM is focused on optimization of the various services like battery life, memory and performance of VM. The Dalvik VM was written Dan Bornstein. The compiler Dex converts the .class files into the .dex file that is compatible with the Dalvik VM. Using Dex compiler multiple class files can be converted into one .dex file.

The flowchart is showing the compilation process of DVM. The javac command compiles the **.java** source file into the .class file. The. dex tool of DVM takes all the .class files of application and produce a single .dex file. Dex is a platform-specific component of DVM. The **aapt** stands for android assets packaging tool that handles the packaging process during .apk generation.

A detailed discussion about the use of Java Virtual Machine and Dalvik Virtual Machine. With the android 5.0, the DVM has been replaced by ART (Android Runtime).

Dalvik

The DVM is designed to run on various platforms. The variant UNIX OS support and running the GNU C compiler. Both the little-endian and big-endian CPUs have been exercised and supported implicitly and explicitly.

Core Libraries

The native code of the core libraries like dalvik/libcore or dalvik/vm/native is written in C/C++. These libraries are expected to operate without changing in the Linux environment. The core libraries using code from different projects including zlib, OpenSSL etc.

JNI Call Bridge

Most of the code for DVM runtime is written in C (portable). JNI call bridge is one non-portable part of the android runtime. This help to converts an integers array as arguments of a function with different types and calls a function. this process is similar to C conventional calling mechanism. The open-source foreign function interface library is used when a custom bridge is not available.

Interpreter

Dalvik runtime uses two types of interpreters named "portable" and "fast". The portable interpreter mostly working in a single C function and compile with any system that supports GCC. If your system doesn't have GCC then disable "threaded" execution model. For this move on the GCC "goto table" implementation and search THREADED_INTERP to modify value.

The fast interpreter uses assembly fragments written by the developer. If the current architecture doesn’t support it then internally system will generate an interpreter out of C "stubs". The resultant stubs are little bit slower than the portable interpreter. The services of fast interpreter are enabled automatically. The platform that doesn’t supports native code may need to switch on the portable interpreter. The setting of interpreter can be controlled with the dalvik.vm. execution-mode available in system property.

The code segment is:

>adb shell "echo dalvik.vm. execution-mode = int:portable >> /data/local.prop"

>reboot

Then the VM will start with the enabled portable interpreter.

Interpreter Switching

The DVM is using third type of interpreter for implementation known as the debug interpreter. This is an advance variation of the portable interpreter and includes debugging and profiling features.

After configured an interpreter with debugger attaches and enabled profiling feature then the VM will switch interpreters based on the convenient point. All these happen when GC is checking for the safest point of their different thread with exception. Similarly, when the services of debugger detach or profiling is disabled then the execution transfers again on "fast" or "portable" interpreter automatically.

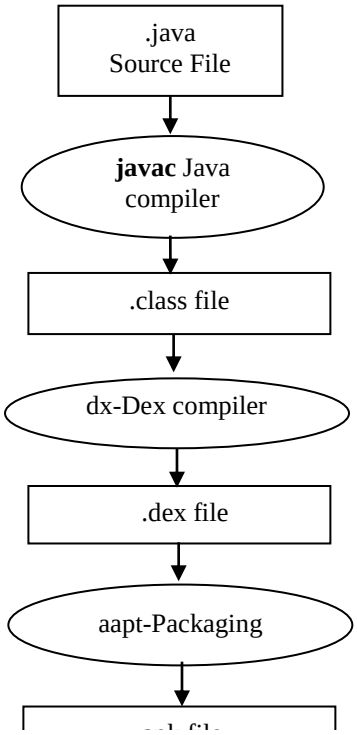


Figure 2.1 Source code to .apk file processing.

2.2. AndroidManifest.xml file

This file contains various information of used packages, components of the application under design such as activities, broadcast receivers, content providers, services etc.

The following jobs are performed by this file:

- **It protects the application** from outside access for any protected parts until providing the permissions.
- It includes **the android API** used in the application.
- It **provides the list of documentation classes**. These classes reveal profiling and other information. When the application is published before that this information are removed.

So, this is an important xml based file for every android application available in root directory of the app.

A snippet of AndroidManifest.xml file is:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.sunilverma.hello"
  android:versionCode="1"
  android:versionName="1.0" >
   <uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="15" />
   <application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
      android:name=".MainActivity"
      android:label="@string/title_activity_main" >
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
         <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

Figure 2.2 Elements of AndroidManifest.xml file.

Elements of AndroidManifest.xml

The elements are described as

<manifest>

It is a root element of the AndroidManifest.xml file. This provides package attribute that describes activity class package name.

<application>

application is a next element of manifest. The namespace declaration is mention here. This element holds various sub elements (**icon**, **label**, **theme** etc.) of the application component such as activity etc.

android: icon- it represents the icon for the components of all the android application.

android: label- for all the application components works as the default label.

android: theme- for all the android activities it represents a common theme

<activity>

activity is must be defined for every component in the AndroidManifest.xml file. It includes various flag attributes such as label, name, theme, launch Mode etc.

android: label- A label displayed on the screen.

android: name Every activity class is represents by a name.

<intent-filter>

intent-filter This is a sub element of an activity describes the type of intent to which different activity, service or broadcast receiver response.

<action>

intent-filter action is defined here. Intent-filter should have at least one action element.

<category>

an intent-filter must have a category name.

2.3. Android R.java file

This android **R.java** is an auto-generated file during android project creation by Android Asset Packaging Tool **(aapt)** that contains the details of every component resource IDs for all the resources available in res/ directory.

If you design or add any component in the file activity_main.xml, ID of every corresponding component is automatically generated in this file. In further this id may be used in activity source file to perform the any task on the component. Sometime, if the user deleted the R.jar file by mistake, android generate it automatically.

Let's go through the R.java file. A lot of static nested classes such as menu, id, layout, attribute, drawable, string etc are included in this file.

```
/* 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.gjuandroid;
 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 {
      /** here comment is written*/
      public static final int AppBaseTheme=0x7f050000;
      public static final int AppTheme=0x7f050001;
   }
}
```

Figure 2.3 A sample of R.java file.

2.4. Android Software Stack

Android OS is consisting of various layers of software Each layer groups several programs Each program has its own services to provide the specific task. Android architecture is a form of the software stack which consist of the four layers. Android consist of a Linux kernel and collection of the C, C++ libraries

These services exposed from the application framework.

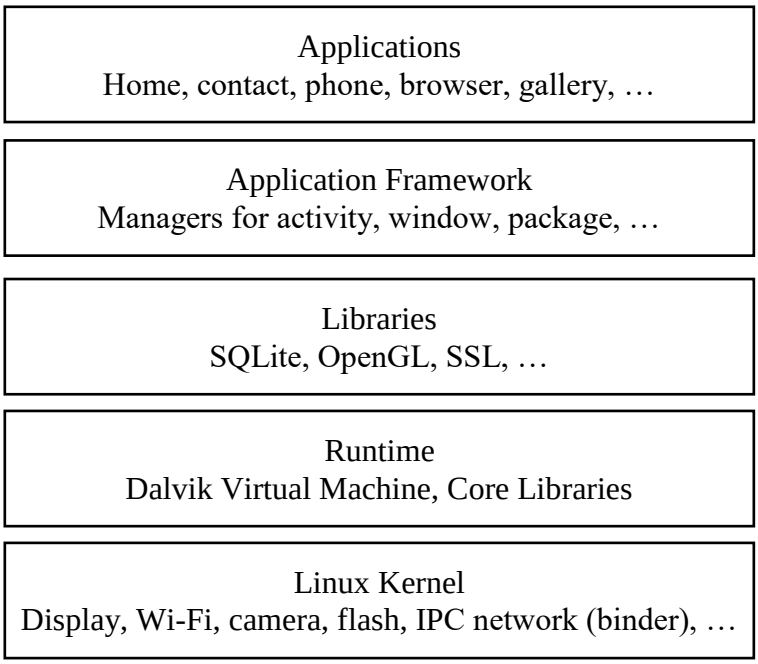


Figure 2.4 Android Software Stack Framework

2.5. Linux Kernel

- Android was created on the open-source kernel of Linux.
- it is heart of the architecture that exist at the root of the android architecture.
- It is responsible for the device drivers power management, memory management, device management, and resource management.

2.6. Security

- Linux kernel provide the security issues between application and the system

- Memory management provide the freedom or implement the app
- Process management: It is used to manage the processes well and allocate resources as per the app needs
- Network stack. It handles the network-based activities or communication

2.7. Libraries

- The libraries are running on the top level of the Linux kernel
- it was developed with various features
- It consists of various C/C++ libraries with numerous of open-source tools
- Some of the open-source tools are: Android Runtime, OpenGL, Webkit, Media framework, Secure socket layer etc.
- It is consisting of libraries of java and ART.
- Open GL This graphics libraries are used to produce 2d, 3D computer graphics for Cross language, Cross platform interface.
- Webkit: this is an open-source web browser engine.
- It provides the functionality to display web content and to simplify page loading.
- Media framework allow us to play audio, video and record audio and videos.
- SSL libraries are used for internet security.

2.8. Android Runtime

- Android runtime layer is a subset of the libraries layer
- it is the third layer of software stack
- it provides one of the key components which is called Dalvik VM
- It is act as a JVM which specially designed for android
- It consumes less memory and provides fast performance

2.9. Application framework

This layer provides many high-level services to application in the form of the java. Android framework includes the following key services name as activity manager (controlling all aspect of application life cycle, and activity stack), window manager, viewing system, content providers (Allow

to push and share data with the other application), Resource manager(it provide access to non-code embedded resources such as strings, color setting, user interface layout) , location manager (provide access to the location services by receiving an update about location), notification manager(Allow to display alert message and notification to the user).

Application

It is available on the top of the layer at the application layer we can write our own app and, install it and use it. These applications are interacting with the users. Examples are games, message, contact, phone and other components.

2.10. Android Core Building Blocks

Android is a software kits which is primarily design for mobile device and tablets with touchable screen. It is a modified version of the Linux kernel including few minimum required features and other open-source software package. The fundamental components of core building blocks of android are activities, intents, content providers, fragments, views, services and XML based manifest file.

Every android application is a collection of various blocks to help developers to maintaining the things in an organized manner. This also provide the flexibility to arrange the assets, pictures, animations, movie clips, and implements user defined local functions. Core building blocks is a piece of code that executing through a set of activities during the life-cycle.

All the components of application are coupled by the manifest file AndroidManifest.xml which describes components of the application and way of interact they do.

Android Core building blocks are-

- Activity
- View
- Intent
- Services
- Content Providers
- Fragments
- AndroidManifest.xml

- Android Virtual Device (AVD)

Activity

An activity is an entry point with single-screen interface for a user. It is found as a subclass of the Theme wrapper class. The onCreate() callback method is used to start the activities.

Views

The basic component of the UI (User Interface) elements in android are comes under the view such as button, label, text, etc. All these are designed or drawn as on the screen that we can see is considered a view.

Intent

It is process with name intent that communicates among various major building blocks. These intents are asynchronous method. They share message to other components and doesn't wait for the completion of task. The intent is mainly used to invoke the available components by these steps:

- Start service
- Launch activity
- Display web page
- Display contacts list
- Broadcast message
- Dial a phone call etc.

Services

This doesn't have any viewing components that visible to user. Service is a section of code running in background as a process. it may run for longer time. It doesn't interact with the user and sometime runs after the application destroyed. There are two types of services:

- starService()
- bindServices()

starService(): It runs in the background till the end of current task.

bindServices(): It offers a client-server based interface that allows to communicate with the services using send or get requests.

Content Provider

The block is most important part of application that handles the management of data & database issues. This is also sharing the data between various components of the application on request. applications on requests. The content provider also helps in storing the data in several ways like in database, files, or over a network.

Fragment

It is considered the portioning of user interface in an activity area. It is a process to display more than one fragment on the same mobile screen. So, it is assumed as a sub-activity.

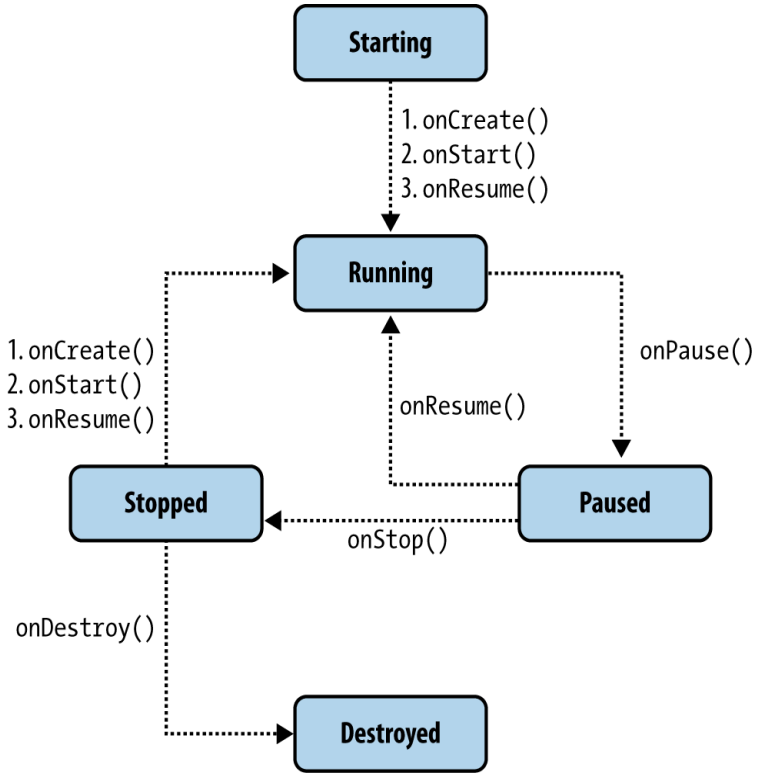


Figure 2.5 Android Service life cycle.

AndroidManifest.xml

This xml file is the most important file of an android application. It contains general and security information about the activities running in app. The file should available in our project as it is mandatory like the web.xml file in Java.

Android Virtual Device

It is a virtual environment of mobile device that is used to test the android application without connecting the mobile or other similar devices. To attach simulator, we can it configure with various setting or version of android OS as per the need of application under design.

2.11. Run apps on the Android Emulator

The Emulator kit simulates the android devices on your laptop or computer. It gives option to test application on a variety of devices and different Android API levels without connecting physical devices. The use of emulator offers advantages like:

Flexibility: With the existing features of simulating with various devices or levels of API. it also rich with the predefined configuration of various types of devices like tablet, android mobile, Wear OS, and Android smart TV devices.

High fidelity: The emulator device shows all the required components and capabilities to feel that you are using real Android device. We can also simulate the incoming and outgoing phone call, message, location setting, simulate different types of network speeds, rotation and other hardware sensors simulating, accessing of Play Store etc.

Speed: When we are testing the application on simulator is much faster and easier as compare to the physical device. You can also transfer data to the emulator devices on high speed rather than device connected over the USB.

Most of the developer are using the emulator as the best option to test the application under development. This topic will cover all the relevant core emulator functions.

Starts with the emulator: The emulator package comes with Android Studio software by default, so we don't have to install it manually. The basic step for the workflow of the emulator as follows:

Create an Android Virtual Device named AVD. Run your application on the emulator device. Move to the emulator. Using emulator, we can perform advance option or implementation.

Emulator device requirements:

Following are the specification of emulator in Android Studio: maximum we can assign the 8GB RAM,64-bit Windows/macOS/Linux operating system and 16GB disk space. If you don't have this specification the emulator may work slower or will not work smoothly. Alternatively, you can use your physical device to run the app.

Create an Android Virtual Device

Each component of the android emulator uses the services of Android virtual device with all the configuration including hardware characteristics. To test the app, you need to install the AVD using device manager. Device manager has both options to connect with virtual or physical devices.

The AVD is working as an independent device that storing everything privately like user data, SD card, and so on. This private memory is the directory of ADV device where it is created. When you click on load/view AVD it will automatically load the relevant directory service from the same directory. You can also connect your device using Wi-Fi connection using pairing of mobile phone. For this the mobile device Wi-Fi debugging option should enabled.

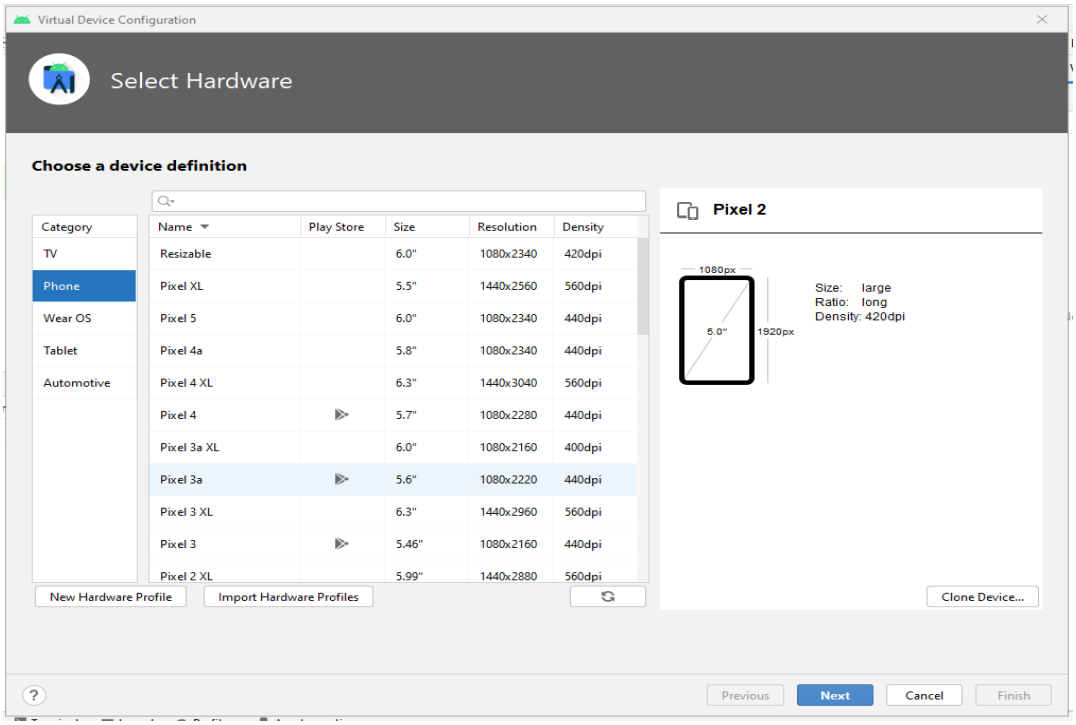


Figure 2.6 Select device configuration for AVD.

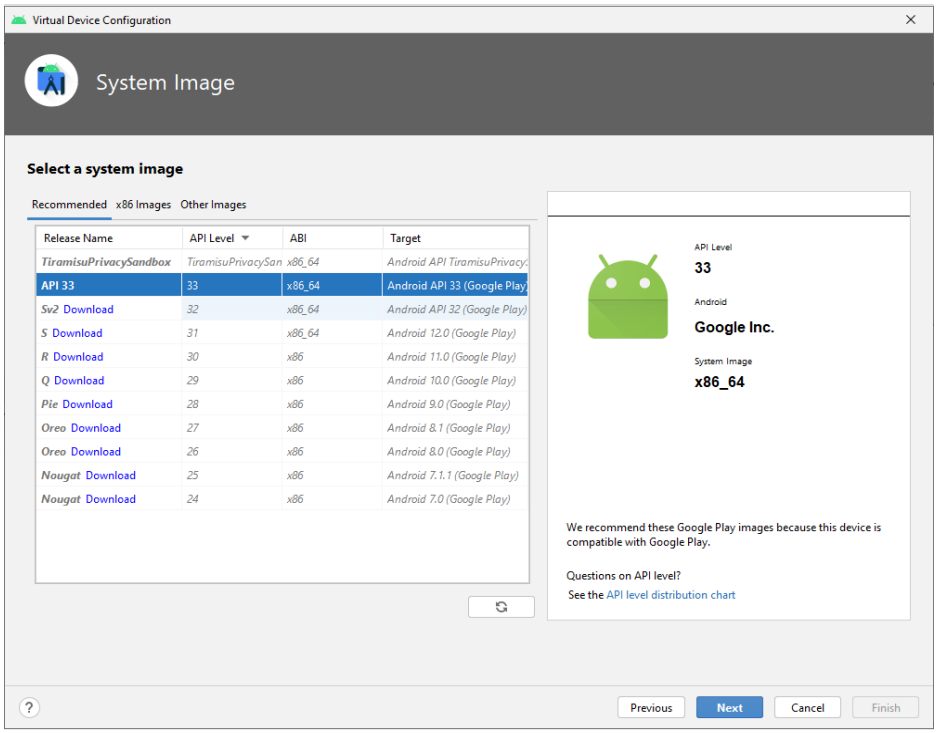


Figure 2.7 Download selected device image.

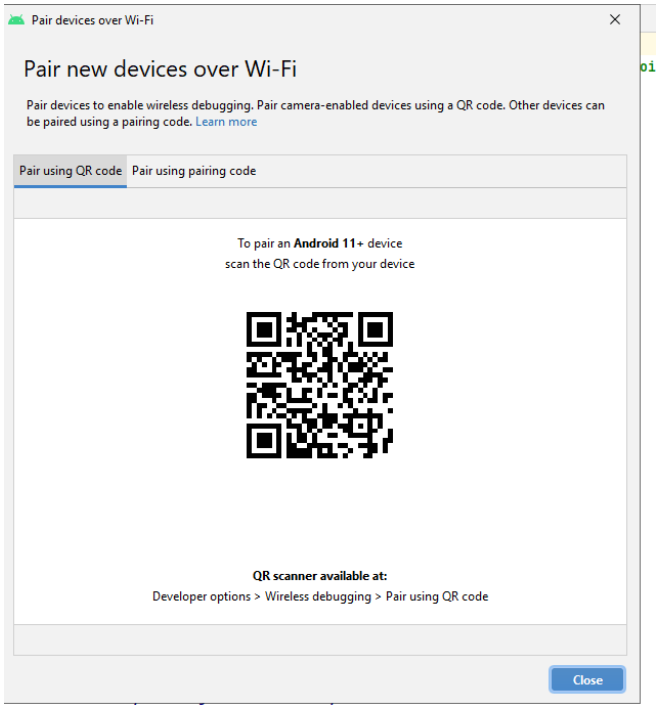


Figure 2.8 Pair mobile using Wi-Fi.

Pairing using the **Wireless debugging** setting on a Google Pixel phone.

2.12. Run your app on the emulator

When AVD ready and in running state to work then you can run your app it will activate the AVD automatically. Emulator is a simple and virtual screen work with android studio/ react native tool. So, you can run the code and see the result using the AVD. However, you may connect with your phone physically through wire or wireless. This emulator device gives all details and features as you do with physical devices. So, most of the developer use emulator instead of engaging your hardware mobile phone for testing mobile application.

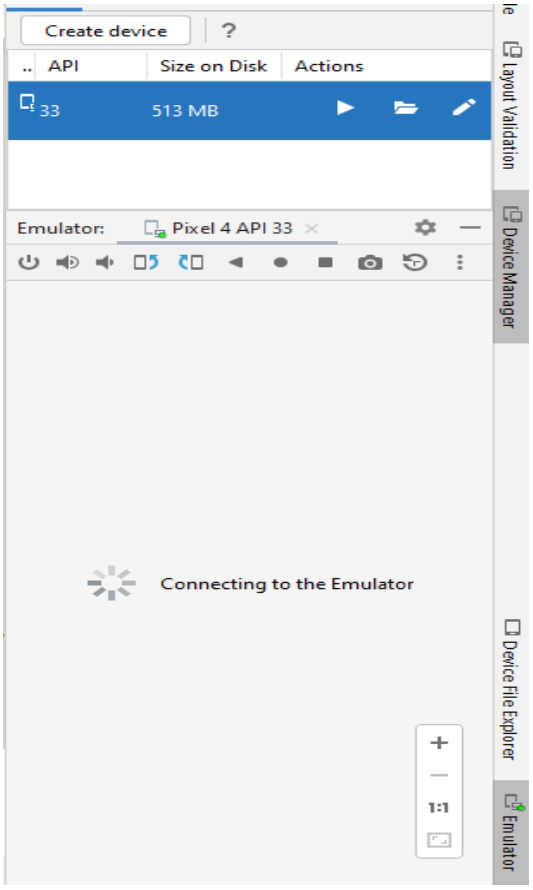


Figure 2.9 Emulator in starting mode.

Tool menu also provide the option to select the SDK or device manager option to start the emulator. First time it makes take few minutes to load the application next time subsequently it will just faster than previous run.

After the loading and installing of your app. Once your app is installed on your AVD, you can run it from the device as you run on physical device. Further you can make modification it will automatically update in AVD when run in android studio editor.

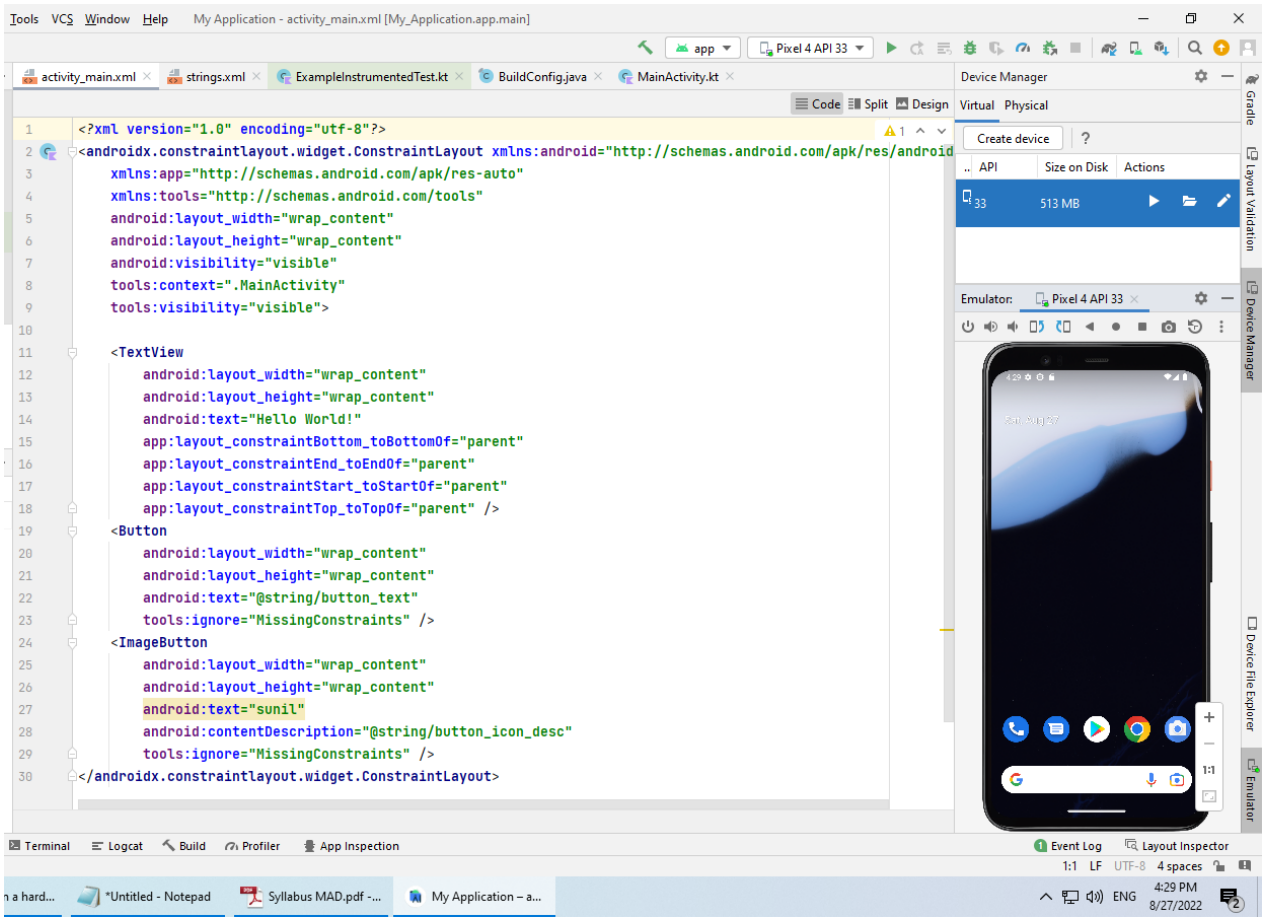


Figure 2.10 Screenshot for android app output with running emulator.

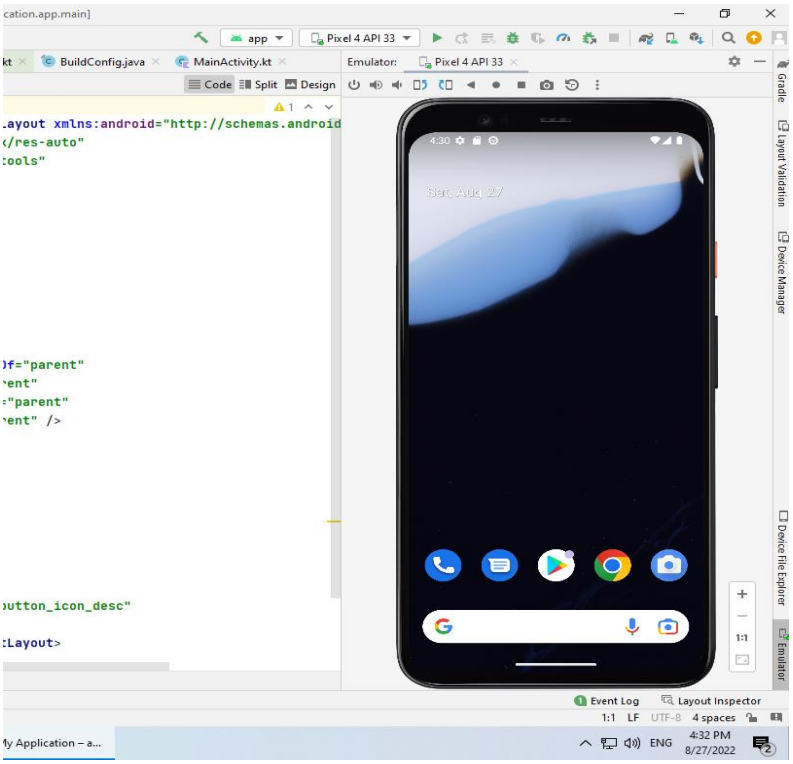


Figure 2.11 Screenshot of emulator.

Navigation the emulator device

During the running of emulator, you can use mouse in replacement of figure that we use in physical mobile device to perform the touch screen operation.

Navigate the emulator screen

To check the application, through mouse we can select menu items and input/ output components like text fields, button and other controls. We can also use shortcut command to use the emulator.

Pressing Control (Command on Mac) brings up a pinch gesture multi-touch interface. The mouse acts as the first finger, and across the anchor point is the second finger. Drag the cursor to move the first point. Clicking the left mouse button acts like touching down both points, and releasing acts like picking both up. Keyboard shortcuts commands can be used to perform common action. Using F1 key in window or command+/ in MacOS for the complete list of the command.

2.13. Hello Android Example

Create new project from file menu then click on new submenu Them select basic activity template for application. Set the name for application and location of the app directory. Select the API for android simulator, if you have installed multiple emulators. Set the name for package and select language.

After finishing the whole step of creating new project or activity configuration, Android Studio will auto generate the activity class and required configuration files. Now the android project is created and ready. Now you can open the project file explorer window and go through all the files. make a modification in activity_main.xml or in fragment_first.xml.

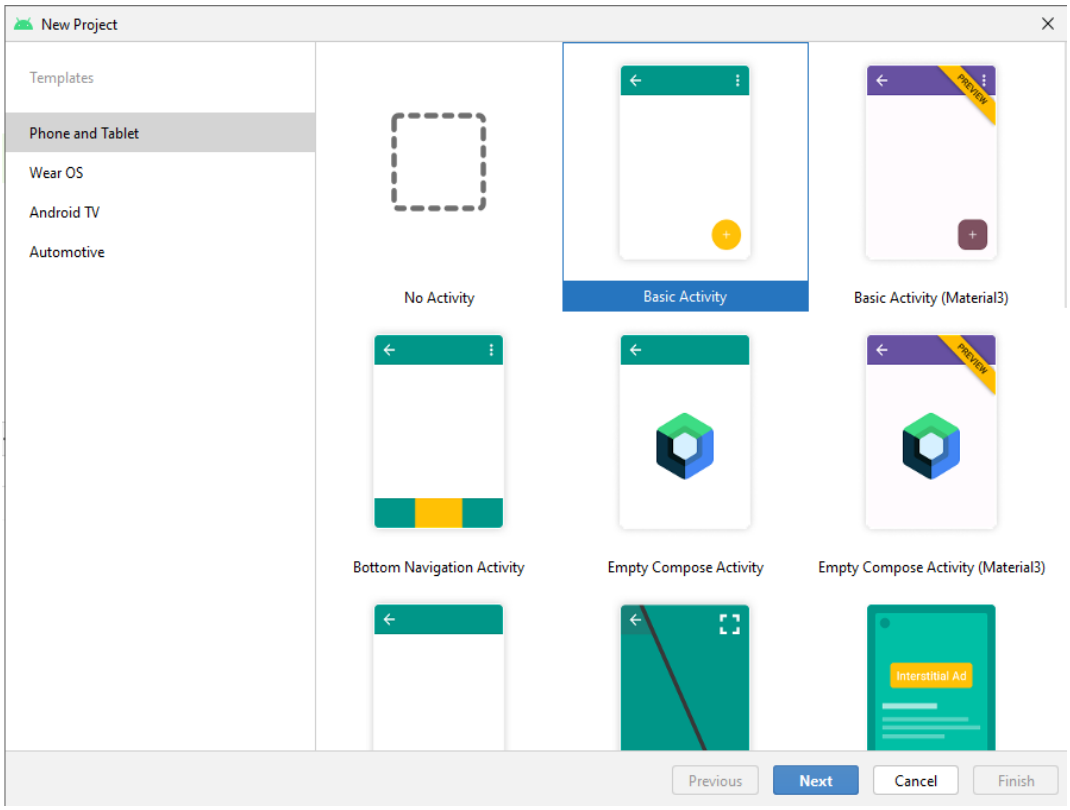


Figure 2.12 Screenshot of basic app selection options.

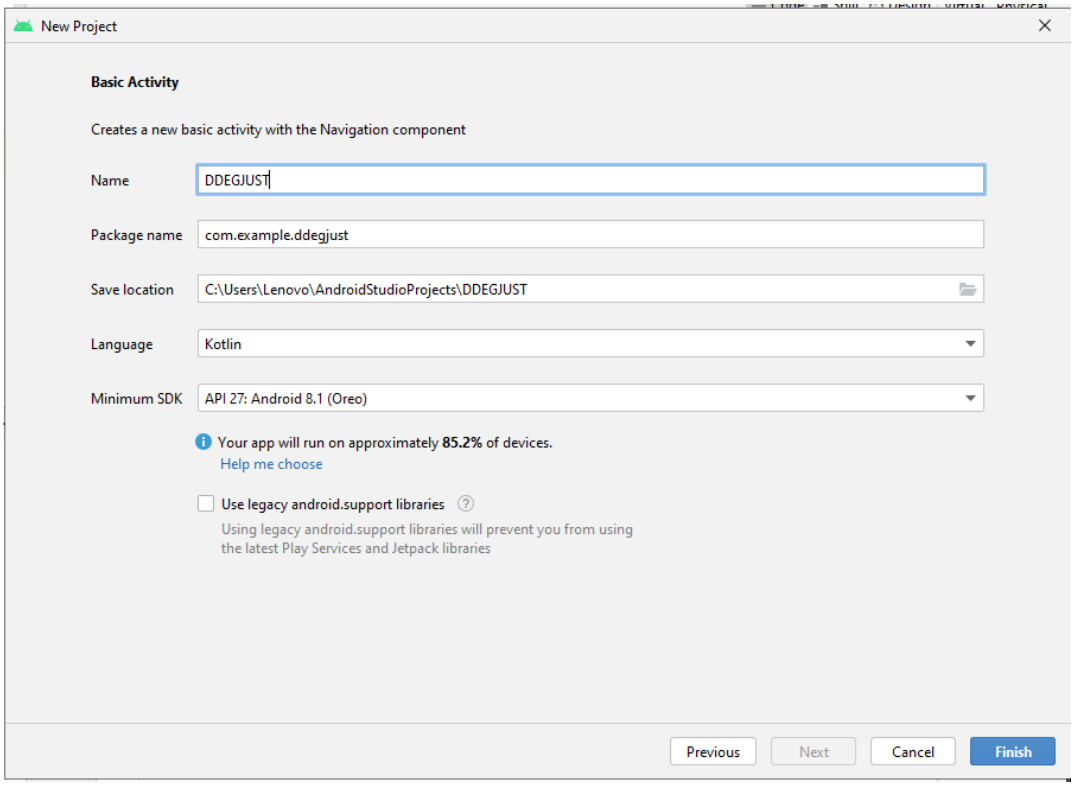


Figure 2.13 Option to choose project name, language and API version.

The code of fragment.xml and snippet of activity_main.xml files are

```
<?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="first.javatpoint.com.welcome.MainActivity">
   <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello Android Example"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
 </android.support.constraint.ConstraintLayout>
```

Figure 2.14 Code for fragment.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
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=".MainActivity">
    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/Theme.DDEGJUST.AppBarOverlay">
        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/Theme.DDEGJUST.PopupOverlay"/>
    </com.google.android.material.appbar.AppBarLayout>
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Figure 2.15 Code for activity_main.xml file.

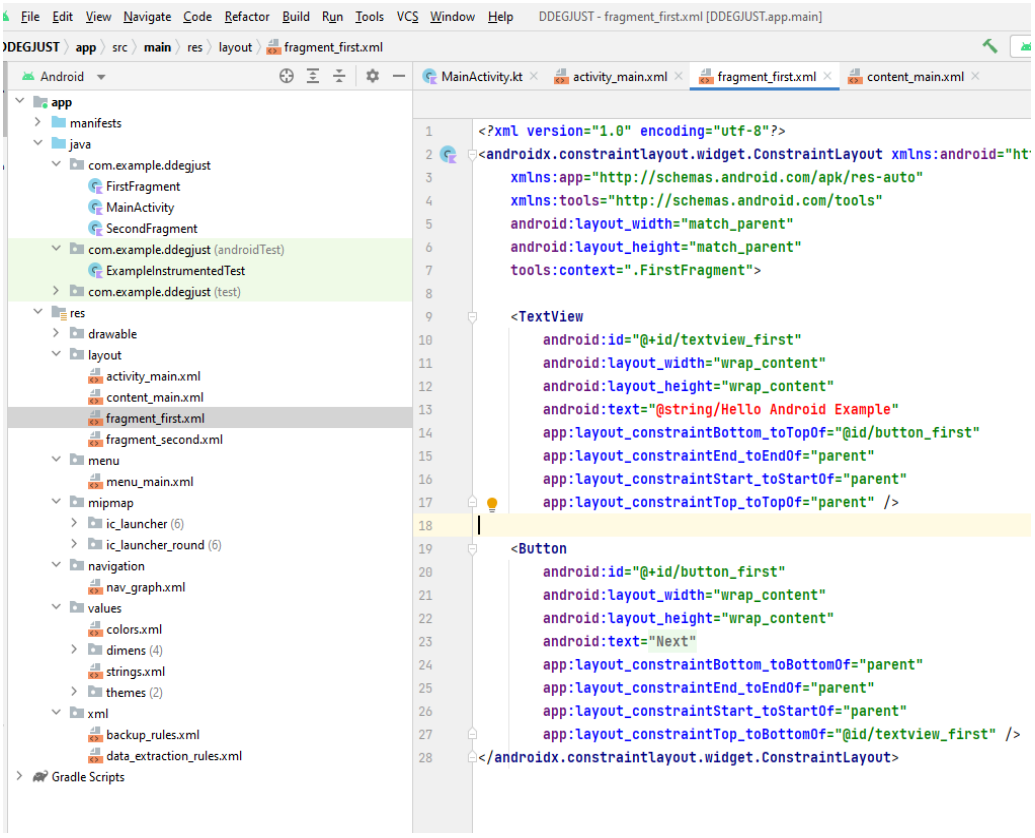


Figure 2.16 A look of fragment and other xml files screenshots.

Now click on the run button to view the application preview in the AVD.

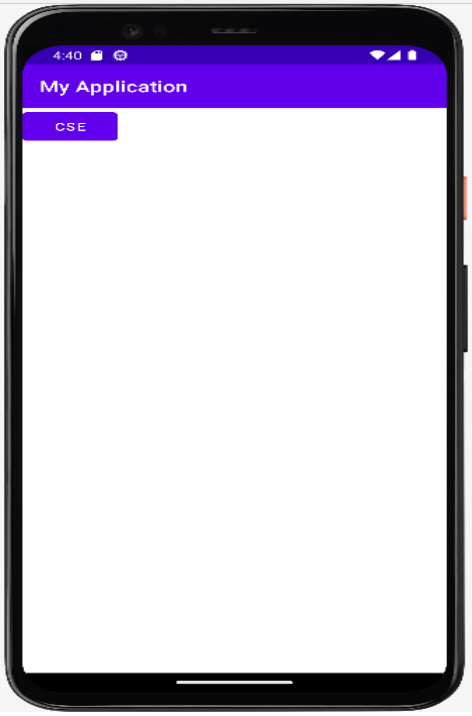


Figure 2.17 First view of designing App.

2.14. Summary

This chapter has covered the Dalvik machine framework components. Core libraries used to handle the application request. AndroidManifest.xml and R.java is explained in detail with their all attributes to understand the working process. The role of android software stack in accessing various groups classes related to file, security, kernel etc. It also elaborated the Android runtime environment for application execution, use of emulator device using AVD. Finally, this chapter end with a simple example of android application.

2.15. Keywords

Activity: A component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map.

Intent: An abstract description of an operation to be performed.

Service: A component that runs in the background to perform long-running operations or to perform work for remote processes.

Content Provider: A component that manages a shared set of application data.

Fragment: A portion of user interface in an Activity.

View: A widget in an activity's layout that is responsible for drawing and event handling.

Layout: An XML file that defines the architecture of an activity's user interface.

Manifest: An XML file that describes the fundamental characteristics of the app and defines each of its components.

Resources: Non-code files, such as strings, colors, and layouts, that are external to the source code and provide alternative resources for localization.

Gradle: A build system that automates the process of building, testing, and deploying Android applications.

SDK: A set of development tools that allow developers to create apps for a specific platform.

ADB: Android Debug Bridge, a command-line tool that allows developers to communicate with an emulator or connected Android device.

AVD: Android Virtual Device, an emulator configuration that allows developers to test apps in a virtual environment.

APK: Android Package Kit, the file format used by the Android operating system for distribution and installation of mobile apps.

Dalvik: A discontinued process virtual machine (VM) that was the primary platform for Android apps.

ART: Android Runtime, a cross-platform runtime environment used by Android to run apps written in Java and other languages.

Renderscript: A framework for running computationally intensive tasks at high performance on Android devices.

NDK: Native Development Kit, a set of tools that allows developers to use C and C++ code with Android apps.

Play Store: Google's official app store, where users can browse and download apps developed by third-party developers.

2.16. Check Your Progress

1. JVM stand for _____.
2. Compiler convert the .class file into _____.
3. Dex tool perform the operation to collect all _____ and convert into single ___ file.
4. AndroidManifest.xml contains information to connect _____.
5. API is an _____.
6. DVM is a _____.
7. DVM is written by _____.
8. The initial tag of androidmanifest.xml file is _____.
9. R.java is generated by _____.
10. The major layers of the Android Software Stack are: _____, _____,
11. _____, _____.
12. Fragment is a _____ of an activity area.
13. To simulate your application android studio uses _____ service.
14. When creating a new application, you can select language: _____.

2.17. Self-Assessment Questions

1. Write in brief about Delvik Virtual Machine.
2. Explain the working of android emulator.
3. What is interpreter? explain it uses in program.
4. Elaborates android building blocks.
5. What is the use of androidmanifest.xml file android App?
6. Discuss the resource file of App R.java.
7. Explain the android software stack classes with diagram.
8. Compare JVM and DVM.

9. List the name of security class used in android app.
10. What is android runtime environment?
11. Explain android virtual device functioning.

2.18. Answers (Section 2.15)

1. Java Virtual Machine.
2. Delvik Virtual Machine.
3. Dan Bornstein
4. .dex file
5. classes, .dex file.
6. broadcast services
7. Application Programming Interface
8. <application>
9. aapt
10. Application, Framework, runtime, Linux kernel
11. part
12. Android Emulator
13. Kotlin or Java

2.19. References/ Suggested reading

1. Headfirst Android Development, Dawn Griffiths, 1st edition, .O′Reilly
2. Android Programming for Beginners, John Horton,2nd edition, .Packt Publishing
3. Android Programming with Kotlin for Beginners, John Horton, 1st edition, Packt.
4. Head-First Kotlin, Dawn Griffiths, 1st edition, O'Reilly.
5. Android App Development, Michael Burton, 3rd edition.

SUBJECT: MOBILE APPLICATION DEVELOPEMENT	
COURSE CODE: MCA-42	**AUTHOR: Dr. Sunil Kumar Verma**
LESSON NO. 3	
Android UI Widgets	

STRUCTURE

LEARNING OBJECTIVE

This chapter will give and idea to the student to learn about basic widgets of the android programming. This will discuss about components like Button, Toast message, Image Button, Toggle Button, Switch Button, Check Box, Alert Dialog box, Spinner etc. After going through this chapter, you will be familiar with these widgets and got an idea, how to use these UI in your own App.

Chapter 3. Basic UI Widgets

3.1. Introduction

A widget is a UI controller of android application that is displayed on the main/home screen. Widgets are incredibly useful tools to let you easily access your favorite programs by placing them on your home screen. The likes of a toast widget, button widget, clock widget, etc. are likely familiar to you.

There are several different sorts of widgets, including image widgets, collection widgets, Checkbox, and TextView. We are given a comprehensive foundation by Android to create our own widgets.

3.2. Working with Button

React Native is used to export a <Button/> component that uses the native button element for Android or other devices. The <Button/> component uses title and onPress properties. However, it does not support a style prop, that built it hard to customize the style. We can make to styling a <Button/> inherited from React Native is with the color properties. The image below shows an example of two buttons for mobile devices. The first button is the default <Button /> and the second is another default <Button/> with its color prop set to "black".

React native base code is written for this example and the Visual Studio code is used to execute the code.

```
import React from 'react';
import {Text, View, StyleSheet, Pressable } from 'react-native';
export default function Button(props) {
 const { onPress, title = 'Click me' } = props;
 return (
  <Pressable style={styles.button} onPress={onPress}>
   <Text style={styles.text}>{title}</Text>
  </Pressable>
 );
}
const styles = StyleSheet.create({
 button: {
  alignItems: 'center',
  justifyContent: 'center',
  paddingVertical: 18,
  paddingHorizontal: 42,
  borderRadius: 4,
  elevation: 3,
  backgroundColor: 'green',
 },
 text: {
  fontSize: 16,
  lineHeight: 21,
  fontWeight: 'bold',
  letterSpacing: 0.25,
  color: 'black',
 },
});
```

Figure 3.1 Basic example of code written in App.js file for button.

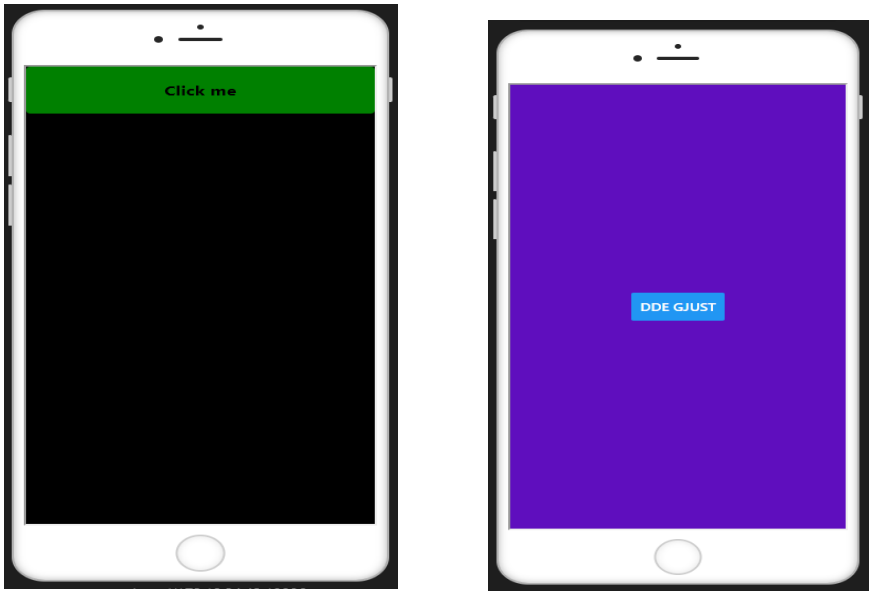


Figure 3.2 Screenshots of basic android application.

```
import React from 'react';
// Import the essential components from react-native
import {
   StyleSheet, Button, View, SafeAreaView,
   Text, Alert
} from 'react-native';
export default function App() {
   return (
      <View style={styles.container}>
         <Button
         // Some properties given to Button
          title="DDE GJUST"
          // onPress={() => Alert.alert('Deparment of Distance Education')}
        />
      </View>
   );
}
// Some styles given to button
const styles = StyleSheet.create({
   container: {
      flex: 1,
      backgroundColor: '#71EC',
      alignItems: 'center',
      justifyContent: 'center',
   },
});
```

Figure 3.3 Code for generating alert message.

3.3. Toast

A Toast in android is a feedback type message. It covers very small space for displaying meanwhile overall activities are interactive and visible to the current user. It automatically disappears after a few time spans in second. If a user tries to keep the message visible permanently then use notification.

Other category of Toast is custom Toast where images can be considered instead of a simple message for display. Toast class: This class provides a simple popup message that will be displayed on the Main Activity of the UI screen.

Toast class have two constant properties: LENGTH_LONG and LENGTH_SHORT which are public and final type with static scope. this property helps in display of message for a longer or short time. The methods of Toast class are:

public static Toast makeText (Context, CharSequence text, int duration): It consisted of text and time duration to display the toast message

public void show (): It is used to make visible a toast message

public void setMargin(float horizontalMargin, float verticalMargin) - It update the horizontal and vertical margin.

"This a simple toast message" is a Toast message which is displayed by clicking on 'CLICK' button. Every time when you click your toast message appears. You can alternatively use showWithGravity(message, duration, gravity) to specify where the toast appears in the screen's layout. May be ToastAndroid.TOP, ToastAndroid.BOTTOM or ToastAndroid.CENTER.

A toast showing a message on screen for a short interval. Some time it is also called popup notification about any activity we perform on app. For example: We write a code to add a edit text message and a button that will generate toast

First, we shall create a screen with an Edit Text (Text box to take input) and a Button.

Step 1: Create a new project in Studio with an empty activity

Step 2: Open file explorer option from left side of android app then open activity_main.xml.

Step 3: Add a button and EditText. and remove exiting xml tag lines for "hello word".

Now XML file should contain you types message.

Step 4: Save the XML file

Step 5: Open the MainActivity.kt file to add the folowing methods shown in figure

The object of EditText is messageEditText that we added in XML layout file in Step 3. A variable is used to get the data from EditText using the text property. Setting a Toast using the code: var toast = Toast.makeText(this, message, Toast.LENGTH_LONG).

Toast.makeText() creates a toast and it accepts three parameters:

context – The context is asking for a class for which you want to use. Usually it may be "android.app.Activity" or "android.app.Application" object.

text – The text to show. Can be formatted text.

duration –A time spans for displaying a message. it can be for #LENGTH_SHORT or #LENGTH_LONG

A variable toast is used to store the message and to make it visible call toast.show().

Step 7: We need to add on click event listener to button for toastMessage() method. Go back to the activity_main.xml layout file and add this attribute to the Button.

activity_main.xml | MainActivity.kt

```
6       android:layout_width="match_parent"
7       android:layout_height="match_parent"
8       tools:context=".MainActivity">
9       <Button
10          android:id="@+id/toastButton"
11          android:layout_width="0dp"
12          android:layout_height="wrap_content"
13          android:text="Toast"
14          android:onClick= "toastMessage"
15          app:layout_constraintEnd_toEndOf="parent"
16          app:layout_constraintHorizontal_bias="0.5"
17          app:layout_constraintTop_toTopOf="parent"
18          app:layout_constraintStart_toEndOf="@+id/messageEditText"
19          />
20      <EditText
21          android:id="@+id/messageEditText"
22          android:layout_width="0dp"
23          android:layout_height="wrap_content"
24          android:ems="10"
25          android:hint="Write a toast message"
26          app:layout_constraintHorizontal_bias="0.5"
27          app:layout_constraintStart_toStartOf="parent"
28          app:layout_constraintTop_toTopOf="parent"
29          app:layout_constraintEnd_toStartOf="@+id/toastButton"
30          />
31  </androidx.constraintlayout.widget.ConstraintLayout>
```

Figure 3.4 Code for activity_main.xml file of the Toast.

Step 8: To hide the Keyboard after clicking on toast button we write the hide keyboard message

Now finally run this App. Enter a message inside textbox and click on the button. It will toast the message.

y_main.xml × MainActivity.kt × AndroidManifest.xml ×

```
import android.widget.EditText
import android.widget.Toast

class MainActivity : AppCompatActivity() {

    fun hideKeyboard(activity: Activity)
    {
        val view = activity.findViewById<View>(android.R.id.content)
        if (view != null)
        {
            val im = activity.getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
            im.hideSoftInputFromWindow(view.windowToken,  flags: 0)
        }
    }
    fun toastMessage(view: View)
    {
        val messageEditText = findViewById<EditText>(R.id.messageEditText)
        val message = messageEditText.text.toString()
        var toast = Toast.makeText( context: this, message, Toast.LENGTH_LONG)
        hideKeyboard( activity: this)
        toast.show()
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

    }
}
```

Figure 3.5 Code for MainActivity.kt file of the Toast.

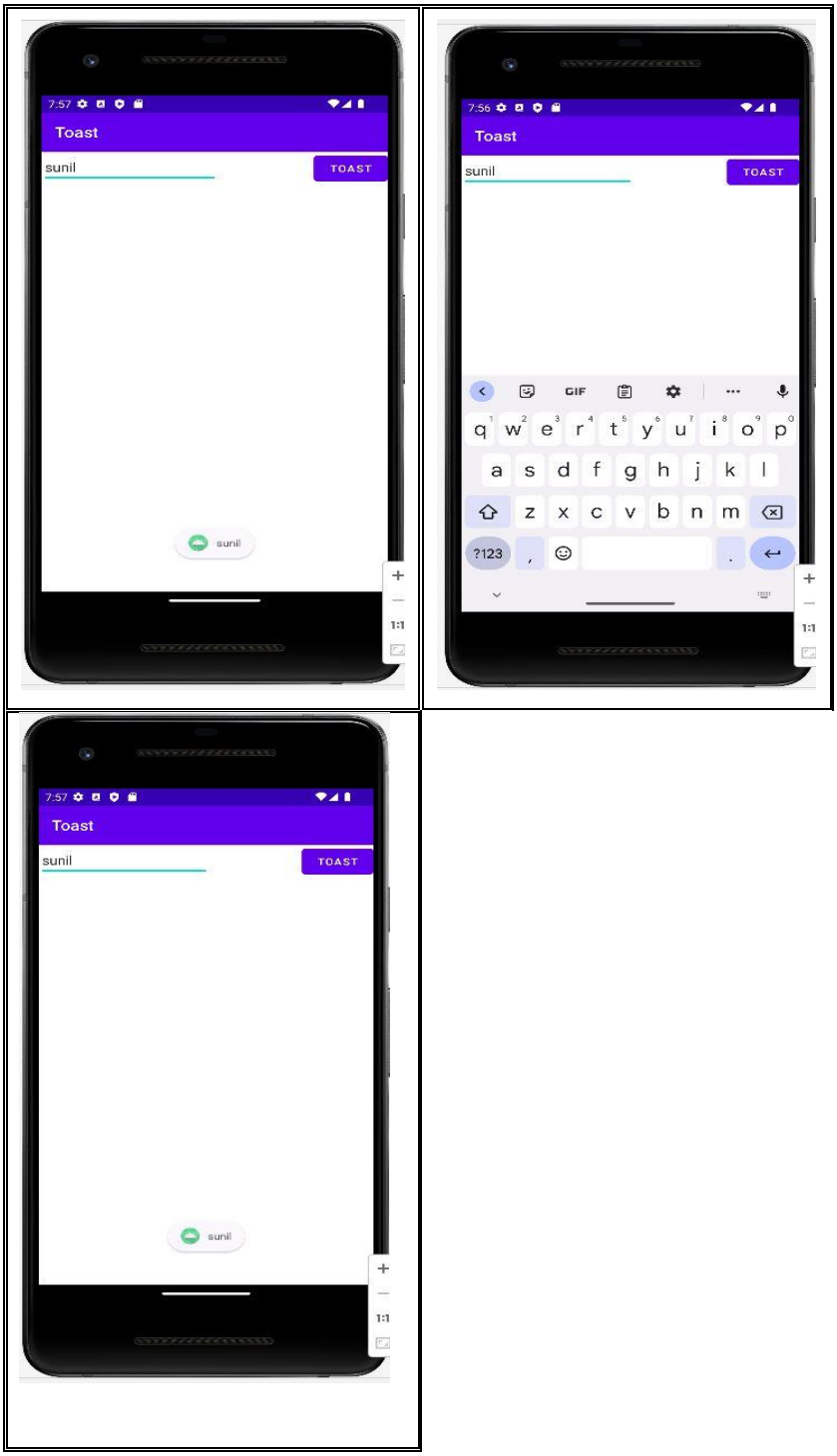


Figure 3.6 Result of the Toast App.

3.4. Button

Button is one of the basic UI widgets to perform some action. Button is a push type component which has a feature to clicked or press on it to perform some action. Android framework have number of buttons like ToggleButton, RadioButton, Simple button and CompoundButton. Button is a derived class of TextView. But the different types of buttons are inherited from Button class. The various types of action can be recognized through button like click, press and touch event. button is considered most popular and click sensible component of android.

Button can be designed in numerous ways:

- Button with text.
- Button with image.
- Combined button (image + text)

Code for button in XML. Using button just write “DDE GJUST”

```
<Button
android:id="@+id/simpleBtn"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="DDEGJUST"/>
```

This section will discuss button-based alert message on mobile screen. This alert screen is a small size window to take click-based input from user. Sometime application has some important properties where before applying some operation, it is the need to alert user about operation and ask again to perform operation really. So, there is a provision for this to generate a dialog box. This dialog box will ask query with few button options.

AlertDialog.Builder class is used to design an interface for alert dialog box. Interface of the alert box include message, button, image, and onClick functions etc.

Attributes of Button

gravity: It is used to control the text alignment like top, bottom, center, left, right, center_horizontal, center_vertical etc.

id: It is used to uniquely recognize button throughout the project.

<Button

android:id="@+id/button"

text: It is used to set text value for button. Text values can be written through xml or java file also.

android:text="exit" //directly assigned

android:text="@string/exit" //from string.xml file

textColor: It is sued to set the color for text value of a Button. Color format is "#rgb", "argb" "#rrggbb", or "#aarrggbb".

textSize: To adjust the text size on Button. unit of size can be in sp stands for scale independent pixel and dp stands for density pixel.

padding: it is a space from all side as from top, bottom, left, and right.

drawableBottom: It is used to add an image in button using drawableBottom property which display by default below the button text. There is various option to display an image like drawableLeft, drawableRight, and drawableTop.

Similarly, there are lots of feature for shaping the button. You need to just write keyword:

"android:" then all properties will display, choose that you want to set.

```
<resources>
    <string
name="app_name">Button</string>
    <string
name="exit">exit</string>
    <string name="button">DDE
GJUST</string>
</resources>
```

Figure 3.7 Code for string.xml file of the Button.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#18EE00</color>
    <color name="purple_700">#B8A2E8</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
</resources>
```

Figure 3.8 Code for color.xml file of the Button.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/button"
        style="@style/Widget.AppCompat.Button.Colored"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:onClick="exit"
        android:text="@string/exit"
        android:textStyle="normal|bold"/>
    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_above="@+id/button"
        android:layout_centerHorizontal="true"
        android:gravity="center"
        android:text="@string/button"
        android:textSize="28sp"
        android:textStyle="normal|bold"
        tools:ignore="MissingConstraints"
        tools:layout_editor_absoluteX="0dp"
        tools:layout_editor_absoluteY="16dp" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Figure 3.9 Code for activity_main.xml file of Button.

```
package gjust.dde.button;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
    @Override   protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);     }
    public void exit(View view){
AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
        alertDialogBuilder.setTitle("Confirm Exit !");
        alertDialogBuilder.setIcon(R.drawable.python);
        alertDialogBuilder.setMessage("Are you sure,You want to exit");
        alertDialogBuilder.setCancelable(false);
alertDialogBuilder.setPositiveButton("Yes", (arg0, arg1) -> finish());
alertDialogBuilder.setNegativeButton("No", (dialog, which) ->
Toast.makeText(MainActivity.this,"You clicked
No",Toast.LENGTH_SHORT).show());
alertDialogBuilder.setNeutralButton("Cancel", (dialog, which) ->
Toast.makeText(getApplicationContext(),"You clicked
Cancel",Toast.LENGTH_SHORT).show());
AlertDialog alertDialog = alertDialogBuilder.create();
        alertDialog.show();
    }}
```

Figure 3.10 Code for MainActivity.java fie of the Button.

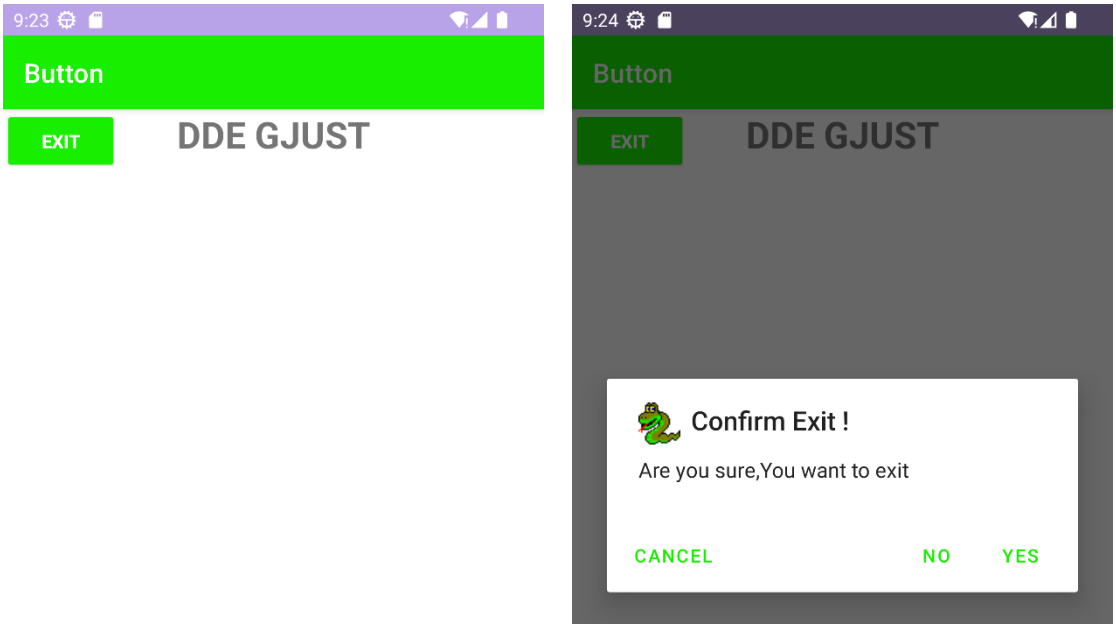


Figure 3.11 Screenshots of the Button App with Alert Dialog Box.

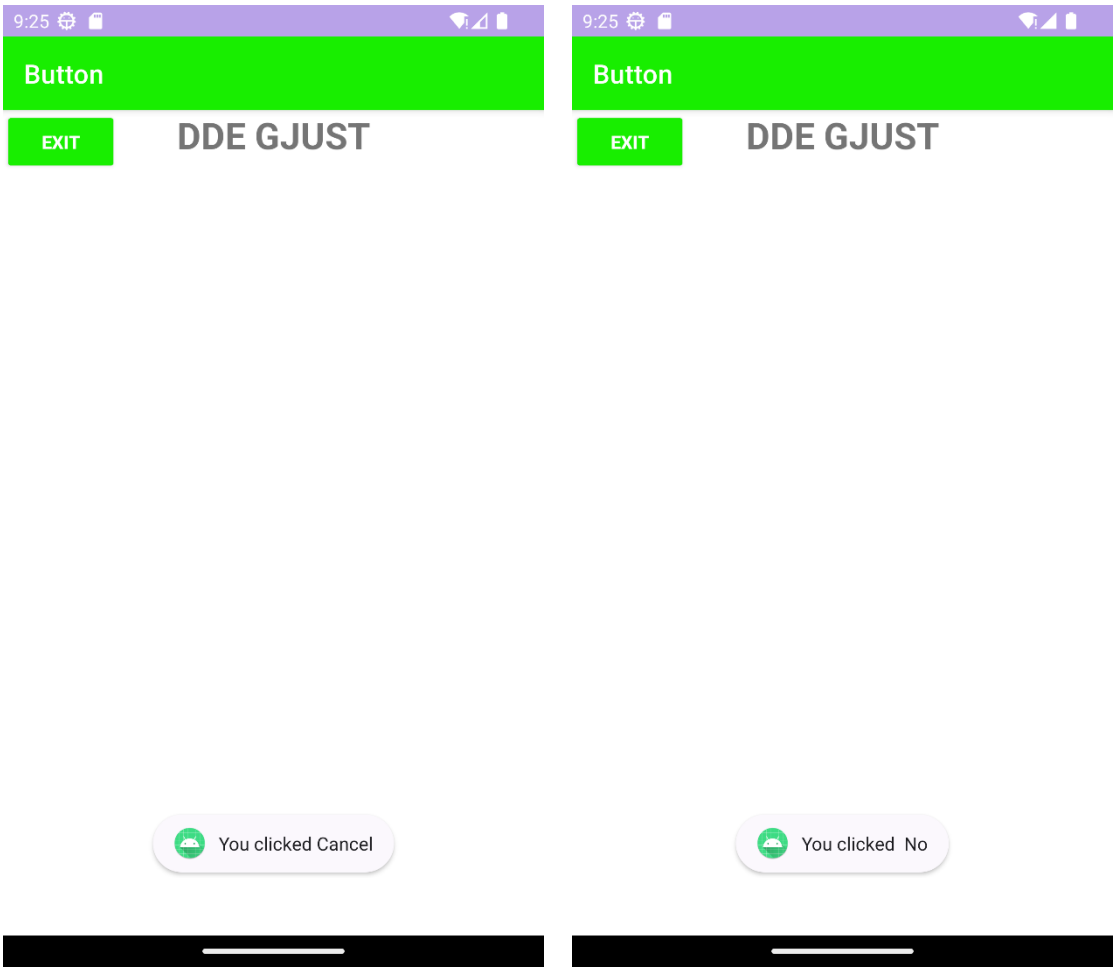


Figure 3.12 Screenshots of the Button App with Toast message.

3.5. Toggle Button

Toggle Buttons in android is just containing two state that is used to display checked/unchecked. It is more useful when user want to change the setting among two states. For example, we use on/off Wi-Fi, Hotspot, Bluetooth, Torch, or alignment of text in MS word etc. New android version also supports another type of toggle button that provides slider control.

The class ToggleButton and Switch for slider control are both works under the CompoundButton class. earlier SwitchCompat class used for switch button. We can add simple toggle

button in app layout by using the ToggleButton object. SwitchCompat is a version of the Switch widget which runs on devices back to API 7.

To make any change in state of button you can try the CompoundButton.toggle() or CompoundButton.setChecked() method. To detect the activity of user on the button or switch we can create a CompoundButton.OnCheckedChangeListener object and set it to the button using setOnCheckedChangeListener().

For example:

```
val toggleb: ToggleButton = findViewById(R.id.togglebutton)
toggleb.setOnCheckedChangeListener
{ _, isChecked ->
        if (isChecked)
            { // The toggle button is enabled }
        else
            { // The toggle button is disabled }
}
```

ToggleButton class provides the property of creating the toggle button. This class have three XML attributes explain below with details

android:textOff - The button text when it is not checked.

android:textOn - The button text when it is checked.

android:disabledAlpha - The indicator when button is disabled.

ToggleButton class have various methods. Here we include only widely used methods

Method with details

CharSequence getTextOff()-It returns the text when button state is not checked.

CharSequence getTextOn()-It returns the text when button state is checked.

void setChecked(boolean checked)- It changes the button state.

```
package com.example.togglebutton

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Toast
import android.widget.ToggleButton

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val toggle: ToggleButton = findViewById(R.id.toggleButton)
        toggle.setOnCheckedChangeListener { _, isChecked ->
            Toast.makeText( context: this, if(isChecked)
                "Wifi ON" else "Wifi OFF", Toast.LENGTH_SHORT).show()
        }
    }
}
```

Figure 3.13 Toggle Button code for MainActivity.java file.

```
<?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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ToggleButton
        android:id="@+id/toggleButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Figure 3.14 Code for activity_main.xml file of the Toggle Button.

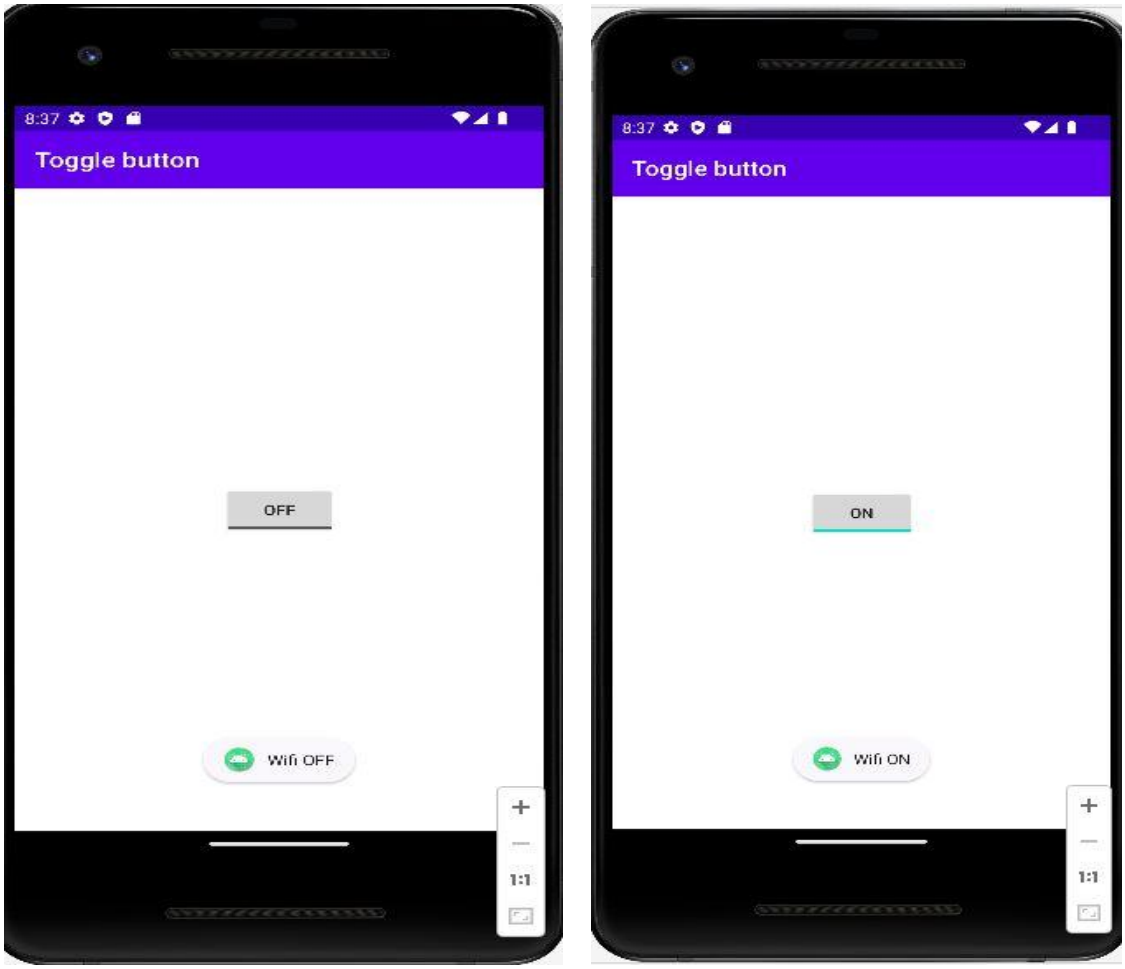


Figure 3.15 Screenshots of Toggle button.

3.6. Switch Button

In android, Switch is a two-state user interface element that 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.

The Switch element is useful for the users to change the settings between two states either ON or OFF. We can add a Switch to our application layout by using Switch object.

Switch button is shown below.

By default, the android Switch will be in the OFF (Unchecked) state. We can change the default state of Switch by using android:checked attribute.

In case, if we want to change the state of Switch to ON (Checked), then we need to set android:checked = “true” in our XML layout file. In android, we can create Switch control in two ways either in the XML layout file or create it in the Activity file programmatically.

activity_main.xml | MainActivity.kt

Code | Spli

```
3   android:layout_width="match_parent" android:layout_height="match_parent
4   <Switch
5       android:id="@+id/switch1"
6       android:layout_width="wrap_content"
7       android:layout_height="wrap_content"
8       android:switchMinWidth="56dp"
9       android:layout_marginLeft="100dp"
10      android:layout_marginTop="120dp"
11      android:text="Switch-1:"
12      android:checked="true"
13      android:textOff="OFF"
14      android:textOn="ON"/>
15  <Switch
16      android:id="@+id/switch2"
17      android:layout_width="wrap_content"
18      android:layout_height="wrap_content"
19      android:switchMinWidth="56dp"
20      android:layout_below="@+id/switch1"
21      android:layout_alignLeft="@+id/switch1"
22      android:text="Switch-2:"
23      android:textOff="OFF"
24      android:textOn ="ON"/>
25  <Button
26      android:id="@+id/getBtn"
27      android:layout_width="wrap_content"
28      android:layout_height="wrap_content"
29      android:layout_marginLeft="150dp"
30      android:layout_marginTop="200dp"
31      android:text="Check" />
32  </RelativeLayout>
```

Figure 3.16 Switch Button activity_main.xml file.

The above code snippet uses a switch button setting. In which the checked attribute of switch is set to true and property of textOff and textOn are set "OFF" and "ON". Respectively, the same thing we can implement in Kotlin or java program inside activity file.

```
Switch s = (Switch)findViewById(R.id.switch2);
s.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
  public void onCheckedChanged(CompoundButton buttonView,  boolean isChecked)
{
      if (isChecked) {   // The switch is on }
else { // The switch is off  }
   }
});
```

Now the final code for switch button is shown in below diagram.

activity_main.xml | MainActivity.kt

```
package com.example.switchbutton
import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.Switch
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
class MainActivity : AppCompatActivity() {
    private var sw1: Switch? = null
    private var sw2: Switch? = null
    private var btnGet: Button? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        sw1 = findViewById<View>(R.id.switch1) as Switch
        sw2 = findViewById<View>(R.id.switch2) as Switch
        btnGet = findViewById<View>(R.id.getBtn) as Button
        btnGet!!.setOnClickListener {  it: View!
            val str1: String
            val str2: String
            str1 = if (sw1!!.isChecked) sw1!!.textOn.toString()
                    else sw1!!.textOff.toString()
            str2 = if (sw2!!.isChecked) sw2!!.textOn.toString()
                    else sw2!!.textOff.toString()
            Toast.makeText(
                applicationContext,
                text: "Switch Button 1 -  $str1 \nSwitch Button 2 - $str2",
                Toast.LENGTH_SHORT
            ).show()
        }
    }
}
```

Figure 3.17 Switch button code for MainActivity.kt file.

Finally, we may run this program in android studio and see the result on AVD emulator.

The detailed description of each attribute of switch given below:

android:id :To uniquely identify the control.

android:checked :To check the current state of switch control.

android:gravity :It tells the alignment of the text like left, right, center, top, etc.

android:text :To set the text for element.

android:textOn :To set the element text when the toggle button is in the ON/Checked state.

android:textOff :To set the element text when toggle button is in OFF / Unchecked state.

android:textColor :Setting the color of the text.

android:textSize :The size of the text.

android:textStyle :The font style of text.

android:background:Setting of the background color for button control.

3.7. Image Button

Image button is an element to display an interface for button as an image. It performs an action when user click on the this or use tab.

It helps to display a button with image.it can be clicked or pressed by the user. Image can be described by the android:src attribute of the ImageButton tag in XML file or in code section use setImageResource() method. If user want to set their own image, then he may set the button background color as transparent.

To indicate the different state of button like focused or pressed user can set different image for both states.

In the above code section, we have set the ImageButton control showing the image placed in drawable folder by using android:src attribute in xml file available in Layout section of application. Other alternate is to write code in Kotlin or java for setting image source like.

LinearLayout lout = (LinearLayout)findViewById(R.id.l_layout);

ImageButton btn = new ImageButton(this);

btn.setImageResource(R.drawable.add_icon);

lout.addView(btn);

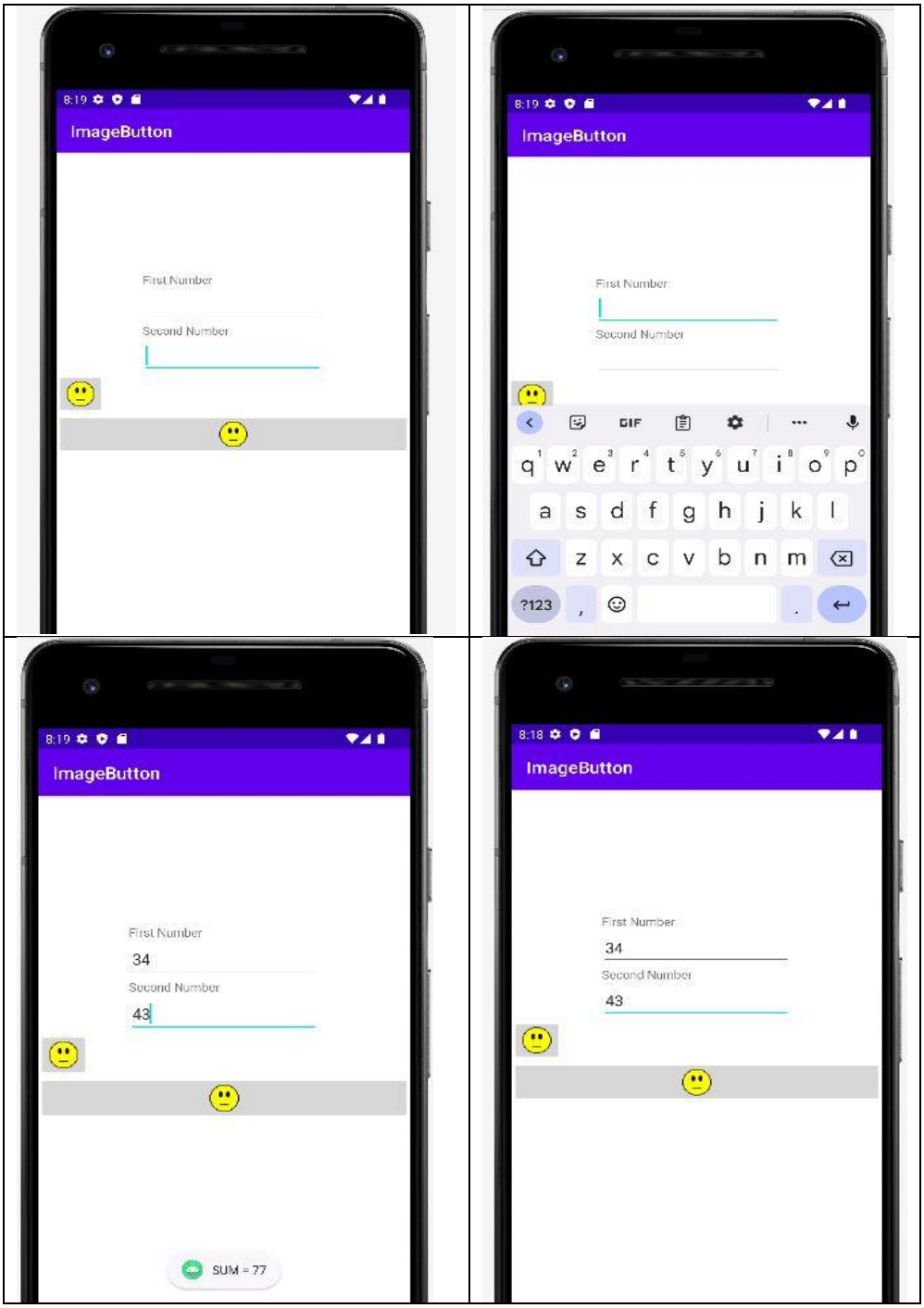
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent" android:id="@+id/l_layout">>
    <TextView
        android:id="@+id/fstTxt"   android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:layout_marginTop="150dp"
        android:text="First Number"/>
    <EditText
        android:id="@+id/firstNum"     android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:ems="10"/>
    <TextView
        android:id="@+id/secTxt"      android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Second Number"
        android:layout_marginLeft="100dp" />
    <EditText
        android:id="@+id/secondNum"      android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:ems="10" />
    <ImageButton
        android:id="@+id/addBtn"     android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="addOperation"
        android:src="@drawable/add_icon" | android:text="Add"/>
</LinearLayout>
```

Figure 3.18 Image Button code for activity_main.xml file.

```
import android.widget.EditText
import android.widget.ImageButton
import android.widget.LinearLayout
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
class MainActivity : AppCompatActivity() {
    fun hideKeyboard(activity: Activity){
     val view = activity.findViewById<View>(android.R.id.content)
     if (view != null)
     { val im = activity.getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
     im.hideSoftInputFromWindow(view.windowToken, flags: 0)
     } }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val firstNum = findViewById<View>(R.id.firstNum) as EditText
        val secNum = findViewById<View>(R.id.secondNum) as EditText
        val btnAdd = findViewById<View>(R.id.addBtn) as ImageButton
        val layout = findViewById<View>(R.id.l_layout) as LinearLayout
        val btn = ImageButton( context: this)
        btn.setImageResource(R.drawable.add_icon)
        layout.addView(btn)
        btnAdd.setOnClickListener { it: View!
            hideKeyboard( activity: this)
            if (firstNum.text.toString().isEmpty() || secNum.text.toString().isEmpty()) {
            Toast.makeText(applicationContext, text: "Please fill all the fields",
                Toast.LENGTH_SHORT).show()
             }
            else {val num1 = firstNum.text.toString().toInt()
                 val num2 = secNum.text.toString().toInt()
                 Toast.makeText(applicationContext, text: "SUM = "
                        + (num1 + num2), Toast.LENGTH_SHORT).show()
  } } }}
```

Figure 3.19 Image button code for MainActivity.kt file.

The click event handler can be used for button by adding android:onClick attribute into the <ImageButton> element.



.

Figure 3.20 Result of the Image Button App

The value of this attribute must be the exact name of method that we need to call on click event response and the Activity file must implement the corresponding method of XML layout.

3.8. Check Box

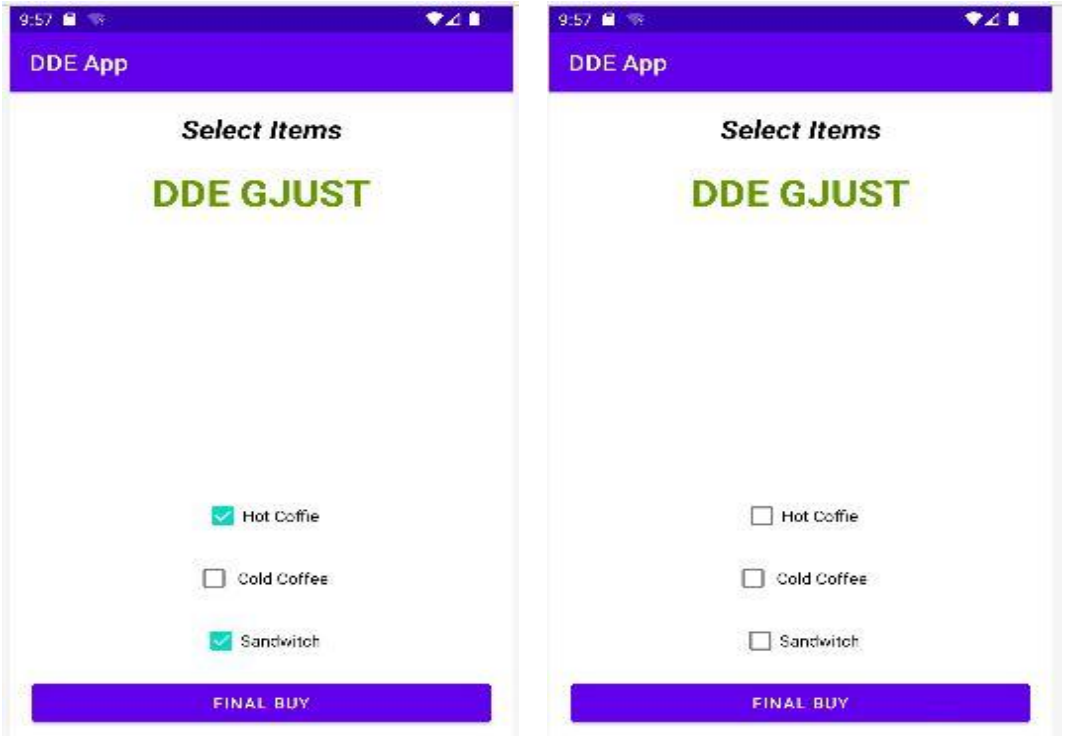
A Check Box is a special case of button where it has two states (checked or unchecked). It is a common element available on widget. It is a good example is “Remember me” option when login in any app. There are some other examples like to choose from a list of items or it may be mutually exclusive. There is also option for select more than one option. In survey application checkbox is used to select multiple options as answer. Default property of checkbox can be set by user using android:checked as checked or unchecked

```
package com.example.checkbox
import android.os.Bundle
import android.widget.Button
import android.widget.CheckBox
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import java.lang.StringBuilder
class MainActivity : AppCompatActivity() {
    lateinit var HotCoffee: CheckBox
    lateinit var ColdCoffee: CheckBox
    lateinit var Sandwitch: CheckBox
    lateinit var button: Button
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        title = "DDE App"
        HotCoffee = findViewById(R.id.checkBox1)
        ColdCoffee = findViewById(R.id.checkBox2)
        Sandwitch = findViewById(R.id.checkBox3)
        button = findViewById(R.id.button)
        button.setOnClickListener {
            var totalAmount: Int = 0
            val result = StringBuilder()
            result.append("Selected Items")
            if (HotCoffee.isChecked) {
                result.append("\n Hot Coffee 120Rs")
                totalAmount += 100
            }
            if (ColdCoffee.isChecked) {
                result.append("\n Cold Coffee 100Rs")
                totalAmount += 50
            }
            if (Sandwitch.isChecked) {
                result.append("\n Sandwitch 150Rs")
                totalAmount += 150
            }
            result.append("\nTotal: " + totalAmount + "Rs")
            Toast.makeText(applicationContext, result.toString(), Toast.LENGTH_SHORT).show()
        }
    }
}
```

Figure 3.21 Checkbox code for MainActivity.java file.

```
<?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"
   android:padding="18dp"
   tools:context="MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="52dp"
        android:text="DDE GJUST"
        android:textAlignment="center"
        android:textColor="@android:color/holo_green_dark"
        android:textSize="34sp"
        android:textStyle="bold" />
     <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="30dp"
        android:text="Select Items"
        android:textAlignment="center"
        android:textColor="@android:color/background_dark"
        android:textSize="24sp"
        android:textStyle="bold|italic" />
     <CheckBox
        android:id="@+id/checkBox1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="Hot Coffie" />
     <CheckBox
        android:id="@+id/checkBox2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/checkBox1"
        android:layout_centerInParent="true"
        android:layout_marginTop="10dp"
        android:text=" Cold Coffee" />
     <CheckBox
        android:id="@+id/checkBox3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/checkBox2"
        android:layout_centerInParent="true"
        android:layout_marginTop="10dp"
        android:text="Sandwitch" />
     <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/checkBox3"
        android:layout_marginTop="10dp"
        android:text="Final Buy" />
</RelativeLayout>
```

Figure 3.22 Check Box code for activity_main.xml. file.



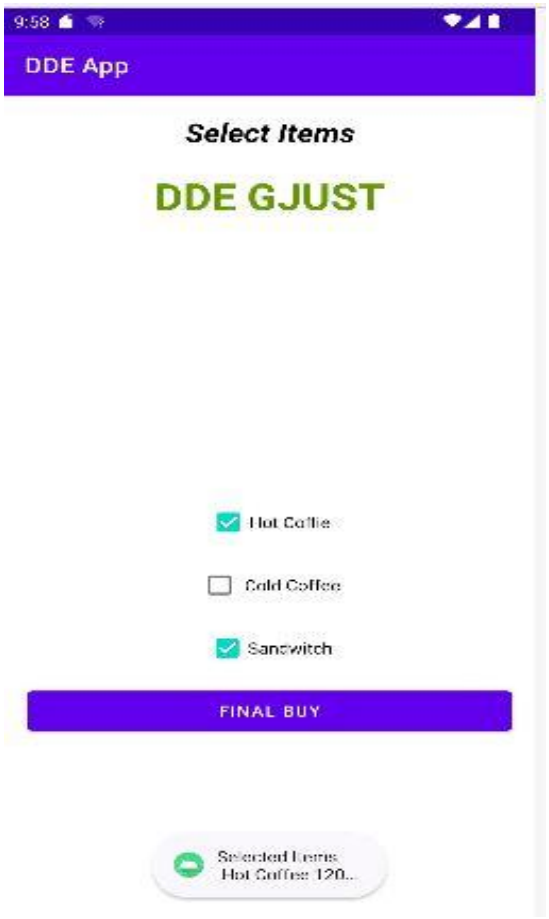


Figure 3.23 Result of Checkbox App.

3.9. Alert Dialog

Alert Dialog is a dialog window that prompt on the android screen. They usually show some information and demand for user action. There are three main components that helps in building an Alert Dialog.

- Title Text
- Message Text
- Buttons- Button have three types of variants: Positive, Negative, and Neutral

To create a new Alert Dialog, use a AlertDialog.Builder inner class.

val alertDialogB = AlertDialog.Builder(this)

We pass context as a parameter inside the constructor. There are few methods used on an Alert Dialog.

setTitle, setMessage, setIcon, setCustomTitle for a custom view. The setPositiveButton also a string name or Button that used to activate callback method. setView is used to add a custom view inside the alert dialog box.

- setList is used to bind an array of strings which may be displayed in the list form.
- setMultiChoiceList is also an array but here we can choose multiple items from the list of checkBox.
- setPositiveButtonIcon is an icon set alongside the Button
- show() is used to display the AlertDialog box
- setDismissListener is help to call a trigger when alert dialog is dismissed.
- setShowListener is used to set the logic to be triggered on dismiss of alert dialog. There is some more function which can be implemented as per requirement.

Android AlertDialog is an inherited from Dialog class. A dialog consists of the title, message, maximum three buttons.

An object of AlertDialog.Builder class is used to make an alert dialog with positive (yes), negative (no) and neutral (cancel) decision.

The example showing prompt an alert message on clicking a button. This dialog window may set three decision actions as positive, negative and neutral.

```
<resources>
  <string name="app_name">AlertDialog</string>
  <string name="button">click button</string>
  <string name="dialogTitle">Delete File</string>
  <string name="dialogMessage">Deleting file may be harm your system</string>
</resources>
```

Figure 3.24 Alert Dialog code for string.xml file.

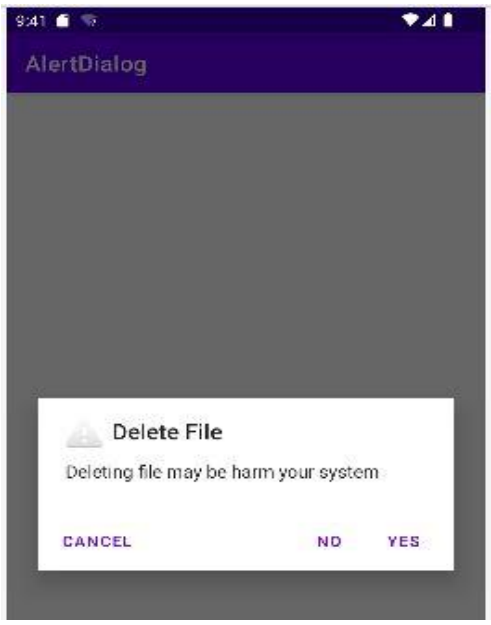
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
        <Button
         android:id="@+id/button"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_marginBottom="10dp"
         android:layout_marginEnd="10dp"
         android:layout_marginStart="10dp"
         android:layout_marginTop="10dp"
         android:text="@string/button"
         app:layout_constraintBottom_toBottomOf="parent"
         app:layout_constraintEnd_toEndOf="parent"
         app:layout_constraintStart_toStartOf="parent"
         app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Figure 3.25 AlertDialog box code for activity_main.xml.

```
package com.example.alertdialog
import android.os.Bundle
import android.widget.Button
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val button = findViewById<Button>(R.id.button)
        button.setOnClickListener {
            val builder = AlertDialog.Builder(this)
            builder.setTitle(R.string.dialogTitle)
            //set message for alert dialog
            builder.setMessage(R.string.dialogMessage)
            builder.setIcon(android.R.drawable.ic_dialog_alert)
            builder.setPositiveButton("Yes") { dialogInterface, which ->
                Toast.makeText(applicationContext, "clicked yes",
Toast.LENGTH_LONG).show()
            }
            builder.setNeutralButton("Cancel") { dialogInterface, which ->
Toast.makeText( applicationContext, "clicked cancel\n
operation cancel", Toast.LENGTH_LONG).show()
            }
            //performing negative action
            builder.setNegativeButton("No") { dialogInterface, which ->
                Toast.makeText(applicationContext, "clicked No",
Toast.LENGTH_LONG).show()
            }
            // Create the AlertDialog
            val alertDialog: AlertDialog = builder.create()
            // Set other dialog properties
            alertDialog.setCancelable(false)
            alertDialog.show()
        }
    }
}
```

Figure 3.26 AlertDialog Bix code for MainActivity.java file.


9:39
AlertDialog
CLICK BUTTON


9:41
AlertDialog
Delete File
Deleting file may be harm your system
CANCEL
NO
YES

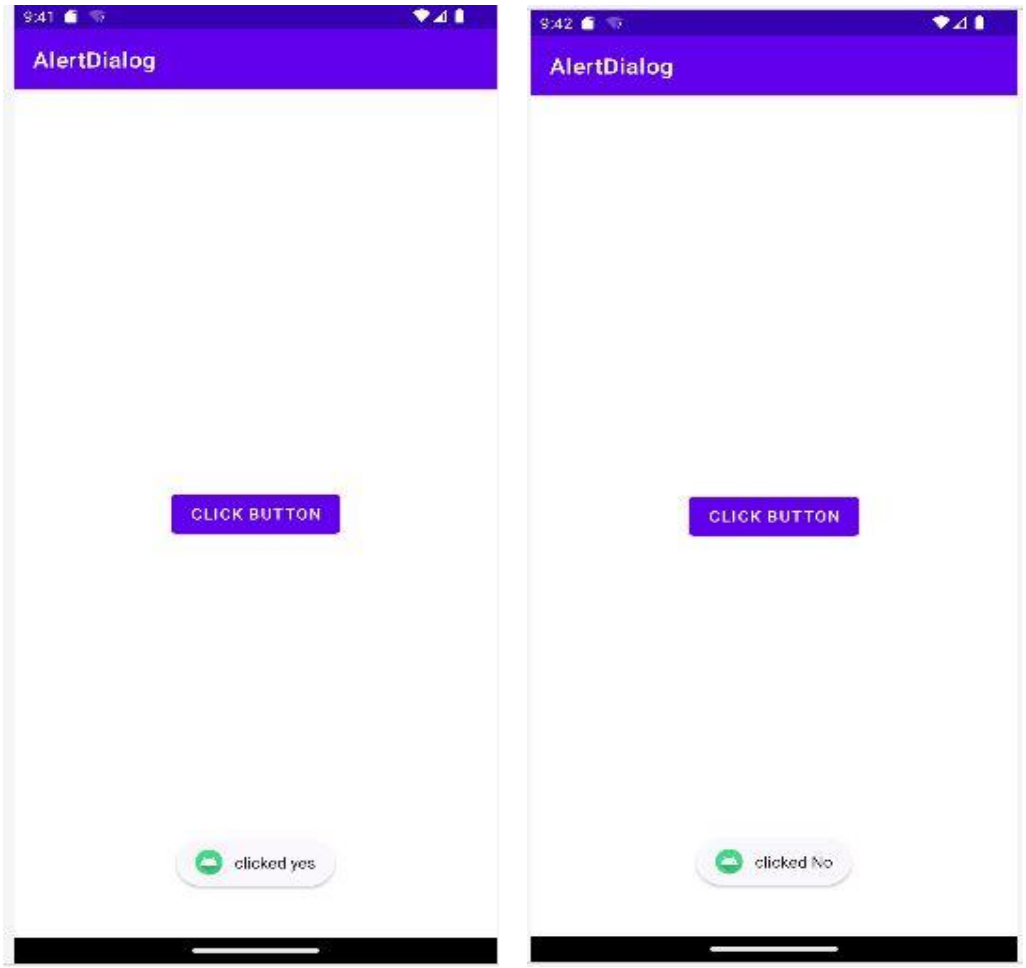


Figure 3.27 Screenshots of AlertDialog App.

3.10. Spinner

Android Spinner is working like a dropdown list used to select one option at a time from the list of options. It shows a complete dropdown list which shows all values when click on it. We can set default value as currently selected value. Adapter is used to bind the items to the spinner objects. We can populate spinner with values using an ArrayAdapter in Kotlin/java file.

Android Spinner is a view of displaying one child at a time on user clicks. It allows lets the user choose among multiple values. Let learn how to create a Spinner in layout file and set listener for the Spinner to serve user actions like clicking, selecting a value from Spinner.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:gravity="center"
    android:orientation="vertical"
    tools:context="com.example.spinner.MainActivity">
    <Spinner
        android:id="@+id/spinner_sample"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
    <TextView
        android:id="@+id/msg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="25dp"
        android:padding="20dp"
        android:textColor="@color/purple_500"/>
</LinearLayout>
```

Figure 3.28 Spinner code for activity_main.xml file.

```
package com.example.spinner
import androidx.appcompat.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.TextView
import com.example.spinner.R.id.*;
class MainActivity : AppCompatActivity(),AdapterView.OnItemSelectedListener {
   var Fruits =arrayOf("Banana", "Apple", "Mango", "Guava", "Pineapple",
"Grapes", "Mausmi", "Kinnu", "Papaya", "Other")
   var spinner:Spinner? =null
   var textView_msg:TextView? =null
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      textView_msg =this.findViewById(msg)
      spinner =this.findViewById(spinner_sample)
      spinner!!.setOnItemSelectedListener(this)
     val aa = ArrayAdapter(this, android.R.layout.simple_spinner_item, Fruits)

aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
      spinner!!.setAdapter(aa)
   }
   override fun onItemSelected(arg0: AdapterView<*>, arg1: View, position: Int, id:
Long) {
      textView_msg!!.text ="Selected: "+Fruits[position]
   }
   override fun onNothingSelected(arg0: AdapterView<*>) {
   }
}
```

Figure 3.29 Spinner code for MainActivity.java file.

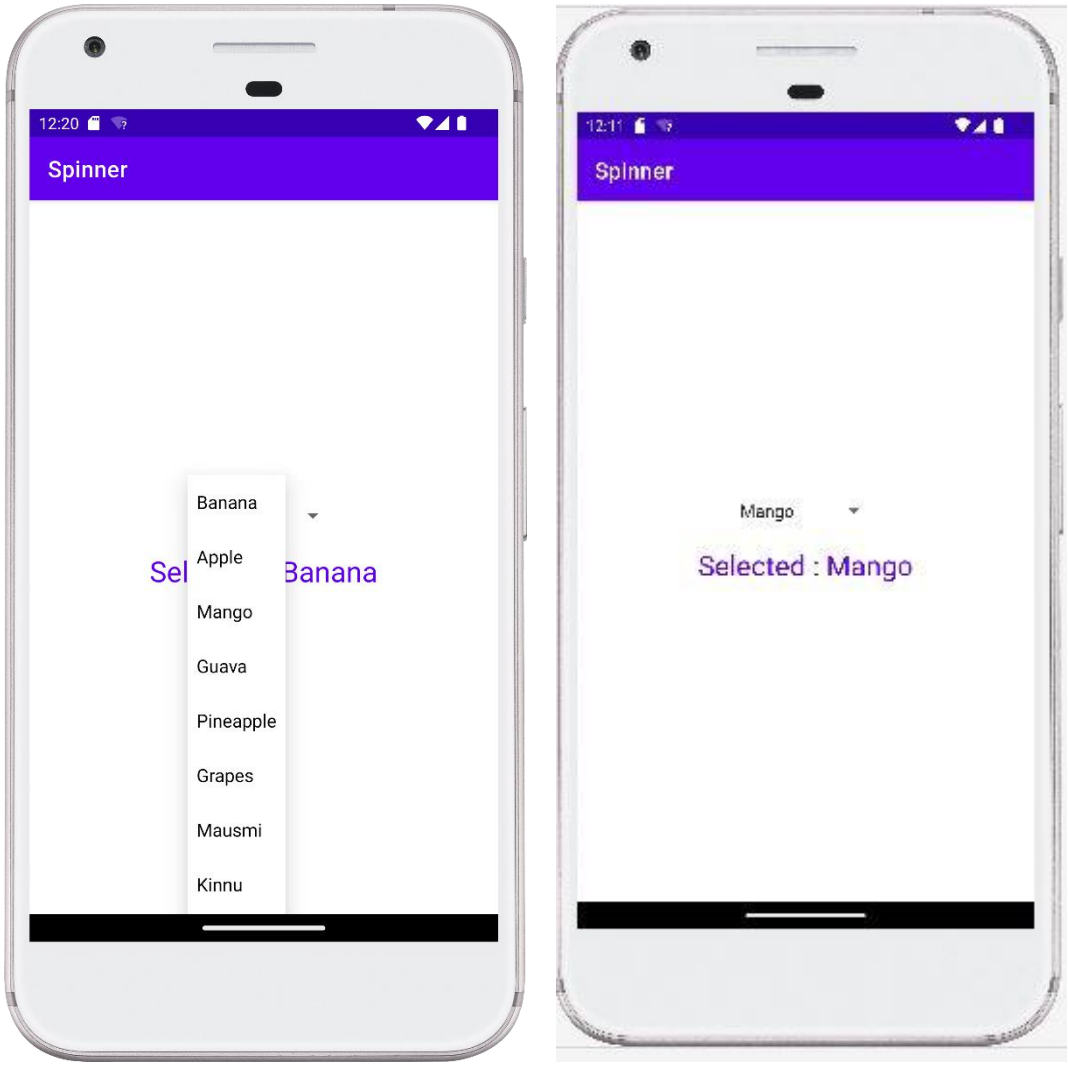


Figure 3.30 Result of Spinner App.

3.11. Summary

This chapter covered the various basic UI components like toast, widget button, spinner, checkbox, and alert dialog. Android Toast is a software development framework that simplifies the process of creating and deploying Android applications. It provides an easy-to-use graphical user interface (GUI) for developing, testing, and releasing apps quickly.

Button is a user interface element in Android that provides the user with a way to interact with the app. It typically consists of a label, an icon, and an action when tapped. There are several types of

buttons available in Android such as ToggleButton, CompoundButton, RadioButton and ImageButton. Each type has its own purpose and use case scenarios.

There are various types of buttons available for use when creating Android apps including TextViews (which display text), CheckBoxes (which allow users to select multiple items from a list), RadioButtons (which allow users to select one item from a list) and AlertDialog boxes (which provide notifications and confirmation dialogs).

A check box is a type of button in Android that allows users to select multiple items from a list. It consists of a label, an icon, and an action when tapped. When the user taps on the checkbox, it will either be checked or unchecked depending on its current state.

An alert dialog is a type of Android widget that displays notifications or confirmation messages to the user. It typically contains two buttons (OK and Cancel) as well as text describing what action should be taken by the user. By tapping one of these buttons, the user can accept or decline whatever message is being displayed onscreen.

A spinner is a type of Android widget that allows users to select one item from a list. It typically consists of an icon, a text label, and an action when tapped. When the user taps on the spinner, it will open up a drop-down menu with available options which can then be selected by tapping one of them.

3.12. Check Your Progress

1. Toast widget is used to check the working of _____.
2. Button has two main properties_____ and _____.
3. Buttons are divided into _____ types.
4. Custom toast is used to add extra _____ components.
5. Toast showing a message on screen for a _____.
6. Toast message can be of_____.
7. Toggle Buttons in android have two states_____ and _____
8. Switch Buttons in android have two states _____ and _____.
9. ImageButton take image by using function _____.
10. For alert Dialog class is used_____.

11. Spinner is working like a _____.
12. Spinner can view at a time _____.

3.13. Self-Assessment Questions

1. What is widget? Explain few basic widgets used for designing an App.
2. Define Android button. Explain various types of buttons.
3. What is toggle button? In which situation we should use toggle button?
4. Discuss checkbox with some basic code snippet.
5. Define Alert Dialog. Why we use alert dialog in App.
6. Explain various type of properties that can be configure in alert dialog.
7. What is Spinner? explain its working in android application.
8. Discuss the widget location in android application with example.

3.14. Answers (Section 3.12)

1. Application quickly
2. Title and onPress
3. four
4. visual
5. #LENGTH_SHORT or #LENGTH_LONG
6. short interval
7. checked/unchecked
8. on /off
9. setImageResource()
10. AlertDialog.Builder
11. dropdown list
12. item

3.15. References/ Suggested reading

1. Headfirst Android Development, Dawn Griffiths, 1st edition, .O′Reilly
2. Android Programming for Beginners, John Horton,2nd edition, .Packt
3. Android Programming with Kotlin for Beginners, John Horton, 1st edition, Packt.

4. Head-First Kotlin, Dawn Griffiths, 1st edition, O'Reilly.
5. Android App Development, Michael Burton, 3rd edition.

SUBJECT: MOBILE APPLICATION DEVELOPEMENT	
COURSE CODE: MCA-42	**AUTHOR: Dr. Sunil Kumar Verma**
LESSON NO. 4	
Advance UI Widgets	

STRUCTURE

LEARNING OBJECTIVE

To understand the use of advance widgets like AutoCompleteTextView, RatingBar used to get review out of few points, DatePicker to get date, and TimePicker to get time from the system, also learn

about ProgressBar to show the processing of file, Quick Contact Budge, to display Analog Clock and Digital Clock. You will learn how to work with hardware Button and Concept of File Download.

Chapter 4. Advance UI Widgets

4.1 Introduction

This chapter introduces advanced Android UI widgets, which are components that allow users to interact with an app. These widgets include auto complete text view, rating bar, date picker, progress bar, file download and more. This chapter will discuss how to create and use these widgets in your Android app. Additionally, this chapter will cover how to customize the look and feel of these widgets, as well as how to handle user input.

4.2. AutoCompleteTextView

AutoCompleteTextView is a class which is inherited from EditText class to inherit all the properties parent class to fulfil all the requirements. This class help in to view auto completion suggestions when user start to enter. The suggestion list displayed from the drop-down menu. The user may select any option to replace the existing content of the textbox. Currently visible drop-down list can be dismissed at any time through not to select any item from list or pressing the back key, or for selection press enter key.

The list of suggested items is taken from the data adaptor. This list become visible when a given threshold limit for number of characters are typed by the user.

A project for creating an AutoCompleteTextView control is shown to give suggestion for city name.

```
<resources>
    <string name="app_name">AutoCompleteTextView</string>
    <string name="hint">City Name</string>
    <string name="submit">Submit</string>
    <string name="submitted_lang">Selected city:</string>
    <string-array name="City">
        <item>Hisar</item>
        <item>Bhiwani</item>
        <item>Sirsa</item>
        <item>Rohtak</item>
        <item>Gurugram</item>
        <item>Faridabad</item>
        <item>Sonipat</item>
        <item>Panipat</item>
        <item>Kaithal</item>
        <item>Karnal</item>
        <item>Kurukshetra</item>
    </string-array>
</resources>
```

Figure 0.1 Code for string.xml file of the AutoCompleteTextView.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/linear_layout"
    android:gravity="center">
    <AutoCompleteTextView
        android:id="@+id/autoTextView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:hint="@string/hint" />
    <Button
        android:id="@+id/btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/submit" />
</LinearLayout>
```

Figure 0.2 Code for activity_main.xml file of the AutoCompleteTextView.

```
package com.example.autocompletetextview
import android.os.Bundle
import android.view.View
import androidx.appcompat.app.AppCompatActivity
import android.widget.ArrayAdapter
import android.widget.AutoCompleteTextView
import android.widget.Button
import android.widget.Toast
class MainActivity : AppCompatActivity() {
   override fun onCreate(savedInstanceState: Bundle?)
   {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      val autotextView
            = findViewById<AutoCompleteTextView>(R.id.autoTextView)
      val city= resources.getStringArray(R.array.City)
      val adapter = ArrayAdapter(this, android.R.layout.simple_list_item_1,city)
      autotextView.setAdapter(adapter)
      val button = findViewById<Button>(R.id.btn)
      if (button != null)
      {
      button ?.setOnClickListener(View.OnClickListener {
         val enteredText = getString(R.string.submitted_lang) + " " +
autotextView.getText()
         Toast.makeText(this@MainActivity, enteredText,
Toast.LENGTH_SHORT).show()
      })
   }
   }
}
```

Figure 0.3 Code for MainActivity.kt file of the AutoCompleteTextView.

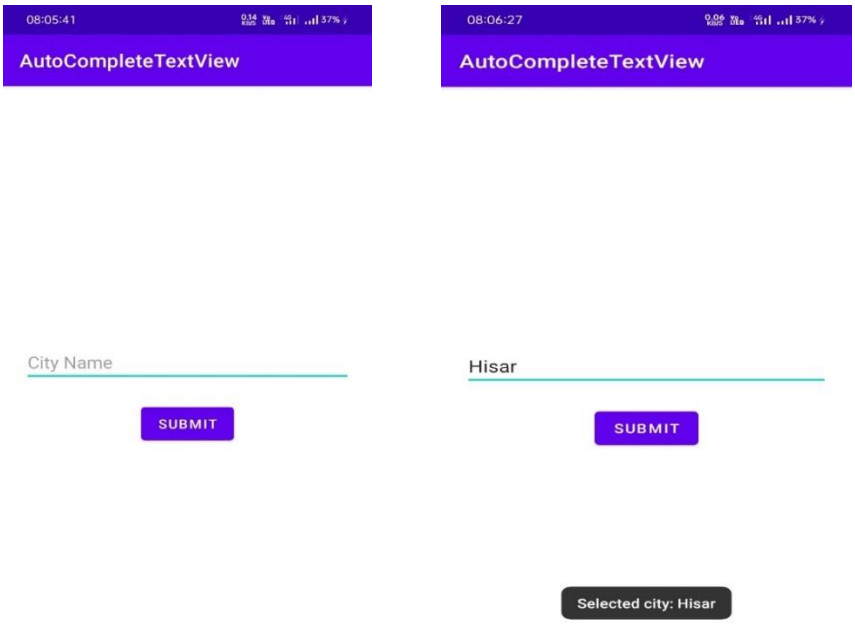


Figure 0.4 Screenshots of AutoCompleteTextView.

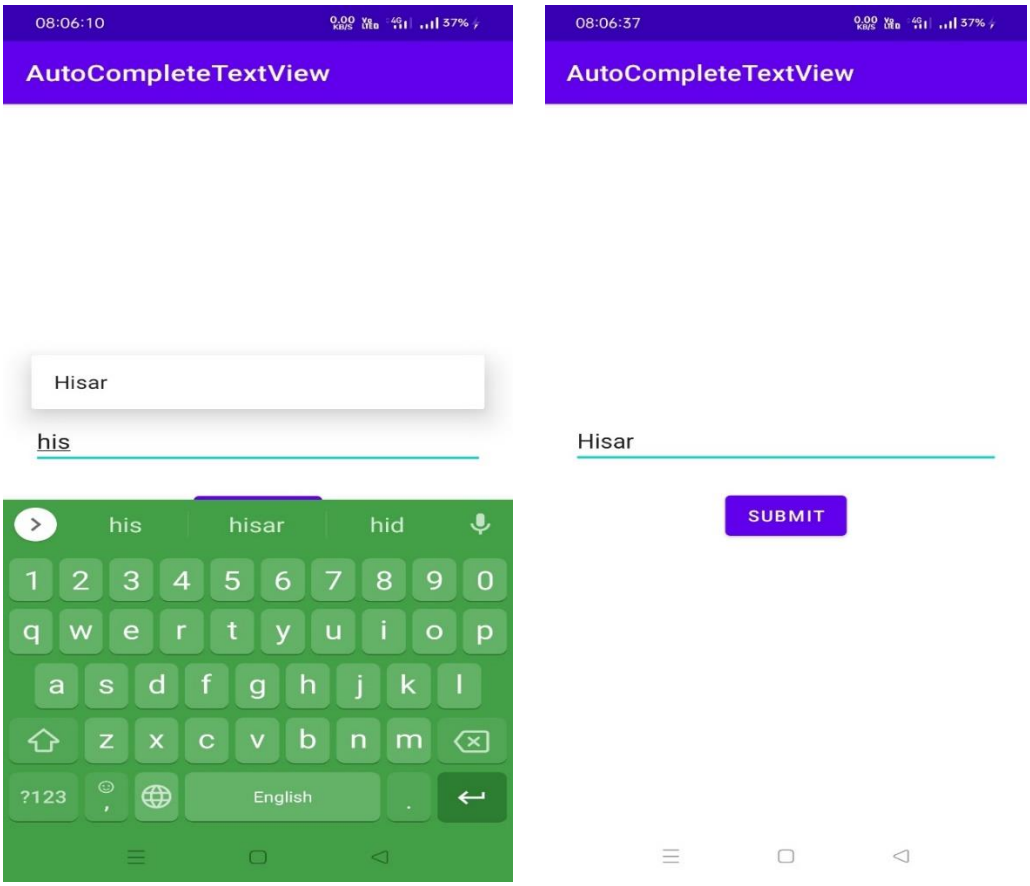


Figure 0.5 Screenshots of AutoCompleteTextView.

4.3. RatingBar

Android RatingBar is an element in widget which is used to collect the rating from the users. It is an expansion of SeekBar and ProgressBar to show the star ratings and this allow the user to give the rating through the stars of the Rating bar.

We can use or set the step size parameter like android:stepSize that will reflect float numbers such as 1.0, 1.5, 2.0, 2.5 etc. The attribute android:numStars helps to specify the counting of stars in RatingBar. RatingBar element is specially used to get the review in rating the form users about the movie, product, hotel experience etc.

RatingBar example is shown below.

```
<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:padding="20sp"
tools:context=".MainActivity">
<RatingBar
   android:id="@+id/ratingBar"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignParentTop="true"
   android:layout_centerHorizontal="true"
   android:numStars="7"
   android:rating="3.0"
   android:stepSize="2.0" />
<Button
   android:id="@+id/buttonCheck"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_below="@id/ratingBar"
   android:layout_centerHorizontal="true"
   android:text="Check Status" />
<TextView
   android:id="@+id/textView"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_centerInParent="true"
   android:textColor="@color/purple_200"
   android:textSize="25sp" />
</RelativeLayout>
```

Figure 0.6 Code for activity_main.xml file of the RatingBar.

```
package com.example.ratingbar
import android.os.Bundle
import android.widget.Button
import android.widget.RatingBar
import android.widget.RatingBar.OnRatingBarChangeListener
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
class MainActivity : AppCompatActivity() {
   lateinit var ratingBar: RatingBar
   lateinit var button: Button
   lateinit var textView: TextView
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      title = "RattingBar App"
      ratingBar = findViewById(R.id.ratingBar)
      ratingBar.numStars = 5
      button = findViewById(R.id.buttonCheck)
      textView = findViewById(R.id.textView)
      ratingBar.onRatingBarChangeListener =
         OnRatingBarChangeListener {_, rating, _ ->Toast.makeText(
               this@MainActivity, "Stars: " + rating.toInt(),
Toast.LENGTH_SHORT).show()
         }
      button.setOnClickListener {
         textView.text = "You got " + ratingBar.rating.toInt()+ " stars"
      }
   }
}
```

Figure 0.7 Code for MainActivity.kt file of the RatingBar.

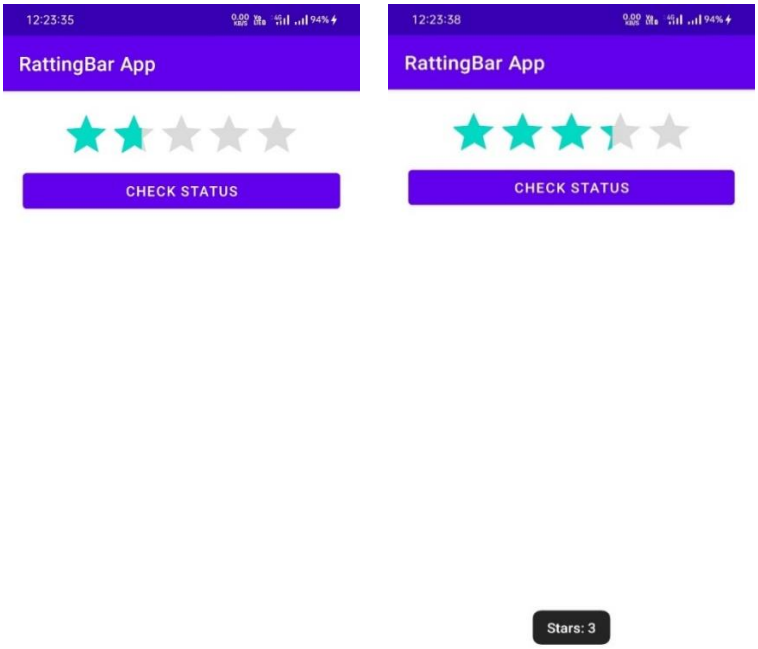


Figure 0.8 Output of the Ratting Bar App.

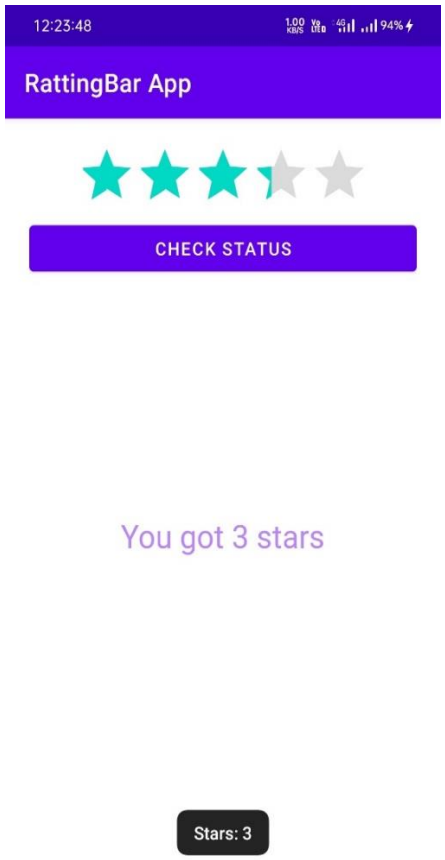


Figure 0.9 Output of the Ratting Bar App.

4.4. DatePicker

Android DatePicker is a user-based interface control which is opened to choose the date by day, month and year in android application. DatePicker is also ensure that the users have enter/select a valid date.

The DatePicker work two modes: one is the showing of complete calendar and second is showing the dates in spinner view. The implementation of DatePicker can be in two ways either in XML or in Activity file.

We can write code to set OnClickListener to a Button to open the DatePickerDialog. When we choose the specific date, the date will save into a variable and displayed in TextView. If the datapicker mode attribute is set to spinner and the date can be selected using day, month and year in spinners or a

CalendarView. The setting of spinners and calendar view can be arranged automatically synchronized. A user can decide what type of view or both he want to use for displaying.

```
package com.example.datepicker
import android.os.Bundle
import android.widget.TextView
import android.view.View
import android.widget.Button
import java.util.*
import android.app.DatePickerDialog
import android.widget.DatePicker
import androidx.appcompat.app.AppCompatActivity
import java.text.SimpleDateFormat
class MainActivity : AppCompatActivity() {
   var button_date: Button? = null
   var textview_date: TextView? = null
   var cal = Calendar.getInstance()
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      textview_date = findViewById<TextView>(R.id.text_view_date_1)
      button_date = findViewById<Button>(R.id.button_date_1)
      textview_date!!.text = "--/--/----"
      val dateSetListener = object : DatePickerDialog.OnDateSetListener {
      override fun onDateSet(view: DatePicker, year: Int, monthOfYear:
Int,dayOfMonth: Int) {
            cal.set(Calendar.YEAR, year)
            cal.set(Calendar.MONTH, monthOfYear)
            cal.set(Calendar.DAY_OF_MONTH, dayOfMonth)
            updateDateInView()
         }
      }
   button_date!!.setOnClickListener(object : View.OnClickListener { override fun
onClick(view: View) {
    DatePickerDialog(this@MainActivity, dateSetListener,
      cal.get(Calendar.YEAR),cal.get(Calendar.MONTH),
       cal.get(Calendar.DAY_OF_MONTH)).show()
         }
      })
   }
   private fun updateDateInView() {
      val myFormat = "MM/dd/yyyy"
      val sdf = SimpleDateFormat(myFormat,Locale.ENGLISH)
      textview_date!!.text = sdf.format(cal.getTime())
   }
}
```

Figure 0.10 Code for MainActivity.kt file of the DatePicker.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:gravity="center"
    android:orientation="vertical"
    tools:context="com.example.datepicker.MainActivity">
    <TextView
        android:id="@+id/text_view_date_1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="35dp"
        android:padding="20dp"
        android:textColor="@color/design_default_color_primary"/>
    <Button
        android:id="@+id/button_date_1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/white"
        android:text="Update Date" />
</LinearLayout>
```

Figure 0.11 Code for activity_main.xml file of the DatePicker.

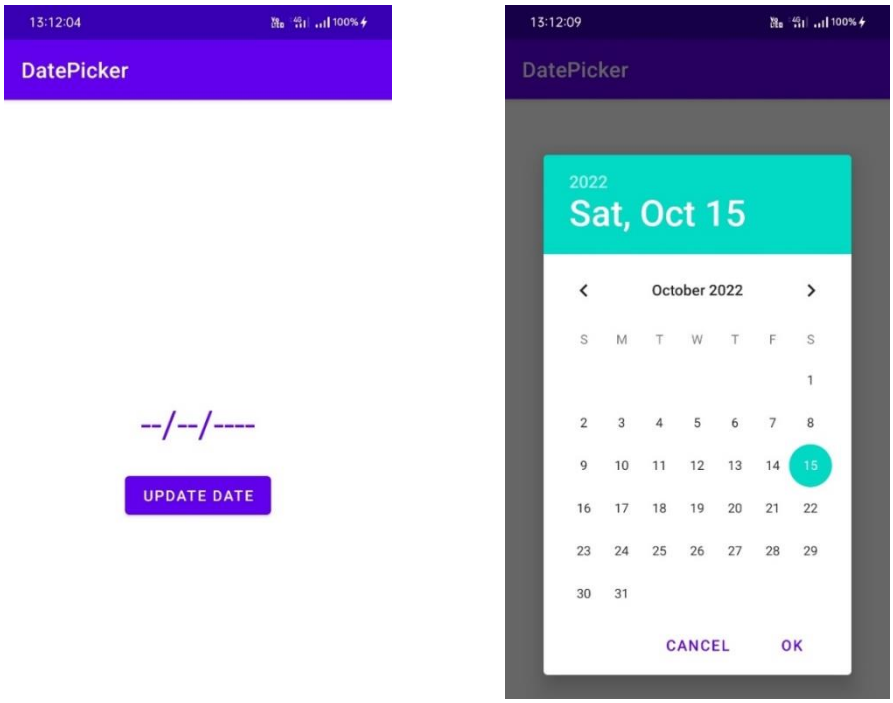


Figure 0.12 Screenshots of DatePicker App.

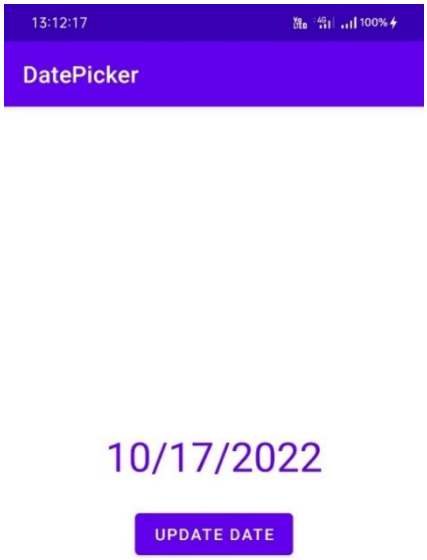


Figure 0.13 Screenshots of the selected date of the DatePicker App.

4.5. Progress Bar

Progress Bar is an interface which is used to show the progress of the application. For example, loading of page, downloading file progress of event to complete.

In this section we will create a progress bar example in Kotlin. It indicates a progress of an operation that is to be performed. This feature supports two modes to show the progress: determinate and indeterminate. The progress of any operation should be in non-interruptive way. User may show the progress at user interface or in notification.

Indeterminate Progress

Indeterminate mode is used for the progress bar when the developer is not sure about how long an operation will go. By default, progress bar is based on indeterminate mode and shows a circular and cyclic animation without any mathematical indication.

Determinate Progress

Determinate mode is used for the progress bar when developer is sure about the activity and want to show a specific quantity of progress that has been completed. For example, during unzip process the percent showing how many contents unzipped, Amount of storing record in database and percentage remaining to download a file from internet.

To indicate determinate progress, the user must set the style of the progress bar in the Kotlin based code of activity file R.style.Widget_ProgressBar_Horizontal and also set the precise value of progress. The setProgress(int) method used to indicate the process of progress bar or can be used incrementProgressBy(int) increased by a specific amount. By default, is 100 to indicate the completion of progress bar. Developer can adjust this value by android:max attribute.

```
<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"
    tools:context=".MainActivity" >
    <ProgressBar
        android:id="@+id/progressBar"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="23dp"
        android:layout_marginTop="20dp"
        android:indeterminate="false"
        android:max="100"
        android:minHeight="80dp"
        android:minWidth="200dp"
        android:progress="1" />
    <ProgressBar
        android:id="@+id/progressBar_cyclic"
        android:layout_width="wrap_content"
        android:minHeight="50dp"
        android:minWidth="50dp"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:layout_height="wrap_content"/>
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/progressBar"
        android:layout_below="@+id/progressBar"/>
</RelativeLayout>
```

Figure 0.14 Code for activity_main.xml file of the ProgressBar.

```
package com.example.progressbar1
import android.os.Bundle
import android.os.Handler
import android.widget.ProgressBar
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
class MainActivity : AppCompatActivity() {
   private var progressBar: ProgressBar? = null
   private var progressStatus = 0
   private var textView: TextView? = null
   private val handler = Handler()
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      progressBar = findViewById(R.id.progressBar) as ProgressBar
      textView = findViewById(R.id.textView) as TextView
      // Start long running operation in a background thread
      Thread {
         while (progressStatus < 100) {
            progressStatus += 1
            // Update the progress bar and display the
            //current value in the text view
            handler.post {
               progressBar!!.progress = progressStatus
               textView!!.text = progressStatus.toString() + "/" + progressBar!!.max
            }
            try {
               // Sleep for 200 milliseconds.
               Thread.sleep(100)
            } catch (e: InterruptedException) {
               e.printStackTrace()
            }
         }
      }.start()
   }
}
```

Figure 0.15 Code for MainActivity.kt file of the ProgressBar.

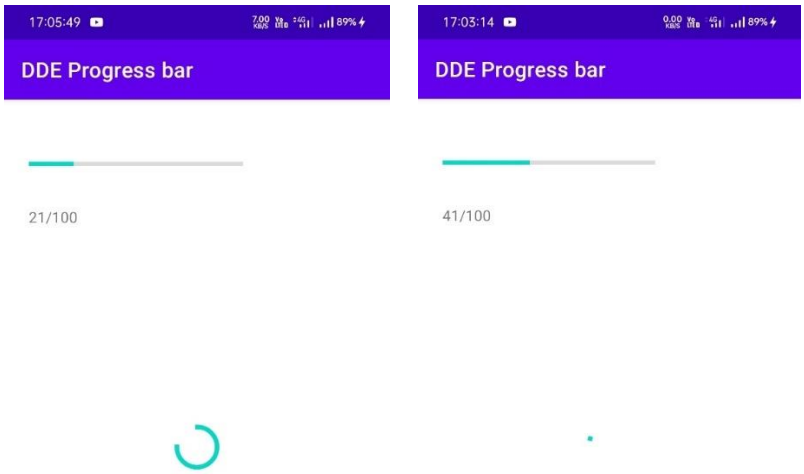


Figure 0.16 Result of ProgressBar.

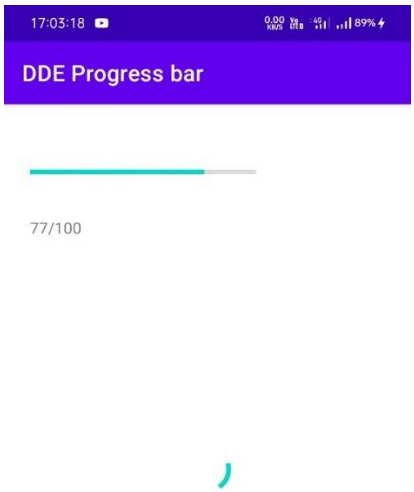


Figure 0.17 Screenshots of the ProgressBar App.

4.6. Quick Contact Budge

Quick contact badge is an advance interface to add contact information through android application to mobile phone directly. Quick contact badge is usually used in app contains user information or advice. With the help of this feature user can store given contact number, email address without focusing on copying process. QuickContactBadge will provide an option to save the contact using one click integration of contact.

```
package com.ddegjust.quickcontactbadge
import android.app.Activity
import android.os.Bundle
import android.provider.ContactsContract
import android.view.View
import android.widget.QuickContactBadge
import android.widget.TextView
class MainActivity : Activity() {
   var Email: TextView? = null
   var Phone: TextView? = null
   var EmailPic: QuickContactBadge? = null
   var PhonePic: QuickContactBadge? = null
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      Email = findViewById<View>(R.id.textView1) as TextView
      Phone = findViewById<View>(R.id.textView2) as TextView
      EmailPic = findViewById<View>(R.id.quickContactBadge1) as
QuickContactBadge
      PhonePic = findViewById<View>(R.id.quickContactBadge2) as
QuickContactBadge
      EmailPic!!.assignContactFromEmail("dde@gjust.ac.in", true)
      EmailPic!!.setMode(ContactsContract.QuickContact.MODE_MEDIUM)
      PhonePic!!.assignContactFromPhone("+919728673500", true)
      PhonePic!!.setMode(ContactsContract.QuickContact.MODE_MEDIUM)
   }
}
```

Figure 0.18 Code for MainActivity.kt file of the QuickContactBadge.

```
<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="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context="com.ddegjust.quickcontactbadge.MainActivity" >
   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="38dp"
      android:text="Email Contact" />
   <QuickContactBadge
      android:id="@+id/quickContactBadge1"
      android:layout_width="100sp"
      android:layout_height="100sp"
      android:layout_below="@+id/textView1"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="18dp"
      android:src="@drawable/b" />
   <TextView
      android:id="@+id/textView2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      android:layout_centerVertical="true"
      android:text="Phone Contact" />
   <QuickContactBadge
      android:id="@+id/quickContactBadge2"
      android:layout_width="100sp"
      android:layout_height="100sp"
      android:layout_below="@+id/textView2"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="20dp"
      android:src="@drawable/c" />
</RelativeLayout>
```

Figure 0.19 Code for activity_main.xml file of the QuickContactBadge.

In existing android application when we click on any small image icon in contact option that open a toolbar with lots of different actions like call, text or email of that person. So, the example of quick contact badge explains it better

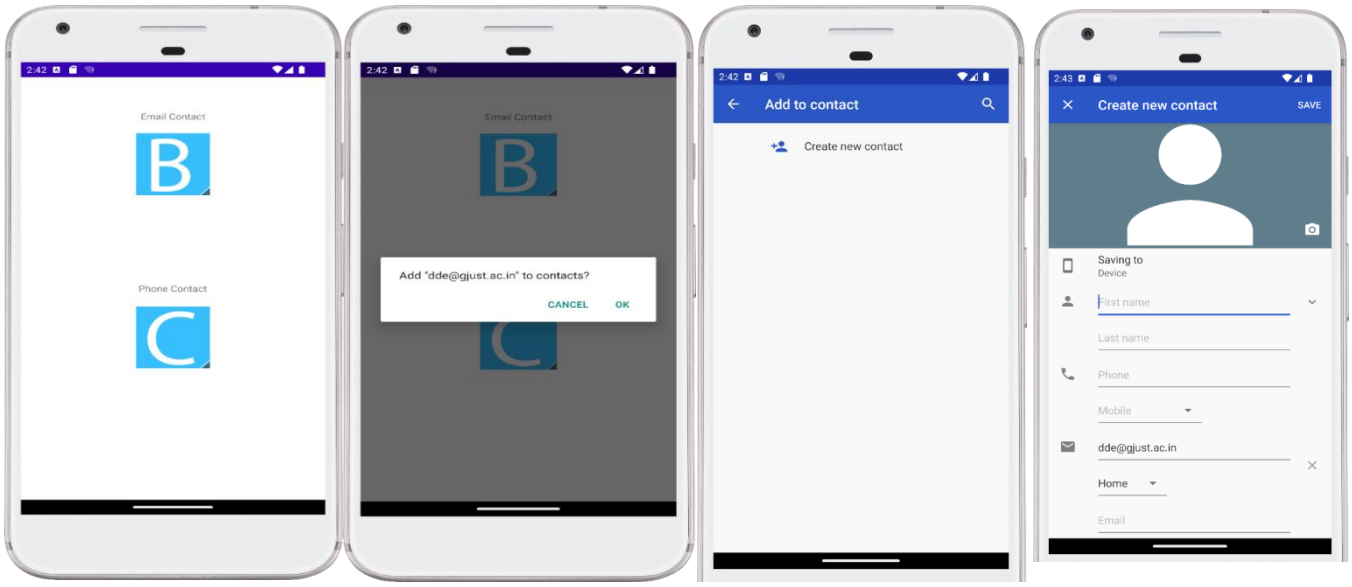
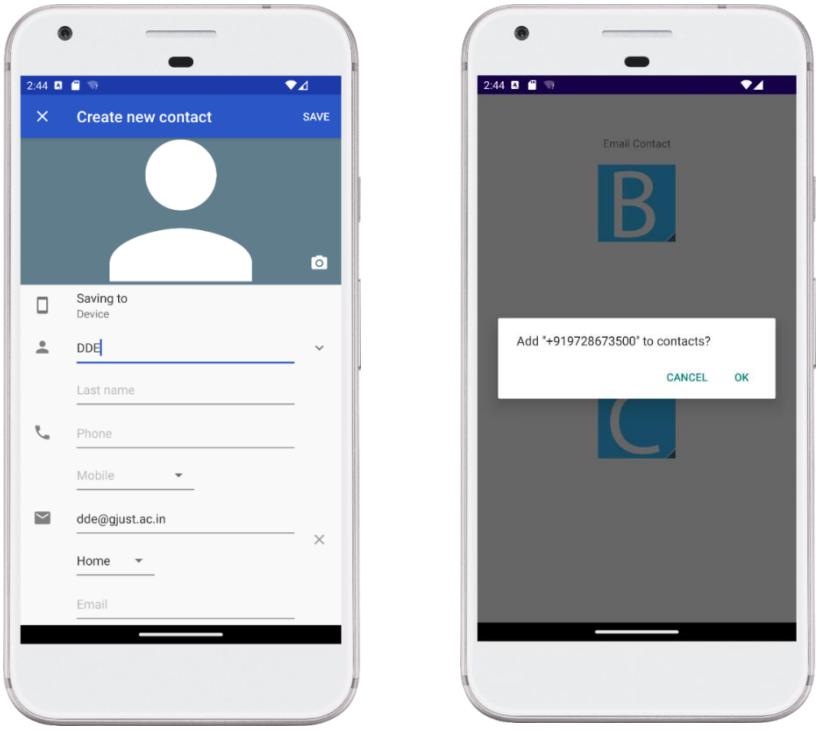


Figure 0.20 Screenshots of the QuickContactBadge



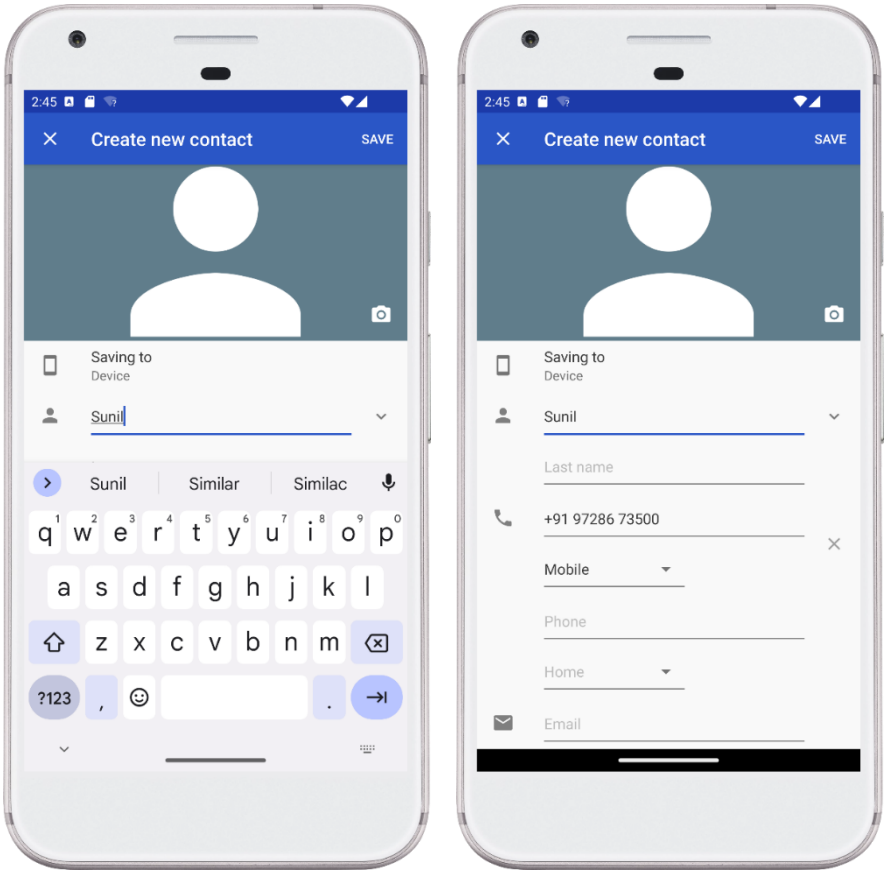


Figure 0.21 Screenshots of the QuickContactBadge.

4.7. Analog Clock and Digital Clock

Clocks components are used in android to display the time. there are three types of clock Analog, Digital and Text.

Analog clock: Analog clock is inherited from the View class. It shows a circular clock which is showing numbers 1 to 12 around the circle to represent the hour. Two types of hands used to indicate the time: smaller is for the hour and longer is for minutes.

Digital clock: Digital clock is inherited from the TextView and uses a set of numbers to display a specific time format in "HH:MM".

The example used in this section explain it better in details. Following step should be followed to complete the components execution.

```
package dde.gjust.analogdigital
import android.os.Bundle
import android.view.View
import android.widget.AnalogClock
import android.widget.DigitalClock
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
class MainActivity : AppCompatActivity() {
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      val simpleDigitalClock = findViewById<View>(R.id.simpleDigitalClock) as
DigitalClock
      val simpleAnalogClock = findViewById<View>(R.id.simpleAnalogClock) as
AnalogClock
        simpleAnalogClock.setOnClickListener {
         Toast.makeText(this@MainActivity, "Analog Clock",
Toast.LENGTH_SHORT)
            .show()
      }
     simpleDigitalClock.setOnClickListener {
         Toast.makeText(this@MainActivity, "Digital Clock",
Toast.LENGTH_SHORT)
            .show()
      }
   }
}
```

Figure 0.22 Code for MainActivity.kt file of the Analog and Digital Clock.

```
<resources>
   <string name="app_name">AnalogAndDigital</string>
   <string name="hello_world">Hello world!</string>
   <string name="action_settings">Settings</string>
   <string name="analogClock">Analog Clock</string>
   <string name="digitalClock" >Digital Clock</string>
</resources>
```

Figure 0.23 Code for String.xml file of the Analog and digital clock.

```
<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"
 tools:context=".MainActivity">
 <TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:text="@string/analogClock"  android:textSize="25sp"
    android:textStyle="bold" />
 <AnalogClock
    android:id="@+id/simpleAnalogClock"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="50dp"  android:background="#ff0"
    android:padding="50dp" />
 <TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/simpleAnalogClock"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="50dp"
    android:text="@string/digitalClock"
    android:textSize="25sp" android:textStyle="bold" />
 <DigitalClock
    android:id="@+id/simpleDigitalClock"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView2"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="20dp"
    android:background="#ff0"  android:padding="20dp"
    android:textColor="#f0f"  android:textSize="25sp"
    android:textStyle="bold" />
</RelativeLayout>
```

Figure 0.24 Code for activity_main.xml file of the Analog and digital Clock.

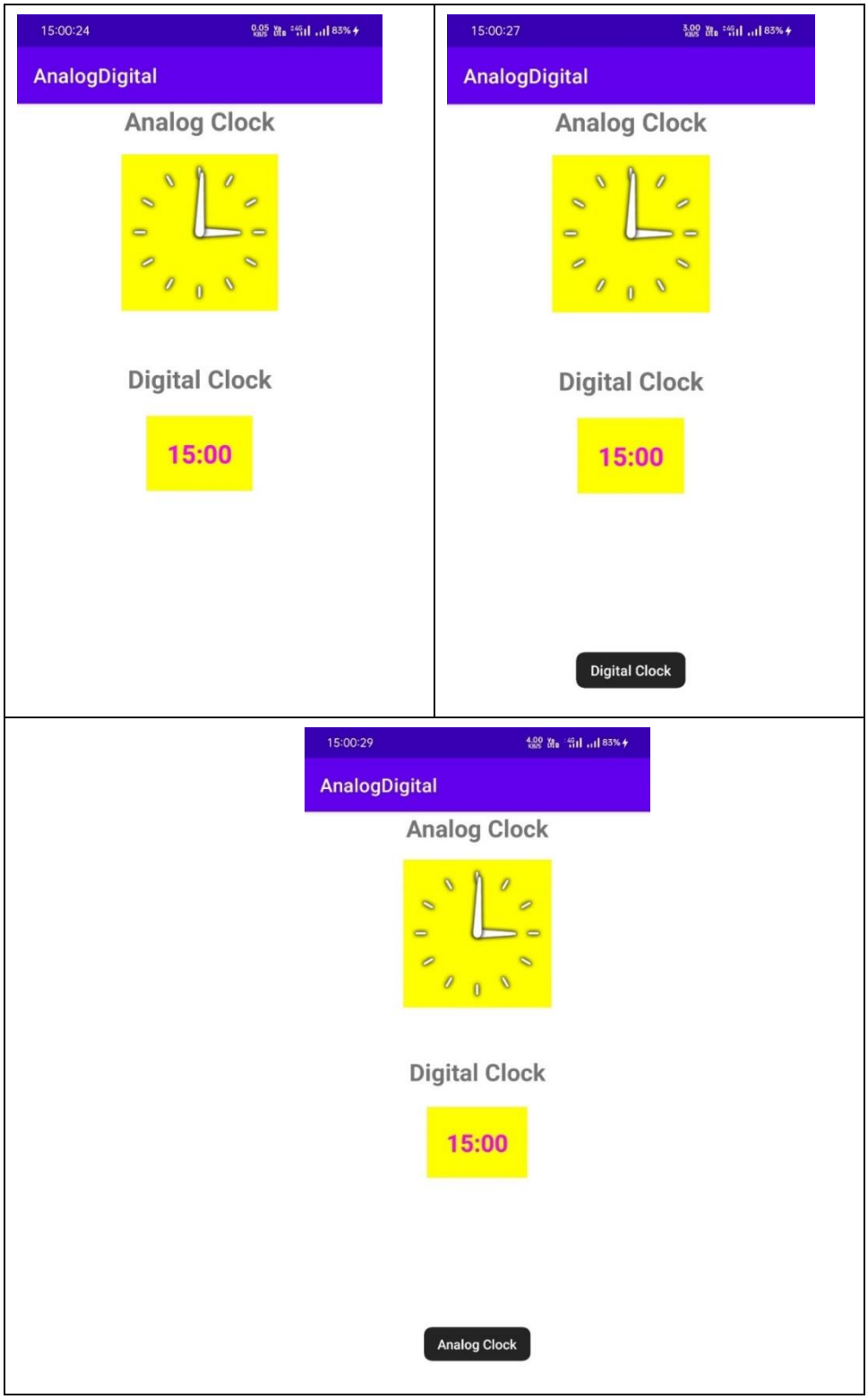


Figure 0.25 Output of the Analog and Digital Clock.

4.8. Working with hardware Button

A wear OS based device contains multiple physical buttons. These physical buttons are known as stems. Wear OS devices usually have at least one button that is power button. There may be multiple functional buttons present with devices.

Different types of configurations can be used for each of these button types. OS buttons: It is reserved for system-based actions like turning power on/off. Multifunction buttons have multifunctional feature like single press, press and hold, press for a prespecified second to activate any action.

Sometimes bination actions are performed like play/pause. The user primarily uses your app without the user looking at the display. These keys are also helpful to execute any event without looking at the mobile screen. Single press button is a "start" and "stop" button on stopwatch design through binary action

To get the information about number of buttons available on the device you can use the method WearableButtons.getButtonCount(). There are multifunction buttons are also available. So, we can minus one from the count of all buttons because the first button is usually the power button.

Keycodes for each button is mapped to an integer constant value from the class KeyEvent. The detail shown below with keycode property value

MF button 1 KEYCODE_STEM_1

MF button 2 KEYCODE_STEM_2

MF button 3 KEYCODE_STEM_3

We can get these keycode and apply to some targeted action. To check the button, press status use the onKeyDown() method.

A new device launched with specific physical keys, such as back key, home key, power key, volume keys etc. These keys respond when you press it. As we know that volume key increase and decrease volume on a single press with some pre-specified step. Power key locks the device on a single press, while on a long press switches on or off the device. This section designs an application to confirm the pressing of hardware key with some notification through toast message.

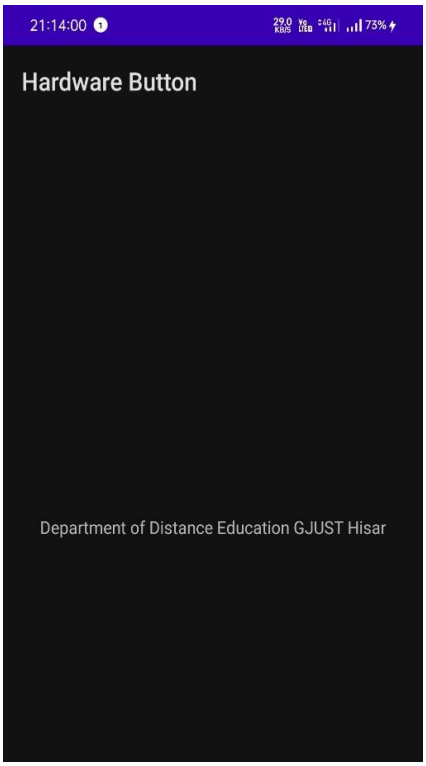
In every application there can be a special use of same key as multi-functional. Google assistant open when home key is pressed for a long. Volume key can be used to set screen brightness. We can configure key in mobile settings for shortcut use.

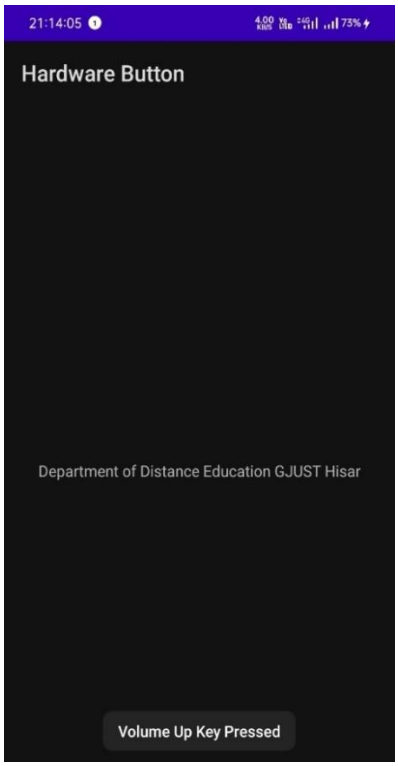
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Department of Distance Education GJUST Hisar"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      tools:ignore="HardcodedText" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

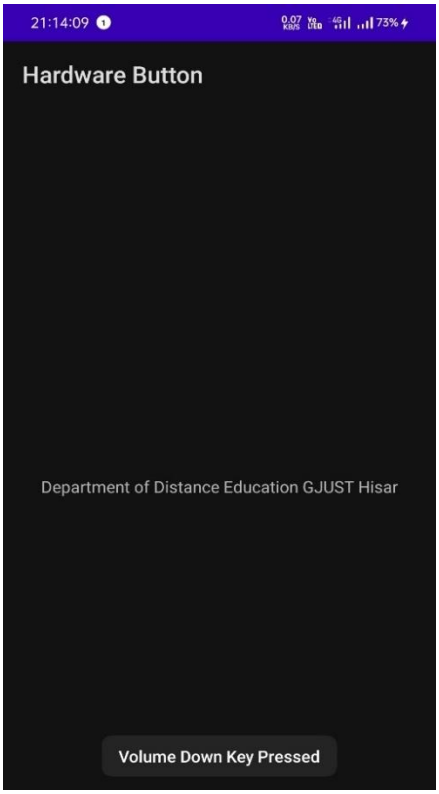
Figure 0.26 Code for activity_main.xml file of the Hardware button.

```
package com.example.filedownload
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.KeyEvent
import android.widget.Toast
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
override fun onKeyDown(keyCode: Int, event: KeyEvent?): Boolean {
       when (keyCode) {
            KeyEvent.KEYCODE_VOLUME_DOWN ->
Toast.makeText(applicationContext, "Volume Down Key Pressed",
Toast.LENGTH_SHORT).show()
            KeyEvent.KEYCODE_VOLUME_UP ->
Toast.makeText(applicationContext, "Volume Up Key Pressed",
Toast.LENGTH_SHORT).show()
            KeyEvent.KEYCODE_BACK ->
Toast.makeText(applicationContext, "Back Key Pressed",
Toast.LENGTH_SHORT).show()
        }
        return true
    }
}
```

Figure 0.27 Code for MainActivity.kt file of Hardware button App.


21:14:00
Hardware Button
Department of Distance Education GJUST Hisar


21:14:05
Hardware Button
Department of Distance Education GJUST Hisar
Volume Up Key Pressed


21:14:09
Hardware Button
Department of Distance Education GJUST Hisar
Volume Down Key Pressed

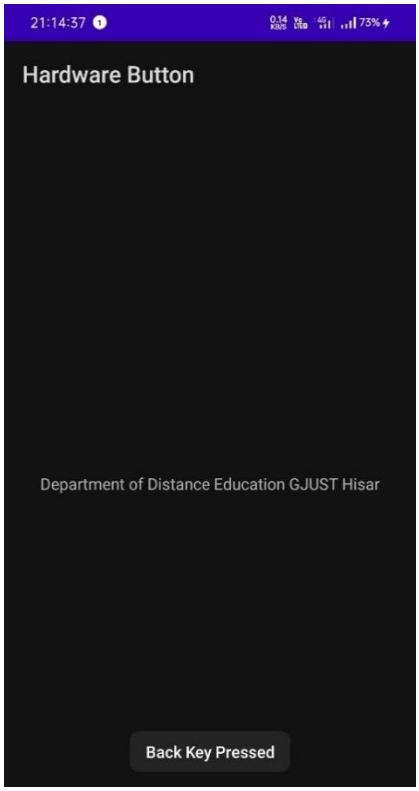

21:14:37
Hardware Button
Department of Distance Education GJUST Hisar
Back Key Pressed

Figure 0.28 Screenshot of the Hardware button App.

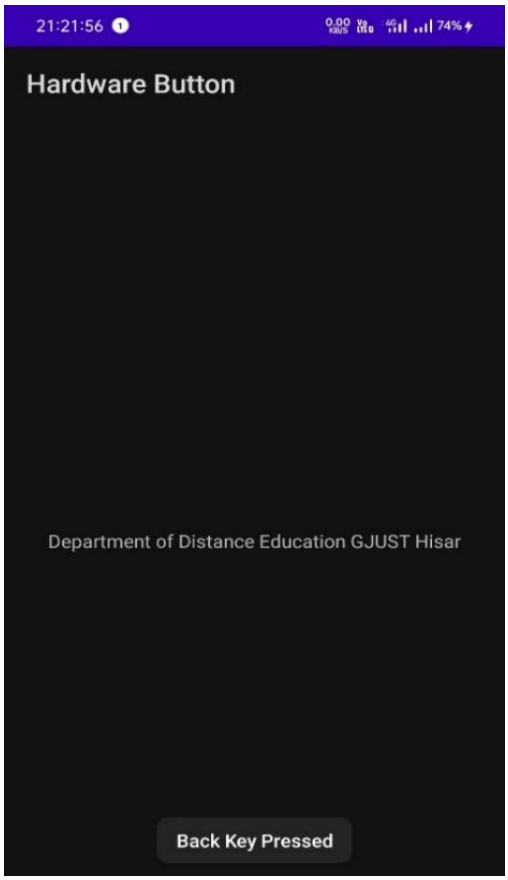


Figure 0.29 Code screenshot of hardware button App.

4.9. File Download.

File download is a special type of interface to interact with internet activity. A DownloadManager class is used to perform the download action. Here, we are going to learn how to download files from the URL using DownloadManager class in android. Here we will just copy the file link and paste in the text filed. When we click on the Button it will be downloaded automatically to our phone storage. This is a common way to download file or we can also download FAQ of any app automatically

Almost steps are same except the AndroidManifest.xml file updating. Here we have set the internet permission to download the file.

<uses-permission android: name=”android.permission.INTERNET”/>

Remaining you can copy and paste the code in relevant file.

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

Figure 0.30 Add following code line in AndroidManifest.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   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:layout_gravity="center"
   android:gravity="center"
   android:orientation="vertical"
   tools:context=".MainActivity">
   <EditText
      android:id="@+id/textlink"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_marginTop="100dp"
      android:layout_marginStart="20dp"
      android:layout_marginEnd="20dp"
      android:hint="Enter link here"
      android:padding="10dp"
      android:textColor="@color/black"
      android:textSize="20sp"        />
   <Button
      android:id="@+id/download"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Download File" />
</LinearLayout>
```

Figure 0.31 Code for activity_main.xml file of the File Download.

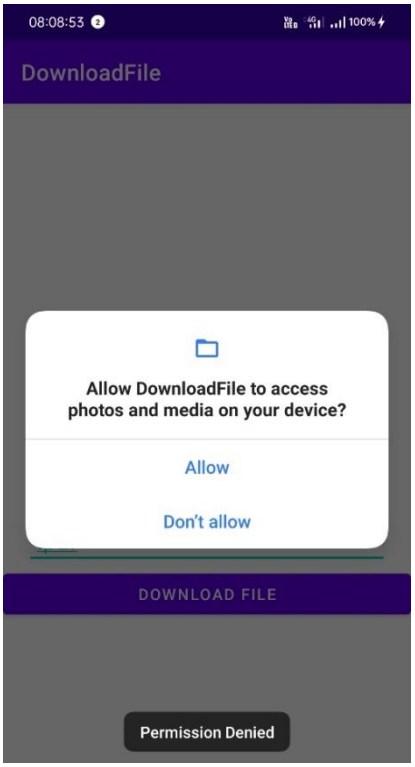
```
package dde.gjust.downloadfile
import android.app.DownloadManager, import android.content.Context
import android.net.Uri, import android.os.Bundle
import android.os.Environment , import android.view.View
import android.widget.EditText, import android.widget.Button
import android.widget.Toast
import androidx.activity.result.contract.ActivityResultContracts
import androidx.appcompat.app.AppCompatActivity
import java.io.File
class MainActivity : AppCompatActivity() {
   private lateinit var downButton: Button
   private lateinit var enterlink: EditText
   private var permission=0
   private val requestPermissionLauncher=registerForActivityResult
           (ActivityResultContracts.RequestPermission())
   {permission=if(it)  { 1  } else { 0  } }
  protected override fun onCreate(savedInstanceState: Bundle?) {
     super.onCreate(savedInstanceState)
     setContentView(R.layout.activity_main)
     downButton = findViewById(R.id.download)
     enterlink = findViewById(R.id.textlink)
     downButton.setOnClickListener {
requestPermissionLauncher.launch(android.Manifest.permission.WRITE_EXTERNAL_STORAG
E)
    if (permission == 1) {
    download(enterlink.text.toString(), "DDE Programmer")
    } else {
  Toast.makeText(this, "Permission Denied", Toast.LENGTH_LONG).show()
  }  }  }
   private fun download(url: String , filename: String) {
      try {
         var manager = getSystemService(Context.DOWNLOAD_SERVICE) as
DownloadManager
         val uri: Uri = Uri.parse(url)
         val request = DownloadManager.Request(uri)
    request.setAllowedNetworkTypes(DownloadManager.Request.NETWORK_MOBILE or
DownloadManager.Request.NETWORK_WIFI)
            .setMimeType("image/jpeg")
            .setAllowedOverRoaming(false)
            .setTitle(filename)
.setNotificationVisibility(DownloadManager.Request.VISIBILITY_VISIBLE_NOTIFY_COMPLE
TED)
            .setDestinationInExternalPublicDir(
              Environment.DIRECTORY_PICTURES,
              File.separator + filename
            )
         manager.enqueue(request)
         Toast.makeText(this, "Image downloaded", Toast.LENGTH_LONG).show()
      } catch (e: Exception) {
         Toast.makeText(this, "image download failed", Toast.LENGTH_LONG).show()
}   } }
```

Figure 0.32 Code for MainActivity.kt file of the File Download.


08:08:01
https://www.gjust.ac.in
/archiveddata/Downloads/2015
/05/Prospectus_2015_250515
.pdf


08:07:53
DownloadFile
https://www.gjust.ac.in
/archiveddata/Downloads/2015
/05/Prospectus_2015_250515
.pdf
DOWNLOAD FILE


08:08:12
DownloadFile
https://www.gjust.ac.in
/archiveddata/Downloads/2015
/05/Prospectus_2015_250515
.pdf
DOWNLOAD FILE
Permission Denied


08:08:53
DownloadFile
Allow DownloadFile to access
photos and media on your device?
Allow
Don't allow
DOWNLOAD FILE
Permission Denied

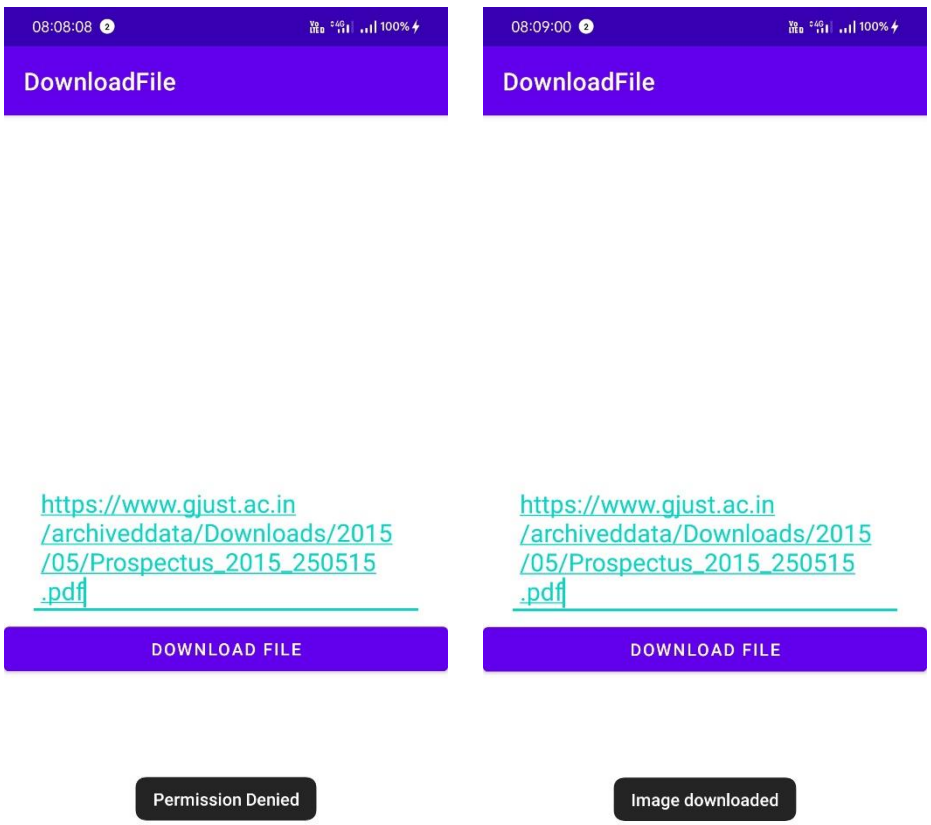


Figure 0.33 Screenshots of File download App.

4.10. Summary

This chapter has covered all the advance topic of UI widgets. Auto complete text view is used to select the complete name of content as mentioned in the list. Ratting bar is used to collect the feedback from user for pre specified range. Progress bar indicate the process completion ratio. This chapter also discussed the Quick contact badge which is used to save the contact number quickly in contact list. After the few topics such as Analog clock, digital clock, detection of hardware button pressing, file downloading from internet were discussed. Overall, this chapter update the reader with recent and most important components of android application

4.11. Check Your Progress

1. AutoCompleteTextView is a subclass of _____.
2. For auto completion of text, the suggestion items taken from _____.

3. Rating bar is an expansion of _____.
4. For step size of rating bar _____ tag is used.
5. To specify the counting of star _____ tag used.
6. DatePicker class is used to set _____.
7. To selected the specific date _____ class is used.
8. Android studio program can be written in _____ language.
9. When application operation time not confirmed then _____ mode is used.
10. _____ is used to add contact through android application.
11. Analog Clock is inherited from _____.
12. Digital Clock is inherited from _____.
13. Hardware button is used to perform _____.
14. A file is downloaded using _____.

4.12. Self-Assessment Questions

1. What is the use of AutoCompleteTextView class in designing app.
2. Discuss the various types of rating used to get feedback from user.
3. Explain the working of DatePicker class.
4. What is progress bar? Discuss both types of progress bar.
5. Why we use QuickContactBadge class in App?
6. Explain the various attributes and classes of Analog and Digital classes.
7. Mention the name of wear code used to detect hardware key pressed or not. Explain in detail.
8. How file downloading is performed i application? Explain.

4.13. Answers (Section 5.11)

1. EditText
2. DataAdapter
3. SeekBar and ProgressBar
4. android:stepSize
5. android:numStars

6. Date
7. DatePickerDialog
8. Java and Kotlin
9. Indeterminate
10. View class
11. TextView class
12. Multiple function
13. DownloadManager

4.14. References/ Suggested reading

1. Headfirst Android Development, Dawn Griffiths, 1st edition, .O′Reilly
2. Android Programming for Beginners, John Horton,2nd edition, .Packt
3. Android Programming with Kotlin for Beginners, John Horton, 1st edition, Packt.

SUBJECT: MOBILE APPLICATION DEVELOPEMENT

COURSE CODE: MCA-42	AUTHOR: Dr. Sunil Kumar Verma
LESSON NO. 5	

Android Activity, Intent & Fragment

STRUCTURE

LEARNING OBJECTIVE

To understand and learn about Activity, Intent & Fragment. How activity handle the all components of android application. Also learn the use of intent and its type. How to divide android activity areas in different fragments.

Chapter 5. Activity, Intent & Fragment

5.1. Introduction

This chapter discussed about the core components of Android App like Activities, Fragments, and Intents. An activity is a user interface component that is mainly used to construct a single screen of the application and represents the main focus of attention on a screen. An activity can host one or more fragments at a time. Fragments, as tablets emerged with larger screens, are reusable components that are attached to and displayed within activities. It is basically a piece of an activity that enables more modular activity design. We can call a fragment is a kind of sub-activity. It is always hosted by an activity. It has its own layout and its own behavior with its own life cycle callbacks. We can add or remove fragments in an activity while the activity is running. It is possible to develop a UI only using Activities, but this is generally a bad idea since their code cannot later be reused within other Activities, and cannot support multiple screens. Activity is the UI of an application through which user can interact and Fragment is the part of the Activity, it is a sub-Activity inside activity which has its own Life Cycle which runs parallel to the Activities Life Cycle.

5.2. Activities

- Activity is one of the most important components for any android app.
- Activities are the User Interface (UI) screens which user see.
- It is similar to the main() function in different programming languages.
- It is the main entry point for user interaction.
- You can have multiple activities in your app.
- All your activities must be declared in the manifest file, with their attributes.

Every activity has different functions throughout its life, onCreate(), onStart(), onResume(), onPause(), onStop(), onRestart(), onDestroy().

Below is the image from android's documentation, which clearly shows the lifecycle of an android activity.

5.3. Fragments

In most of the applications these days, fragments are largely used. As there are a lot of android devices with different resolutions, it's a bit tough to handle all of those, that's where fragments come handy. We can combine 2 or more fragments and show them in an activity.

A Fragment is a component that is used by an activity. Even though it is used by an activity, it has its own lifecycle.

There are different types of fragments which you can extend:

- DialogFragment
- ListFragment

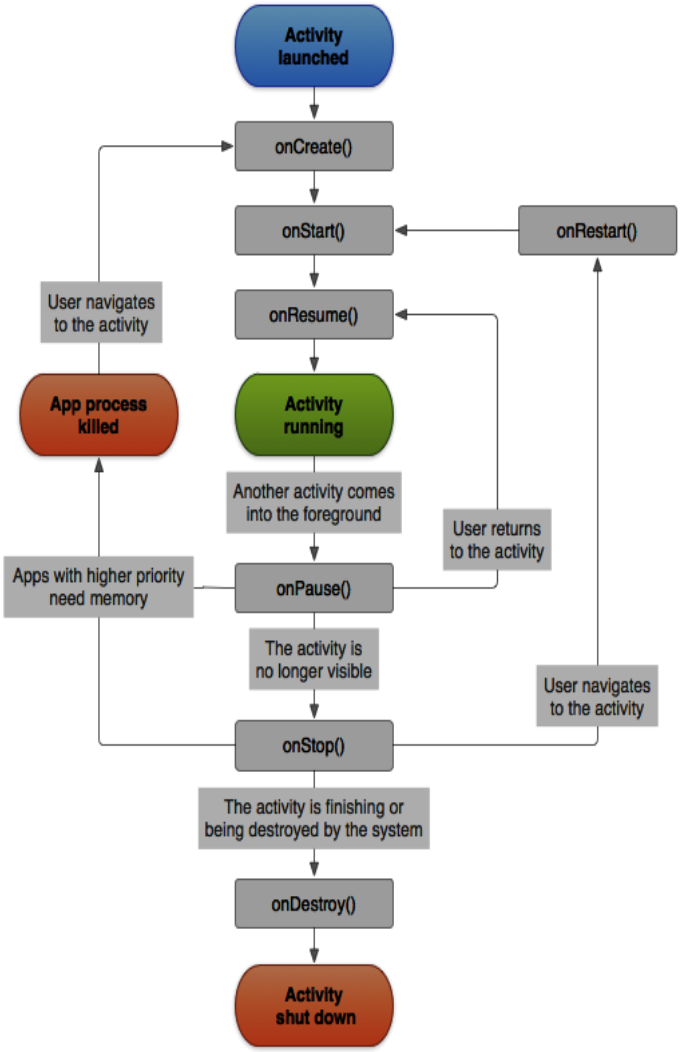


Figure 5.1 Lifecycle of Android Activity.

- PreferenceFragment.

Here are the important things to understand about fragments:

- A Fragment is a combination of an XML layout file and a java class much like an Activity.
- Using the support library, fragments are supported back to all relevant Android versions.
- Fragments encapsulate views and logic so that it is easier to reuse within activities.
- Fragments are standalone components that can contain views, events and logic.

Within a fragment-oriented architecture, activities become navigational containers that are primarily responsible for navigation to other activities, presenting fragments and passing data.

The main classes used for fragments are:

FragmentActivity: It is a base class for activities.

Fragment: It is a base class for each fragment.

FragmentManager: It handles object inside activity

FragmentTransaction: It is sued to perform atomic operations like replace/add a fragment.

Importance of Fragments

There are many use cases for fragments but the most common use cases include:

Reusing View and Logic Components - Fragments enable re-use of parts of your screen including views and event logic over and over in different ways across many disparate activities. For example, using the same list across different data sources within an app.

Tablet Support - Often within apps, the tablet version of an activity has a substantially different layout from the phone version which is different from the TV version. Fragments enable device-specific activities to reuse shared elements while also having differences.

Screen Orientation - Often within apps, the portrait version of an activity has a substantially different layout from the landscape version. Fragments enable both orientations to reuse shared elements while also having differences

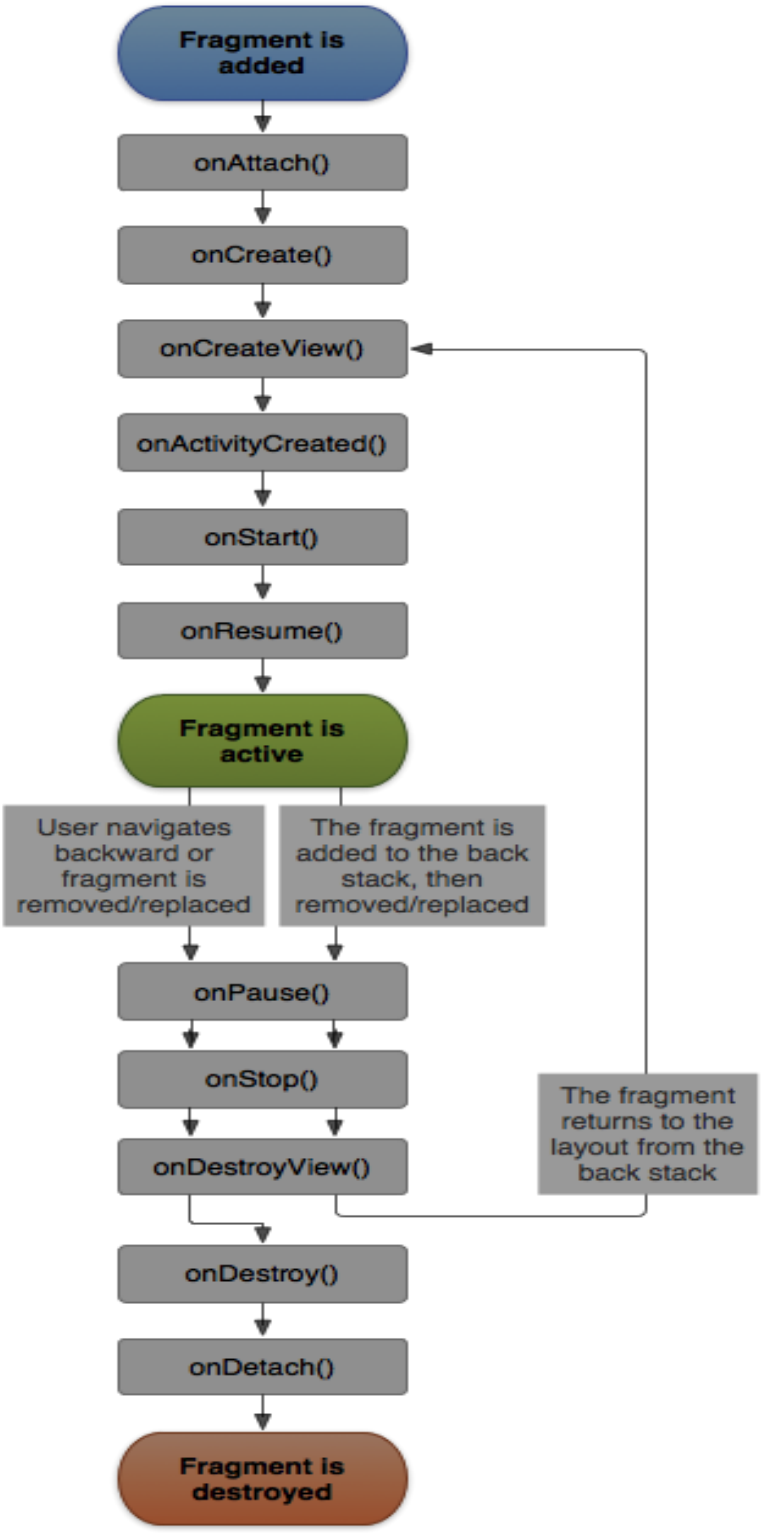


Figure 5.2 Life Cycle of the Android Fragment.

5.4. Intents

Intent is one of the most important and most used app components of an android application. Using Intents, you can call to other app components or to other activity or also call other applications on your mobile phone. It is used to passing information from one activity to another activity that means it is data carrier through object.

Intent intentobj = new Intent(getApplicationContext(), AnotherActivity.class);

startActivity(intentobj);

Intents are of two types:

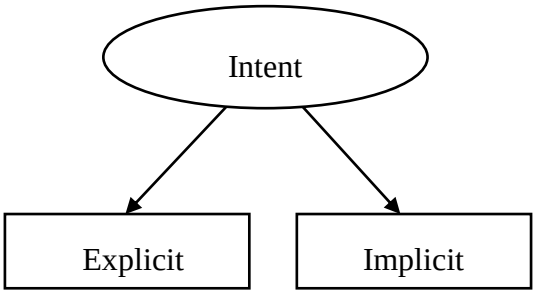


Figure 5.3 Types of Intent.

Explicit

Explicit Intents where you call another activity or something with a class name. For instance, you can call another activity when some action happened in one activity. So, you here explicitly specify which activity to call.

Implicit

Implicit Intents where we do not specify a class name but specify some sort of action, which can be handled by some other inbuilt apps or some other apps. For instance, you may want to open a camera, showing a map, sending emails etc. Here, you don't directly call camera app or map app, you will just specify the action.

All these three components are explained with one example where we are switching from one activity to next activity through intent and every activity may have one or two fragments. Below is a simple application which has one button in one activity and when clicked to goes to other activity using intent and in the other activity, two fragments are shown.

Now an example of an Activity, Intent, Fragment are discussed below with code and results. Let's start to implement the Intent to show the basic use. In this example we will move from first activity to next and open a web page on new activity. This example covers both type of Intent.

5.5. Activity Example

```
package gjust.dde.activity;
import android.os.Bundle;
import android.util.Log;
import androidx.appcompat.app.AppCompatActivity;
public class HomeActivity extends AppCompatActivity {
   private static final String HOME_ACTIVITY_TAG =
HomeActivity.class.getSimpleName();
  private void showLog(String text){
     Log.d(HOME_ACTIVITY_TAG, text);
  }
  @Override    public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     showLog("Activity Created");   }
  @Override    protected void onRestart(){
     super.onRestart();//call to restart after onStop
     showLog("Activity restarted");   }
  @Override    protected void onStart() {
     super.onStart();//soon be visible
     showLog("Activity started");   }
  @Override    protected void onResume() {
     super.onResume();//visible
     showLog("Activity resumed");
  }
  @Override    protected void onPause() {
     super.onPause();//invisible
     showLog("Activity paused");

  }
  @Override    protected void onStop() {
     super.onStop();
     showLog("Activity stopped");
  }
  @Override    protected void onDestroy() {
     super.onDestroy();
     showLog("Activity is being destroyed");
  }
}
```

Figure 5.4 Code for MainActivity.java of the Activity.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Activity"
        tools:targetApi="31">
        <activity
            android:name=".HomeActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
        </activity>
    </application>
</manifest>
```

Figure 5.5 Code for activity_main.xml of the Activity.

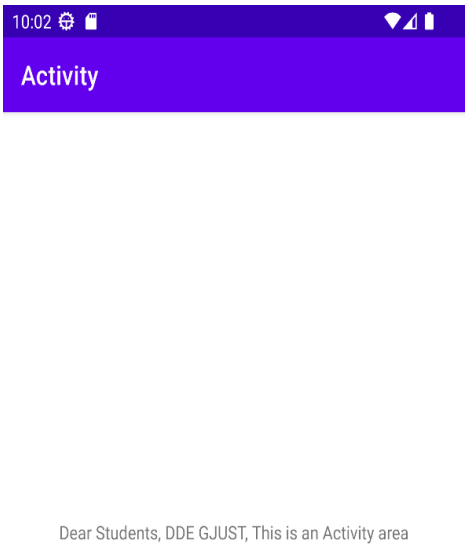


Figure 5.6 Screenshots of the Activity App.

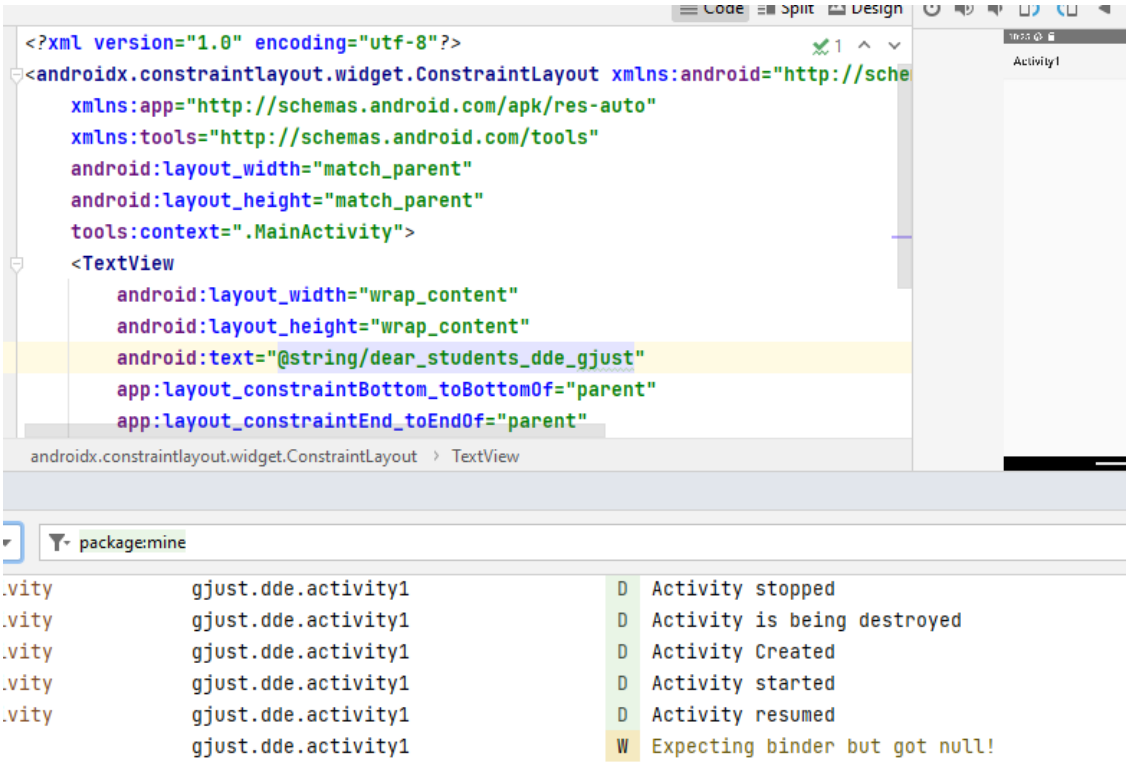


Figure 5.7 Screenshots activity creation.

5.6. Intent example

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.intents" >
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.Light" >
        <activity android:name="gjust.dde.intent.MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="gjust.dde.intent.SecondActivity" >

        </activity>
    </application>
</manifest>
```

Figure 5.8 Code for Manifest.xml file of the Intent.

```
package gjust.dde.intent;
import android.os.Bundle;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
public class SecondActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.secondactivity);
        Toast.makeText(getApplicationContext(), "Now you are at second
Activity",Toast.LENGTH_LONG).show();
    }
}
```

Figure 5.9 Code for SecondActivity.java file of the Intent.

```
<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:paddingLeft="10dp"
   android:paddingRight="10dp"
   android:paddingTop="10dp"
   android:paddingBottom="10dp" tools:context=".MainActivity">
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceMedium"
      android:text="@string/now_check_the_both_activity_operations"
      android:layout_centerHorizontal="true"
      android:textAlignment="center"
      android:id="@+id/textView2"
      android:clickable="false"
      android:layout_alignParentTop="true"
      android:layout_alignParentStart="true"
      android:layout_marginTop="42dp"
      android:background="#3e7d02"
      android:textColor="#ffffff" />
   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/explicit_intent"
      android:id="@+id/explicit_Intent"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="147dp" />
   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/implicit_intent"
      android:id="@+id/implicit_Intent"
      android:layout_centerVertical="true"
      android:layout_centerHorizontal="true" />
</RelativeLayout>
```

Figure 5.10 Code for activity_main.xml file of the Intent.

```
package gjust.dde.intent;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.widget.Button;
public class MainActivity extends AppCompatActivity {
   Button explicit_btn, implicit_btn;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      explicit_btn = (Button)findViewById(R.id.explicit_Intent);
      implicit_btn = (Button) findViewById(R.id.implicit_Intent);
      explicit_btn.setOnClickListener(v -> {
         Intent intent = new  Intent(getBaseContext(),
SecondActivity.class);
         startActivity(intent);
      });
      implicit_btn.setOnClickListener(v -> {
         Intent intent = new Intent(Intent.ACTION_VIEW);
         intent.setData(Uri.parse("https://www.gjust.ac.in"));
         startActivity(intent);
      });
   }
}
```

Figure 5.11 Code for MainActivity.java file of the Intent.

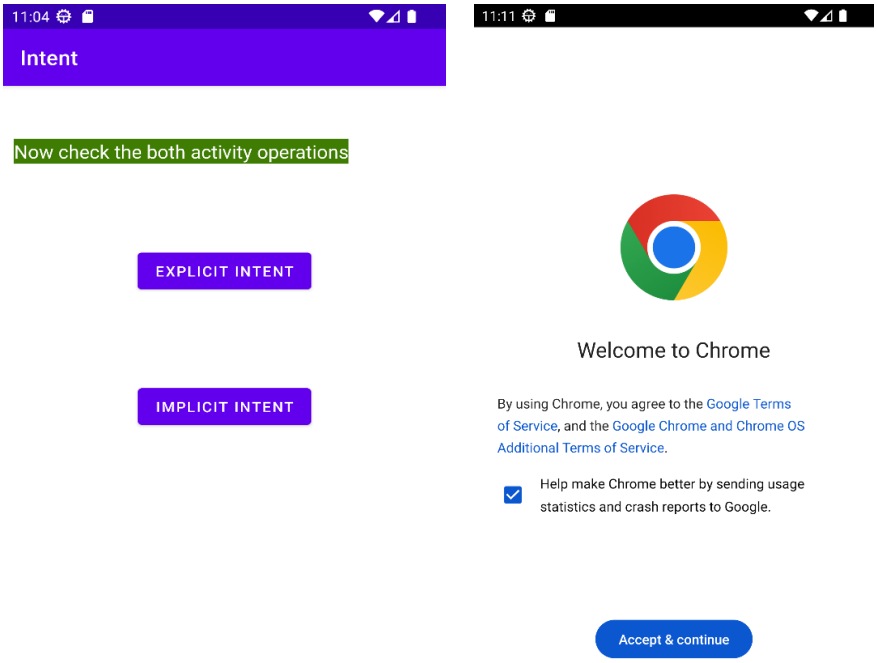


Figure 5.12 Screenshots of the Intent.

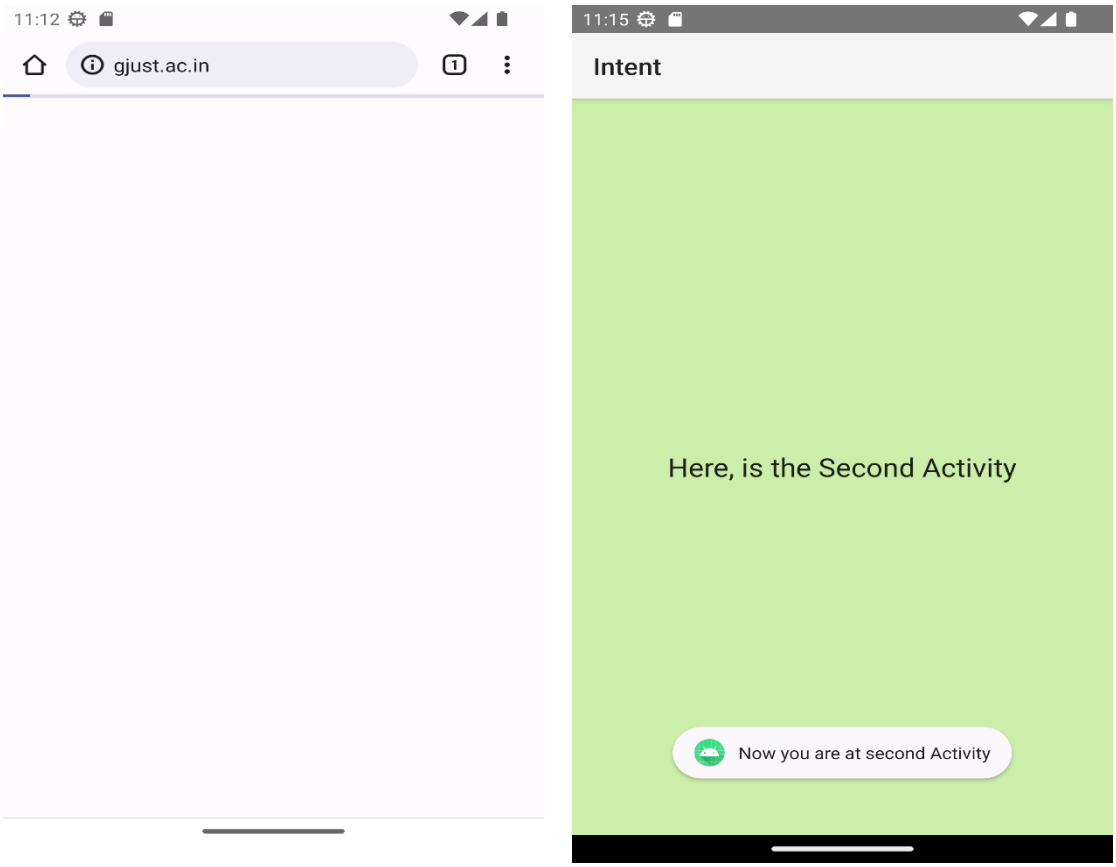


Figure 5.13 Screenshots of the Intent.

5.7. Fragment example

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
     <color name="black">#000</color>
    <color name="green">#0f0</color>
    <color name="white">#2A4227</color>
    <color name="button_background_color">#925</color>
</resources>
```

Figure 5.14 Code for color.xml file of the Fragment.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.abhiandroid.fragmentexample" >
   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/Theme.AppCompat.Light" >
      <activity
         android:name="gjust.dde.fragment.MainActivity"
         android:label="@string/app_name"
         android:exported="true">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   </application>
</manifest>
```

Figure 5.15 Code for AndroidManifest.xml file of the Fragment.

```
package gjust.dde.fragment;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.Toast;
public class FirstFragment extends Fragment {
   View view;
   Button firstButton;
   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
                    Bundle savedInstanceState) {
      view = inflater.inflate(R.layout.firstfragment, container, false);
      firstButton = (Button) view.findViewById(R.id.firstButton);
      firstButton.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {

            Toast.makeText(getActivity(), "1st Fragment",
Toast.LENGTH_LONG).show();
         }
      });
      return view;
   }
}
```

Figure 5.16 Code for FirstFragment.java file of the Fragment.

```
<LinearLayout
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:orientation="vertical"
    android:paddingBottom="10dp"
    android:paddingLeft="10dp"
    android:paddingRight="10dp"
    android:paddingTop="10dp"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/firstFragment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
       android:background="@color/design_default_color_on_primary"
        android:text="First Fragment"
        android:textColor="@color/white"
        android:textSize="20sp" />
    <Button
        android:id="@+id/secondFragment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
       android:background="@color/design_default_color_on_primary"
        android:text="Second Fragment"
        android:textColor="@color/white"
        android:textSize="20sp" />
    <FrameLayout
        android:id="@+id/frameLayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="10dp" />
</LinearLayout>
```

Figure 5.17 Code for activity_main.xml file of the Fragment.

```
package gjust.dde.fragment;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.app.Fragment;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.view.View;
import android.widget.Button;
public class MainActivity extends AppCompatActivity {
   Button firstFragment, secondFragment;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      firstFragment = (Button) findViewById(R.id.firstFragment);
      secondFragment = (Button) findViewById(R.id.secondFragment);
      firstFragment.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            loadFragment(new FirstFragment());
         }
      });
      secondFragment.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            loadFragment(new SecondFragment());
         }
      });
   }
   private void loadFragment(Fragment fragment) {
      FragmentManager fm = getFragmentManager();
      FragmentTransaction fragmentTransaction = fm.beginTransaction();
      fragmentTransaction.replace(R.id.frameLayout, fragment);
      fragmentTransaction.commit();
   }
}
```

Figure 5.18 Code of MainActivity.java file of the Fragment.

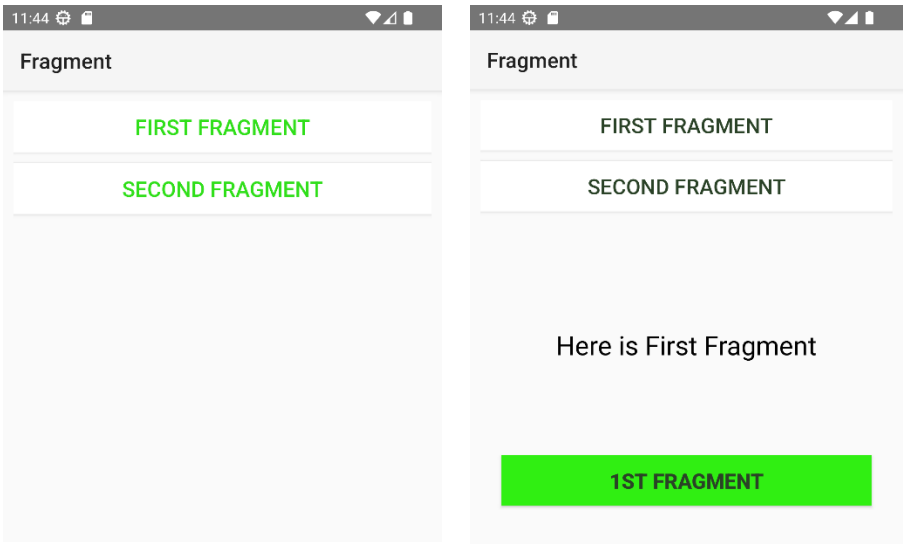


Figure 5.19 Screenshots of the Fragment App.

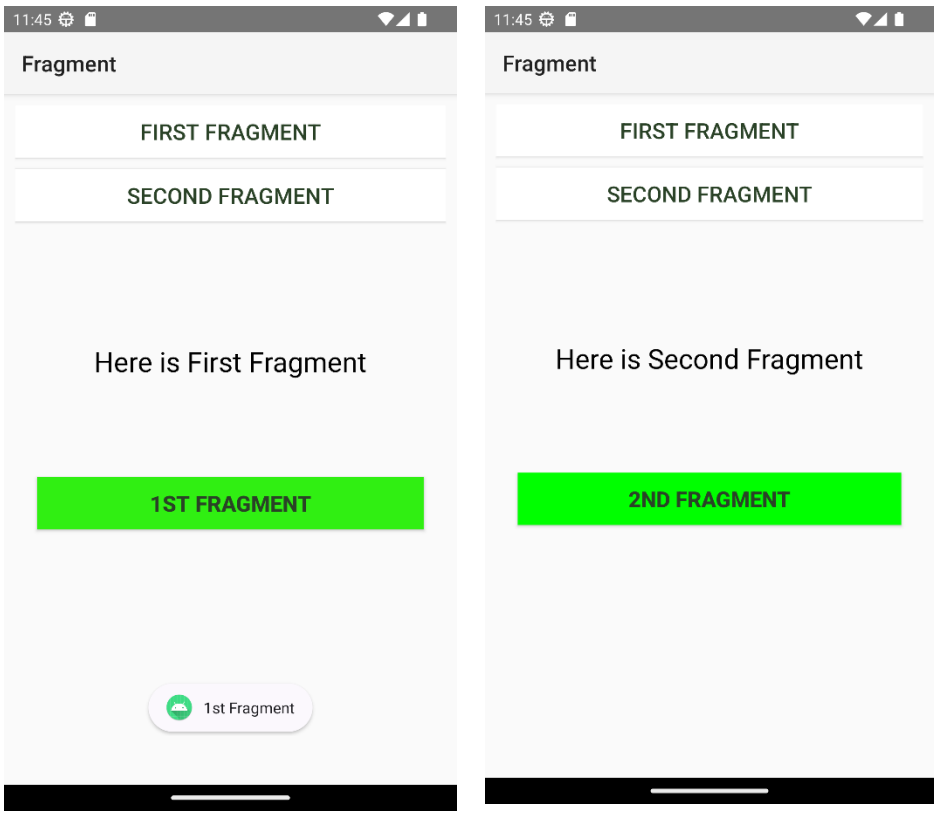


Figure 5.20 Screenshots of the Fragment App.

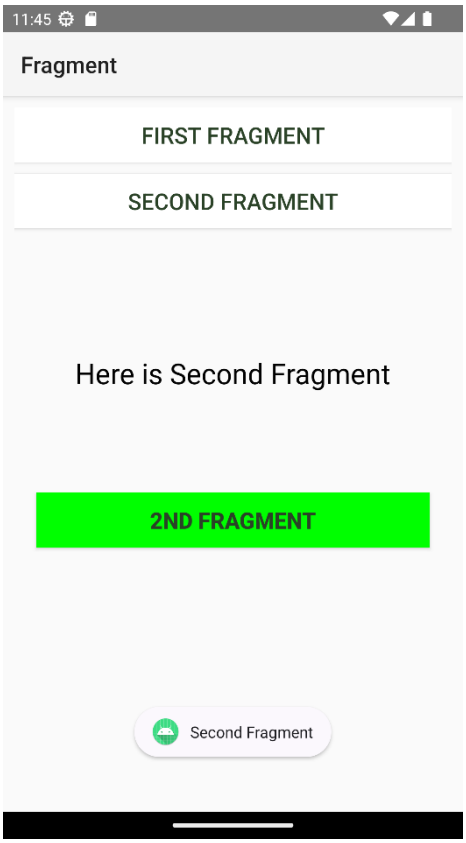


Figure 5.21 Screenshots of the Fragment result.

5.8. Summary

This chapter has covered all the activity related components. Activity lifecycle and example of the main activity discussed in the section 1.2 and 1.5. Next, the Intent features is discussed with their types and implementation shown by a simple example of implicit and explicit intent in section q.4 and 1.6. In last, the fragmentation of activity is explained and also explain the use of various classes of Fragment class. The simple application for understanding is explained in section 1.7. Overall, this chapter shows the functionally to handle the activity of the user visible screen and division of that and switching from one activity to another etc.

5.9. Check Your Progress

1. An activity can host one or more _____ at a time.
2. Activity is particularly focused on _____.

3. Activity has its own _____ in form of lifecycle.
4. Fragment is an _____ of an activity.
5. The main entry point for user interaction is _____.
6. Fragment is a collection of _____ and _____ classes.
7. Intent is used to switching from one _____ to another _____.
8. To start an intent, it calls a function _____.
9. _____ intent is sued to jump from one activity to other class.
10. _____ intent is used to communicate with other outside app.

5.10. Self-Assessment Questions

1. Define Activity. Explain its lifecycle with diagram.
2. Give a brief introduction about the activity, fragment and intent.
3. List the name of classes used to design a fragment application.
4. Explain Intent and its process of working.
5. What is fragmentation. Discuss the types of fragments.
6. What are the uses of explicit intent? explain in detail.
7. List the functions name of Activity class and explain.
8. Discuss with code segment of AndroidManifest.xml that must be updated for intent.

5.11. Answers (Section 5.9)

1. Fragments
2. Single screen
3. stages
4. sub-section
5. Activity
6. XML, Java/Kotlin
7. Activity, activity.
8. startActivity()
9. Explicit
10. Implicit.

5.12. References/ Suggested reading

1. Headfirst Android Development, Dawn Griffiths, 1st edition, .O′Reilly
2. Android Programming for Beginners, John Horton,2nd edition, . Packt
3. Android Programming with Kotlin for Beginners, John Horton, 1st edition, Packt.
4. Head-First Kotlin, Dawn Griffiths, 1st edition, O'Reilly.
5. Android App Development, Michael Burton, 3rd edition.

SUBJECT: MOBILE APPLICATION DEVELOPEMENT	
COURSE CODE: MCA-42	AUTHOR: Dr. Sunil Kumar Verma
LESSON NO. 6	
Android Menus	

STRUCTURE

LEARNING OBJECTIVE

To understand the concept of menu. You will learn how to implements different types of menus in android application. What is the scenario where you can apply particular menu types. Learn about different types of classes used for menu designing. All types of menus are explained through example.

Chapter 6. Menus

6.1. Introduction

Menu is a feature in any application to execute a command directly without going into details. For example, in basic application like Notepad have File, Edit, Format, etc. Menus are a common user

interface component in many types of applications. Menu API are used to provide a familiar and consistent user experience. Earlier there was a less requirement to provide menu option. But recent applications are based maximum on different types of menu items.

In android application three types of menus are used as name suggested below.

- Option Menu
- Context Menu
- Popup Menu

6.2. Options menu or App Bar

Menus are popular interface for any android application. To make an application interactive we should use menu. Option menu is a very profound feature of android application activity. It uses primary features of application for menu items for an activity window. It involved the useful actions (setting, search) that effect the application globally.

An example of global menu shown in figure 2.3. This type of menu helps in creating or combining multiple options and set actions against each menu.

Write code for the following files you can copy paste at relevant file and path.

- activity_main.xml
- main_menu.xml
- context_menu.xml
- MainActivity.kt (kt stand for Kotlin language)

Code for activity_main.xml

```
<menu 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"
    tools:context="com.example.optionmenu.MainActivity">
    <item  android:id="@+id/item1"
        android:title="File"/>
    <item  android:id="@+id/item2"
        android:title="Edit"/>
    <item  android:id="@+id/item3"
        android:title="Format"
        app:showAsAction="withText"/>
</menu>
```

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
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"
    tools:context="com.example.optionmenu.MainActivity"
        tools:ignore="MissingConstraints">
        <Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="@color/purple_200"/>
    <include layout="@layout/context_main" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Figure **Error! No text of specified style in document.**.1 Code Segment of Options menu for main_menu.xml.

Figure **Error! No text of specified style in document.**.2 Code of Options menu for activity_main.xml.

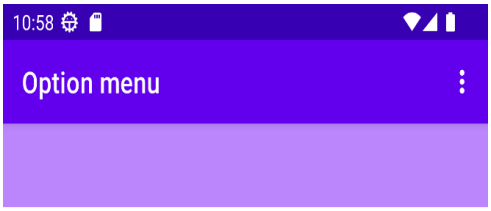
```
package com.example.optionmenu
import android.annotation.SuppressLint
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import android.view.Menu
import android.view.MenuItem
import android.widget.Toast
import android.widget.Toolbar
class MainActivity : AppCompatActivity() {
   protected override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      val toolbar = findViewById(R.id.toolbar) as Toolbar
      setSupportActionBar(toolbar)
   }
   private fun setSupportActionBar(toolbar: Toolbar) {   }
   @SuppressLint("ResourceType")
   override fun onCreateOptionsMenu(menu: Menu?): Boolean {
       getMenuInflater().inflate(R.layout.main_menu, menu)
      return true
   }
   override fun onOptionsItemSelected(item: MenuItem): Boolean {
      val id = item.itemId
      return when (id) {
         R.id.item1 -> { Toast.makeText(getApplicationContext(), "File Menu Selected",
Toast.LENGTH_LONG).show()
            true
         }
         R.id.item2 -> { Toast.makeText(getApplicationContext(), "Edit Menu Selected",
Toast.LENGTH_LONG).show()          true       }
         R.id.item3 -> {
            Toast.makeText(getApplicationContext(), "Format Menu Selected",
Toast.LENGTH_LONG).show()          true       }
         else -> super.onOptionsItemSelected(item)
      }
   }
}
```

Figure **Error! No text of specified style in document.**.3 Code for MainActivity.kt file of Options menu.

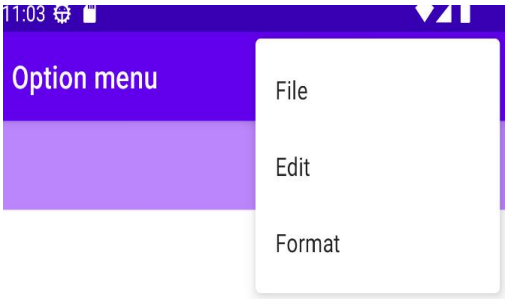
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.optionmenu.MainActivity">
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello Students"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

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

Figure **Error! No text of specified style in document.**.4 Code for context_menu.xml file of Options menu.



Hello Students



Hello Students

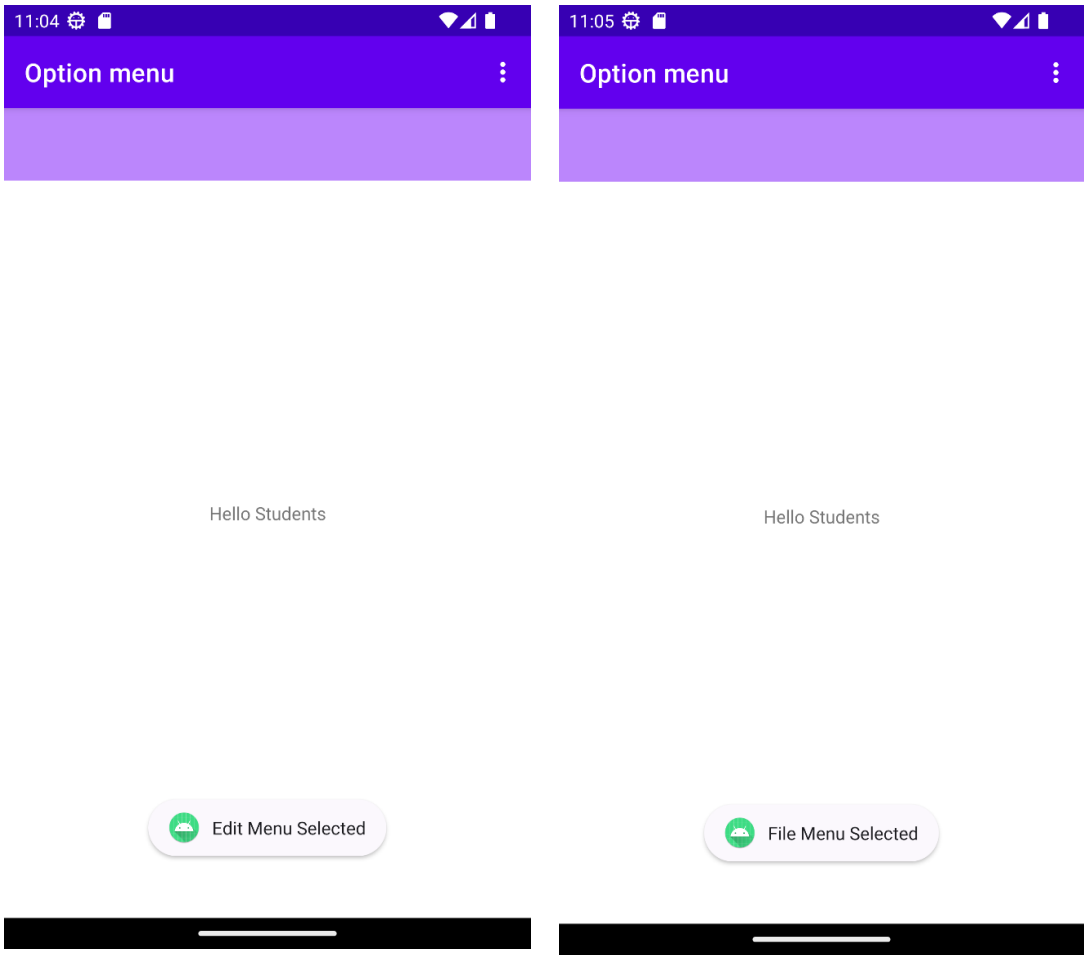


Figure **Error! No text of specified style in document.**.5 Result screenshots of Options menu.

6.3. Context Menu

A context menu is also very popular way to create menu in limited space with components itself. This is also known as floating menu which appears after a long pressing of right click on an element. It offers actions which are applicable on the selected elements or context frame. Context menu display a list of items which get affected in a bar form at the top of the mobile screen. User can also select multiple elements. See below code snippets.

```
<?xml version="1.0" encoding="utf-8"?>
<menu 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"
    tools:context="com.tutorialsbuzz.androidoptionmenu.MainActivity">
    <item
        android:id="@+id/mitem1"
        android:orderInCategory="100"
        android:title="@string/mitem1"
        app:showAsAction="never">
        <menu>
            <item
                android:id="@+id/subitem1"
                android:orderInCategory="100"
                android:title="@string/subitem1"
                app:showAsAction="never" />
            <item
                android:id="@+id/subitem2"
                android:orderInCategory="100"
                android:title="@string/subitem2"
                app:showAsAction="never" />
        </menu>
    </item>
    <item
        android:id="@+id/mitem2"
        android:icon="@mipmap/ic_launcher_round"
        android:orderInCategory="100"
        android:title="@string/mitem2"
        app:showAsAction="never" />
    <item
        android:id="@+id/mitem3"
        android:orderInCategory="100"
        android:title="@string/mitem3"
        app:showAsAction="never" />
    <item
        android:id="@+id/mitem4"
        android:orderInCategory="100"
        android:title="@string/mitem4"
        app:showAsAction="never" />
    <item
        android:id="@+id/mitem5"
        android:orderInCategory="100"
        android:title="@string/mitem5"
        app:showAsAction="never" />
</menu>
```

Figure **Error! No text of specified style in document.**.6 Code for context_menu.xml file of Context Menu.

```
<resources>
    <string name="app_name">ContextMenu</string>
    <string name="mitem1">File</string>
    <string name="subitem1">Open</string>
    <string name="subitem2">New</string>
    <string name="mitem2">Close</string>
    <string name="mitem3">CloseAll</string>
    <string name="mitem4">SaveAs</string>
    <string name="mitem5">Exit</string>
</resources>
```

Figure **Error! No text of specified style in document.**.7 Code for string.xml file of Context Menu.

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/androi
d"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
        <Button
            android:id="@+id/btn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="GJUST" />
    </LinearLayout>
```

Figure **Error! No text of specified style in document.**.8 Code for activity_main.xml file of Context Menu.

```
package gjust.dde.contextmenu
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.ContextMenu
import android.view.MenuItem
import android.view.View
import android.widget.Toast
import kotlinx.android.synthetic.main.activity_main.*
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        registerForContextMenu(btn)
    }
    override fun onCreateContextMenu(
        menu: ContextMenu?,
        v: View?,
        menuInfo: ContextMenu.ContextMenuInfo?) {
        super.onCreateContextMenu(menu, v, menuInfo)
        menu?.setHeaderTitle("Context Menu Header")
        menuInflater.inflate(R.menu.content_menu, menu)
    }
    override fun onContextItemSelected(item: MenuItem): Boolean {
        return when (item.itemId) {
R.id.mitem2 -> {
                Toast.makeText(this@MainActivity, "item2",
Toast.LENGTH_SHORT).show()
                true
            }
R.id.mitem3 -> {
Toast.makeText(this@MainActivity, "item3", Toast.LENGTH_SHORT).show()
                true            }
R.id.mitem4 -> {
Toast.makeText(this@MainActivity, "item4", Toast.LENGTH_SHORT).show()
                true            }
R.id.subitem1 -> {
Toast.makeText(this@MainActivity, "sub item1", Toast.LENGTH_SHORT).show()
                true            }
R.id.subitem2 -> {
 Toast.makeText(this@MainActivity, "sub item2", Toast.LENGTH_SHORT).show()
                true            }
            else -> super.onOptionsItemSelected(item)
        }
    }
}
```

Figure **Error! No text of specified style in document.**.9 Code for MainActivity.kt file of Context Menu.

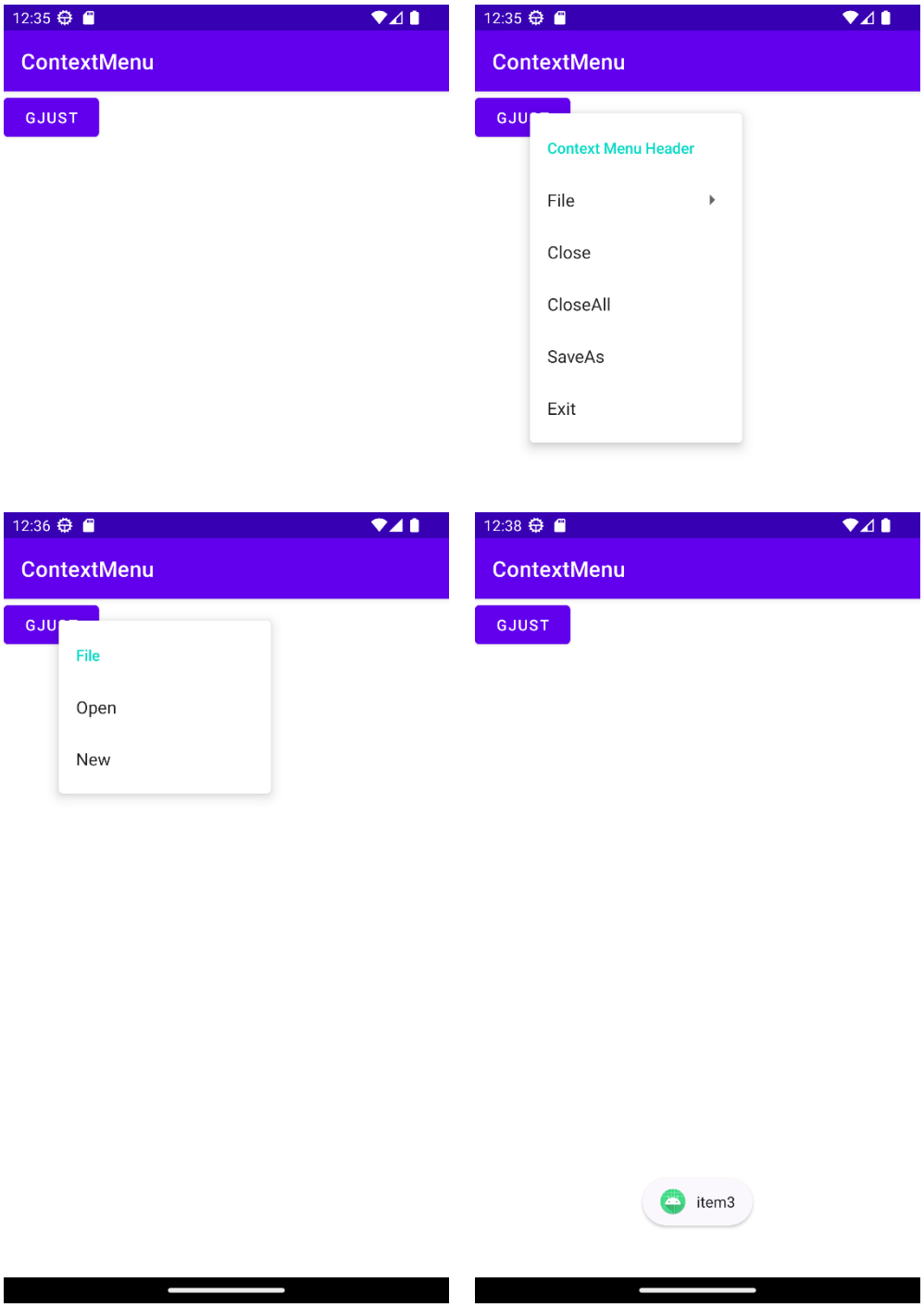


Figure **Error! No text of specified style in document.**.10 Screenshots of result for Context menu.

6.4. Popup menu

A menu shows a list of items in vertical order. It is applicable where a large list of actions is related to specific element and based on this next command will work. The action list of option menu should execute for extended action not for itself.

Setting Menu in XML

For all types of menus, there is a standard format in XML defined by android studio. There is need to write code in activity for menu items you can define all in xml file with menu attributes. This XML based menu resource file can be loaded in activity and fragment window.

XML based menu resource provides various benefits.

Easy to visualize the menu items in XML.

It provides the separate section to you from your application core coding area. It allows us to make alternative menu for other platforms having different screen sizes by using app resource file.

Pop menu help in displaying a list of items in vertical order to the view. It provides the number of actions which are useful and related to specific item. Below we are discussing the Popup Menu. A menu pops up when click on a component or item. Example to design a pop up on button click is shown. A Toast message set on click of pop menu option.

```
<?xml version="1.0" encoding="utf-8"?>
<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/btnShow"
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:text="Popup Menu"
        android:layout marginTop="200dp"
android:layout_marginLeft="100dp"/>
</LinearLayout>
```

Figure **Error! No text of specified style in document.**.11 Code for Popup Menu activity_main.xml.

```
package gjust.dde.popupmenu
import android.os.Bundle
import android.view.MenuItem
import android.view.View
import android.widget.Button
import android.widget.PopupMenu
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
class MainActivity : AppCompatActivity(), PopupMenu.OnMenuItemClickListener {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val btn = findViewById<View>(R.id.btnShow) as Button
        btn.setOnClickListener { v ->
            val popup = PopupMenu(this@MainActivity, v)
            popup.setOnMenuItemClickListener(this@MainActivity)
            popup.inflate(R.menu.popup_menu)
            popup.show()
        }
    }
    override fun onMenuItemClick(item: MenuItem): Boolean
 {
Toast.makeText(this, "Selected Item: " +item.title, Toast.LENGTH_SHORT).show()
        return when (item.itemId) {
            R.id.search -> true
            R.id.upload -> true
            R.id.copy -> true
            R.id.print -> true
            R.id.share -> true
            R.id.bookmark -> true
            else -> false
        }
    }
}
```

Figure **Error! No text of specified style in document.**.12 Code for the Popup Menu MainActivity.kt file.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@+id/search"
        android:title="Search file" />
    <item android:id="@+id/upload"
        android:title="Upload file" />
    <item android:id="@+id/copy"
        android:title="Copy file" />
    <item android:id="@+id/print"
        android:title="Print page" />
    <item android:id="@+id/share"
        android:title="Share link" />
    <item android:id="@+id/bookmark"
        android:title="Bookmark" />
</menu>
```

Figure **Error! No text of specified style in document.**.13 Code for popup_menu.xml file.

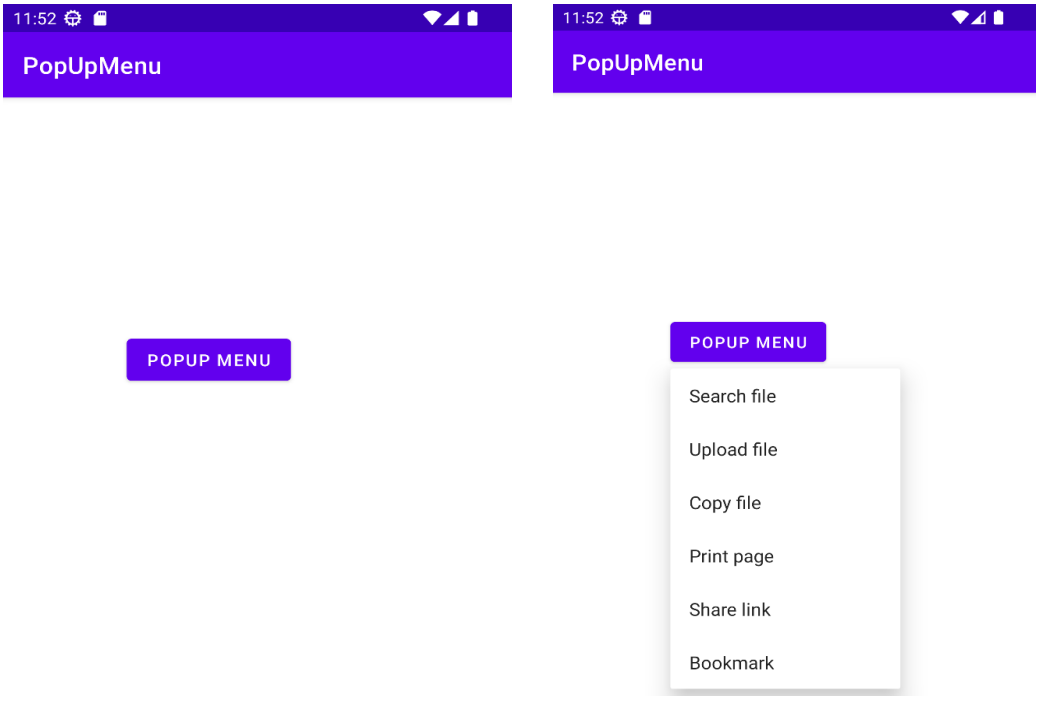


Figure **Error! No text of specified style in document.**.14 Screenshots of the Popup Menu result.

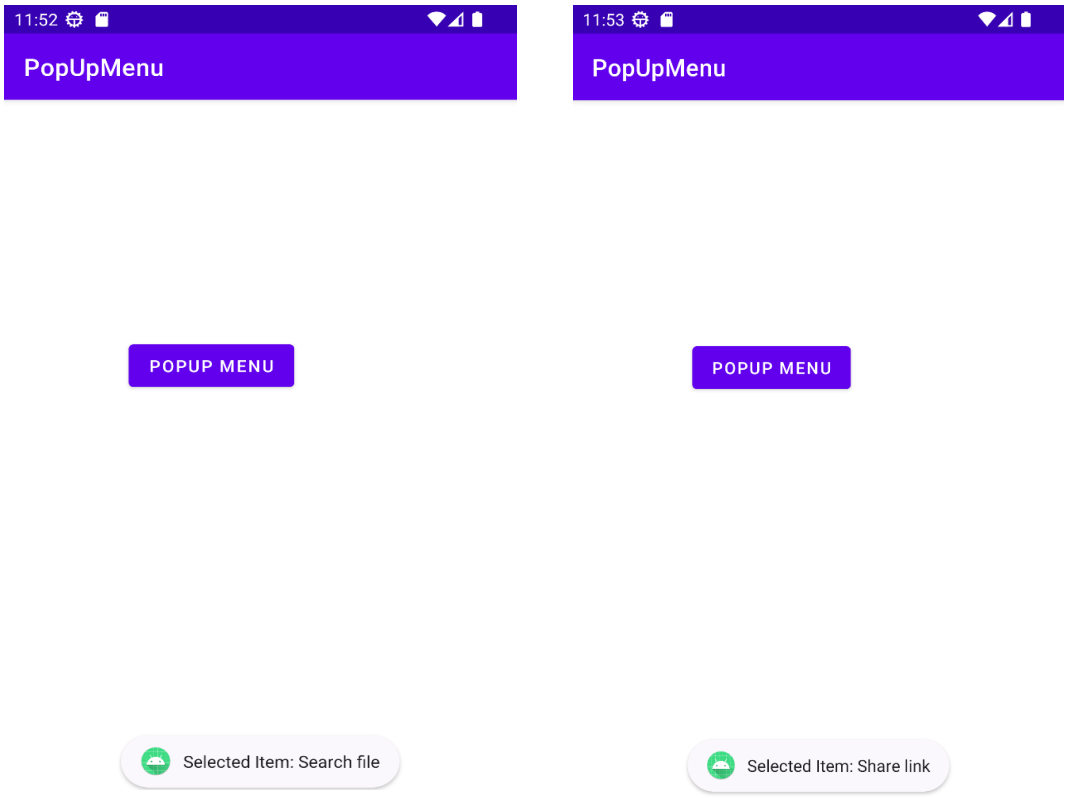


Figure **Error! No text of specified style in document.**.15 Screenshots of the Popup Menu result.

6.5. Summary

The chapter elaborates the use of android menu in android application design. Why the menu is important in app. The various types of menus and their use according to the screen space and shape. Each menu types are explained with example. As the section 1.2, 1.3 and 1.4 explained the basic as well as implementation of all these menus.

6.6. Keywords

Menu: this class is used to declared header or main menu in app.

MenuItem: this is a subclass which add sub menu under parent class.

Option menu: it is used to make easy use of app like adding search and profile option.

Popup Menu: it is used to perform some action on specific object or element.

Context menu: it is a type of menu which visible when you press a long right click.

API: it is a collection of classes provide some specific features,

6.7. Check Your Progress

1. _____ are used to execute a command directly without seeing details of an app.
2. Menu _____ are used to provide a consistent interface.
3. _____ menu is used to make application interactive for primary features.
4. Option menu is also known for _____ menu.
5. A package used for menu is _____.
6. Menu and MenuItem is a class of _____ package.
7. Content menu is popularly known for _____.
8. contact menu activated through a long press of _____.
9. User can select multiple items using _____ menu.
10. _____ used for a long list of operation for specific components.
11. Popup menu display item in _____ order.

6.8. Self-Assessment Questions

1. What is menu bar? Why is it important for android app.
2. Explain the different types of menus.
3. Discuss various packages and classes used to design an option menu.
4. Explain the list of function available in Popup menu class.
5. Write short note on the following option menu, popup menu and context menu.
6. What are the various functions of the Menu class.
7. Write a code snippet of menu and sub meu in XML file.
8. Explain the functions of Context menu class.

6.9. Answers (Section 6.7)

1. Menus
2. API
3. option
4. global
5. android.view.Menu

6. view
7. floating menu
8. right click.
9. context
10. popup.
11. vertical

6.10. References/ Suggested reading

1. Headfirst Android Development, Dawn Griffiths, 1st edition, .O′Reilly

2. Android Programming for Beginners, John Horton,2nd edition, Packt.

3. Head-First Kotlin, Dawn Griffiths, 1st edition, O'Reilly.

4. Android App Development, Michael Burton, 3rd edition.

SUBJECT: MOBILE APPLICATION DEVELOPEMENT	
COURSE CODE: MCA-42	AUTHOR: Dr. Sunil Kumar Verma
LESSON NO. 7	
Android Layout Manager	

STRUCTURE

LEARNING OBJECTIVE

The learning objective of an Android layout manager is to understand how to use the various layout managers available in Android to create a user interface that is both visually appealing and functional. This includes understanding the different types of layouts, such as LinearLayout, RelativeLayout, GridLayout and TableLayout, as well as how to use them to create a user interface that is optimized for different screen sizes and orientations. Additionally, the student should be able to use

the layout manager to create custom views and layouts, as well as how to use the layout manager to optimize performance.

Chapter 7. Menus

7.1. Introduction

Layout manager in android is a container that helps to design user interface and placed UI widgets in it. Layout is used to manage an activity screen to display various components. As we know that each application has visual representation generated from View and ViewGroup. Every android app is a collection different activity that may have its separate layout to manage various UI components. It is also considered an extension of ViewGroup class. Layout can be used in nested way or in combined form also.

Android SDK have a number of layout classes which can hold the UI components for Fragments, Views and Activities.

7.2. RelativeLayout

Relative Layout is an XML source file that helps to display a sub view in relative positions. Position can be decided by the relative child members of the sibling elements. You can set the element view as left-of, right-of or below the view. Another option is in positions (like align to bottom, left, center and top) relative to whole current Relative Layout area.

It is considered one of the high performance and powerful component for designing an interface in flat area instead of nest representation of view. When you designing app with multiple linear layouts then you can use one relative layout in replace of these.

A mobile screen views are arranged in an order as per the requirement of applications. Generally, application found with linear layout including features like weight and gravity even that the screen is not relatively arranged. Some advance attributes/properties are used for relative layout:

layout_alignParentLeft

layout_alignParentTop

layout_alignParentRight

layout_alignParentBottom

layout_centerVertical

layout_centerHorozontal

layout_above

layout_below

You can design buttons in activity_main.xml resource file available in res/layout folder. We have defined five TextView to display according to the relative location in screen.

Figure 7.1 Result of Relative Layout.

7.3. Table Layout

In Android, Table Layout is used to arrange the group of views into rows and columns. Table Layout containers do not display a border line for their columns, rows or cells. A Table will have as many columns as the row with the most cells.

Important points of Table Layout

For building a row in a table we will use the <TableRow> element. Table row objects are the child views of a table layout. Each row of the table has zero or more cells and each cell can hold only

one view object like ImageView, TextView or any other view. Total width of a table is defined by its parent container.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   xmlns:android="http://schemas.android.com/apk/res/android">
   <TextView
      android:id="@+id/text1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Top Left"
      android:textColor="@color/design_default_color_primary"
      android:textSize="25dp"
      android:layout_alignParentLeft="true"
      android:layout_alignParentTop="true"/>
   <TextView
      android:id="@+id/text2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Top Right"
      android:textColor="@color/teal_200"
      android:textSize="25dp"
      android:layout_alignParentTop="true"
      android:layout_alignParentRight="true"/>
   <TextView
      android:id="@+id/text3"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Bottom Left"
      android:textColor="@color/teal_700"
      android:textSize="25dp"
      android:layout_alignParentLeft="true"
      android:layout_alignParentBottom="true"/>
   <TextView
      android:id="@+id/text4"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Bottom Right"
      android:textColor="@color/purple_500"
      android:textSize="25dp"
      android:layout_alignParentRight="true"
      android:layout_alignParentBottom="true"/>
   <TextView
      android:id="@+id/text5"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello GJUian"
      android:textSize="25dp"
      android:textColor="@color/design_default_color_on_secondary"
      android:layout_centerVertical="true"
      android:layout_centerHorizontal="true"
      />
</RelativeLayout>
```

Figure 7.2 Code for activity_main.xml file of the Relative Layout.

Column can be both stretchable and shrinkable. If shrinkable then the width of column can be shrunk to fit the table into its parent object and if stretchable then it can expand in width to fit any extra space available.

We cannot specify the width of the children of the Table layout. Here, width always match parent width. However, the height attribute can be defined by a child; default value of height attribute is wrapping content.

An example for Table Layout is given below for more understanding.

```
package gjust.dde.tablelayout
import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
class MainActivity : AppCompatActivity() {
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
      val loginButton = findViewById<View>(R.id.loginBtn) as Button
            loginButton.setOnClickListener {
         Toast.makeText(applicationContext, "Hello Students..!!!",
Toast.LENGTH_LONG)
            .show()
      }
   }
}
```

Figure 7.3 Code for MainActivity.kt file of the Table Layout.

```
<resources>
    <string name="app_name">Table Layout</string>
    <string name="hello_world">DDE,GJUST</string>
        <string name="action_settings">Settings</string>
        <string name="loginForm">Login Form</string>
        <string name="userName">Username</string>
        <string name="password">Password</string>
        <string name="login">LogIn</string>
   </resources>
```

Figure 7.4 Code for string.xml file of the Table Layout.

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FFEB3B"
    android:orientation="vertical"
    android:stretchColumns="1">
    <TableRow android:padding="5dip">
        <TextView
            android:layout_height="wrap_content"
            android:layout_marginBottom="20dp"
            android:layout_span="2"
            android:gravity="center_horizontal"
            android:text="@string/loginForm"
            android:textColor="@color/black"
            android:textSize="25sp"
            android:textStyle="bold" />
    </TableRow>
    <TableRow>
        <TextView
            android:layout_height="wrap_content"
            android:layout_column="0"
            android:layout_marginLeft="10dp"
            android:text="@string/userName"
            android:textColor="@color/black"
            android:textSize="16sp" />
        <EditText
            android:id="@+id/userName"
            android:layout_height="wrap_content"
            android:layout_column="1"
            android:layout_marginLeft="10dp"
            android:background="#fff"
            android:hint="@string/userName"
            android:padding="5dp"
            android:textColor="#000" />
    </TableRow>
    <TableRow>
        <TextView
            android:layout_height="wrap_content"
            android:layout_column="0"
            android:layout_marginLeft="10dp"
            android:layout_marginTop="20dp"
            android:text="@string/password"
            android:textColor="@color/black"
            android:textSize="16sp" />
        <EditText
            android:id="@+id/password"
            android:layout_height="wrap_content"
            android:layout_column="1"
            android:layout_marginLeft="10dp"
            android:layout_marginTop="20dp"
            android:background="#fff"
            android:hint="@string/password"
            android:padding="5dp"
            android:textColor="#000" />
    </TableRow>
    <TableRow android:layout_marginTop="20dp">
        <Button
            android:id="@+id/loginBtn"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_span="2"
            android:background="@color/purple_200"
            android:text="@string/login"
            android:textColor="#000"
            android:textSize="20sp"
            android:textStyle="bold" />
    </TableRow>
</TableLayout>
```

Figure 7.5 Code for activity_main.xml file of the Table Layout.

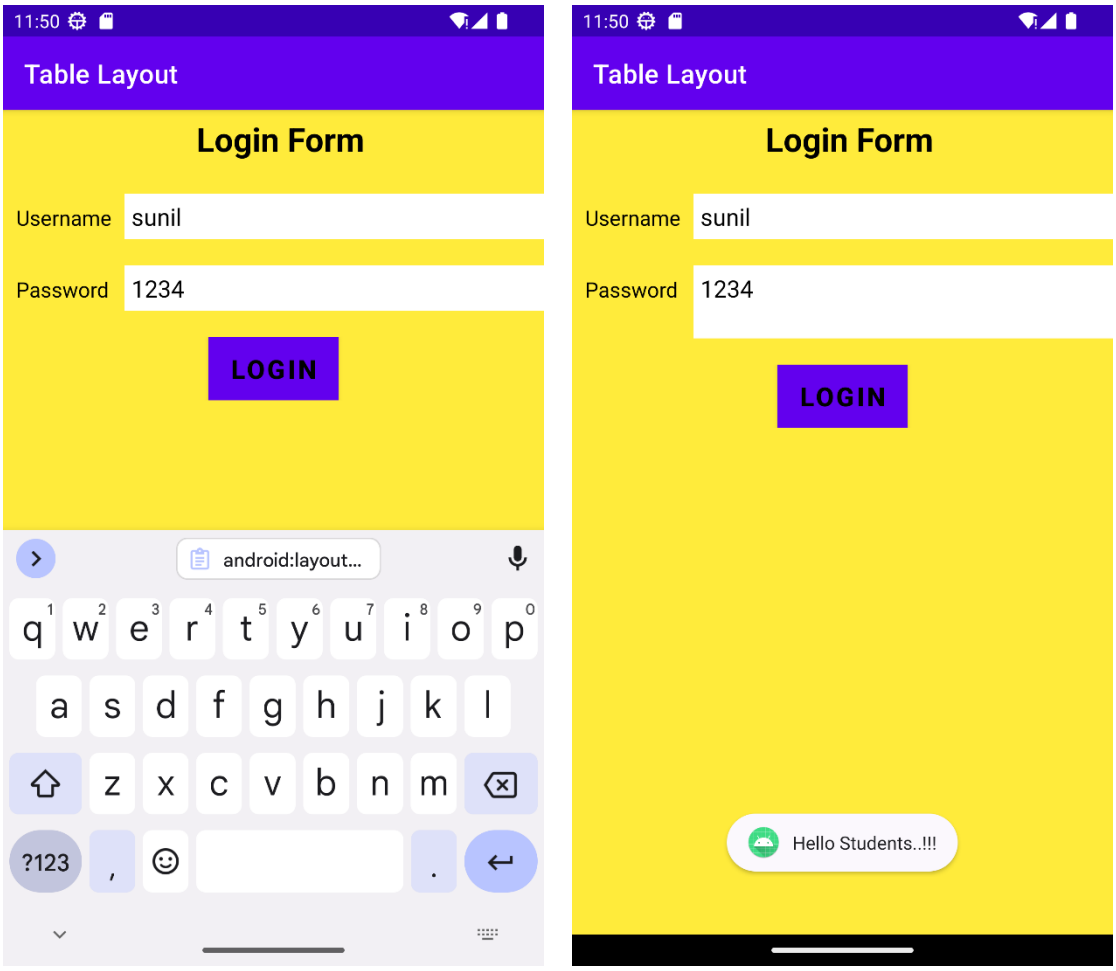


Figure 7.6 Screenshots of the Table Layout app.

7.4. Grid Layout

Grid layout is a collection of rectangular shapes that is like spreadsheet. it sperate the viewing area in different cells/parts. A layout is the continuous cells which is configured by rowSpec and columnSpec parameters. These spec is used to set number of row and column. GridLayout cells don't overlap

Space among components may be specified by various margins like left, top, right and bottom parameters. But you may use useDefaultMargins for automatic margins. GridLayout helps to arrange the views in a grid style. It is also known as layout manager.

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
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:rowCount="4"
   android:columnCount="2"
   android:background="#3F51B5"
   tools:context=".MainActivity">
   <ImageView
      android:layout_height="90dp"
      android:layout_width="130dp"
      android:layout_margin="10dp"
      android:src="@drawable/a"
      android:layout_gravity="center_horizontal"  />
   <ImageView
      android:layout_height="90dp"
      android:layout_width="200dp"
      android:layout_margin="10dp"
      android:src="@drawable/b"
      android:layout_gravity="center_horizontal" />
   <ImageView
      android:layout_height="90dp"
      android:layout_width="130dp"
      android:layout_margin="10dp"
      android:src="@drawable/c"
      android:layout_gravity="center_horizontal" />
```

Figure 7.7 Code for activity_main.xml of the Grid Layout.

```
<ImageView
    android:layout_height="90dp"
    android:layout_width="200dp"
    android:layout_margin="10dp"
    android:src="@drawable/d"
  android:layout_gravity="center_horizontal" />
  <ImageView
    android:layout_height="90dp"
    android:layout_width="130dp"
    android:layout_margin="10dp"
    android:src="@drawable/e"
    android:layout_gravity="center_horizontal" />
  <ImageView
    android:layout_height="90dp"
    android:layout_width="220dp"
    android:layout_margin="10dp"
    android:src="@drawable/f"
    android:layout_gravity="center_horizontal" />
  <ImageView
    android:layout_height="90dp"
    android:layout_width="130dp"
    android:layout_margin="10dp"
    android:src="@drawable/g"
    android:layout_gravity="center_horizontal"/>
  <ImageView
    android:layout_height="90dp"
    android:layout_width="200dp"
    android:layout_margin="10dp"
    android:src="@drawable/h"
    android:layout_gravity="center_horizontal" />
  <ImageView
    android:layout_height="90dp"
    android:layout_width="130dp"
    android:layout_margin="10dp"
    android:src="@drawable/i"
    android:layout_gravity="center_horizontal" />
  <ImageView
    android:layout_height="90dp"
    android:layout_width="130dp"
    android:layout_margin="10dp"
    android:layout_marginLeft="40dp"
    android:src="@drawable/j"
    android:layout_gravity="center_horizontal"  />
</GridLayout>
```

Figure 7.8 Code for activity_main.xml file of the Grid Layout.

7.5. Linear Layout

Linear layout is used for designing layout of android application. All the components displayed in a linear style. All sub components of linear layout are displayed in specified orientation. The values of orientation can be in vertical or horizontal.

Linear Layout Orientation

Orientation of components can be arranged in two ways:

- Vertical
- Horizontal

As the name indicate that the sub components are arrange in a line either horizontally or vertically.

Let's discuss each types individually in details

Vertical

In this the subitems are organized in a vertically order in a line consequently.

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
```

Figure 7.9 Code snippet for linear vertical orientation.

Horizontal

in this the subitems are organized in a horizontal order in a line consequently.

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="horizontal" >
```

Figure 7.10 Code snippet for linear horizontal orientation.

The linear Layout is used as a viewing component where all objects/children are found in single order i.e vertically or horizontally. to specify the vertical or horizontal an orientation attribute is used in xml file. Gravity (right, left, center) attribute is set for margin between components.

Layout Weight attribute is used to assign importance to view the object space. It is applied to take maximum remaining space for visualization and by default it is zero. Equal distribution is set for assigning same space to each component. layout height and width attributes are set to "0dp" for vertical and horizontal respectively. Unequal distribution is set to assign different space for components.

Attributes of Linear Layout

android:id: to uniquely recognizes the layout.

android:baselineAligned: to handle the alignment according to the children baselines.

android:divider: is used to set divider for vertical among components.

android:gravity: it specify the position of object according to X and Y axis.

android:orientation: it specify the direction of components (by default horizontal)

android:weightSum: calculates the child weight.

```
<resources>
    <string name="app_name">LinearLayout</string>
    <string name="to">To:</string>
    <string name="subject">Subject:</string>
    <string name="message">Message:</string>
    <string name="send">Send</string>

</resources>
```

Figure 7.11 Code for sring.xml file of the Linear Layout.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingLeft="16dp"
   android:paddingRight="16dp"
   android:orientation="vertical" >
   <EditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="@string/to" />
   <EditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="@string/subject" />
   <EditText
      android:layout_width="match_parent"
      android:layout_height="97dp"
      android:layout_weight="1"
      android:gravity="top"
      android:hint="@string/message" />
   <Button
      android:layout_width="90dp"
      android:layout_height="wrap_content"
      android:layout_gravity="right"
      android:text="@string/send" />
</LinearLayout>
```

Figure 7.12 Code for the activity_main.xml file of the Linear Layout.

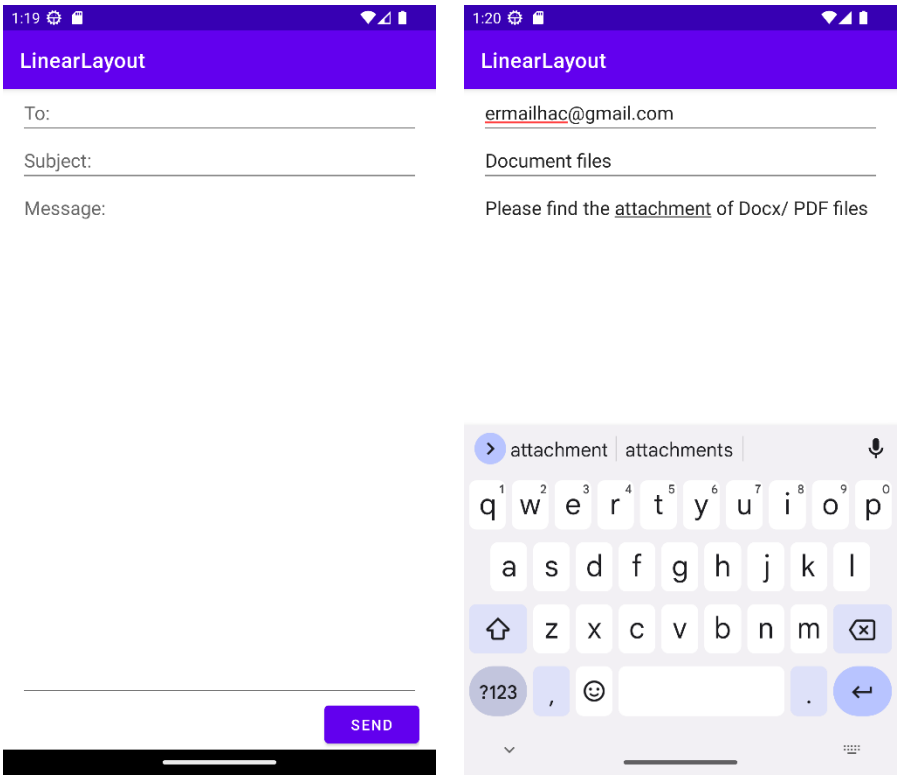


Figure 7.13 Results of Linear Layout.

7.6. Summary

In this chapter, we have discussed the Android Layout Manager which is a system used to arrange the user interface elements in an Android application. It is responsible for positioning and sizing the views within the application's window. The layout manager is responsible for determining how the views are laid out on the screen, as well as how they interact with each other. It is also responsible for responding to user input and making sure the views are updated accordingly. This chapter has covered different layout: Relative layout, Grid Layout, Table Layout, Linear layout.

7.7. Keywords

Linear layout: is a type of layout used in Android development that arranges elements in a single row or column.

Grid layout: is a type of design system used to arrange elements on a page.

RelativeLayout: is a type of layout used in Android development that allows developers to position elements relative to each other.

Orientation: It is used to set the direction of components.

7.8. Check Your Progress

1. Relative Layout is an _____ source file.
2. Linear layout includes features like _____ and _____.
3. Table Layout is used to arrange the group of views into _____ and _____.
4. Each cell can hold only _____ object.
5. Column can be both stretchable and _____.
6. We cannot specify the _____ of the children of the Table layout.
7. Grid layout is a collection of _____ shapes.
8. Orientation of components can be arranged in _____ ways.
9. Unequal distribution is set to assign different _____ for components.

7.9. Self-Assessment Test

1. Explain relative layout in detail.
2. What is orientation? Also explain which layout uses orientation for layout.
3. Define table layout with example.
4. Write about various classes of android layout manager.
5. Write the list of attributes used design Linear layout.
6. List out the various layout manager and explain each with important class.
7. What is grid layout? where is it applicable in android app.
8. Write properties used to design relative layout with explanation.
9. How can you use a combined layout for designing an app.

7.10. Answers (Section 7.8)

1. XML
2. weight, gravity
3. rows, columns
4. one view
5. shrinkable
6. width

7. rectangular
8. two
9. space

7.11. References/ Suggested reading

1. Headfirst Android Development, Dawn Griffiths, 1st edition, .O′Reilly

2. Android Programming for Beginners, John Horton,2nd edition, .Packt

3. Android Programming with Kotlin for Beginners, John Horton, 1st edition, Packt.

4. Head-First Kotlin, Dawn Griffiths, 1st edition, O'Reilly.

5. Android App Development, Michael Burton, 3rd edition.

SUBJECT: MOBILE APPLICATION DEVELOPEMENT

COURSE CODE: MCA-42	AUTHOR: Dr. Sunil Kumar Verma
LESSON NO. 8	
Android Data Adapters	

STRUCTURE

LEARNING OBJECTIVE

To understand the working of adapter. To learn the types of data adapter. How to bind adapter with different data sources and also get update about data bridge, UI components and data sources.

Chapter 8. Data Adapter

8.1. Introduction

This chapter will discuss about the data adapter implementation. In order to connect data from any data source, like as an ArrayList, HashMap, SQLite, etc., with a user interface element, such as a ListView or GridView, we must use an adapter. Fundamentally, an adapter serves as a link between data sources and the UI component.

8.2. Array Adapter

The most used adapter in Android is ArrayAdapter. Use the ArrayAdapter when you have a list of single-type items that are kept in an array. The same applies if you have a list of names, cities, or phone numbers. TextView appears in the layout can have data from the ArrayAdapter.

The Adapter is a way of communication between the UI interface and Source of data. It helps to retrieve data from data source and make the items viewable that can be visible to the UI Component. Data Source can be Arrays, HashMap, Database, etc. and UI Components can be ListView, GridView, Spinner, etc.

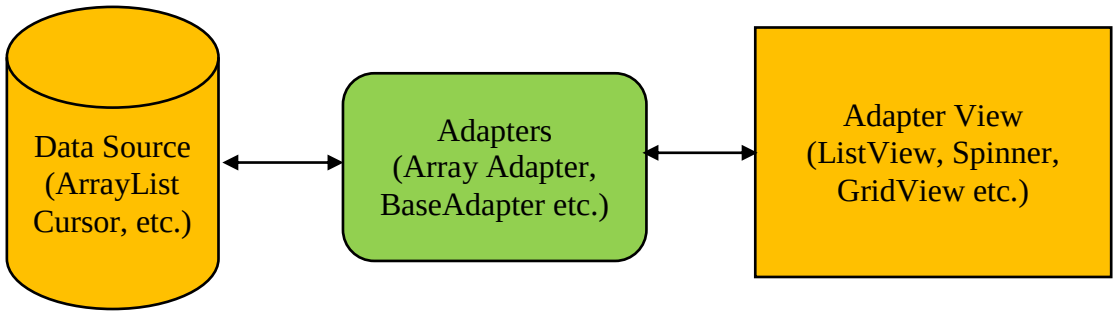


Figure 8.1 Block Diagram of Adapter.

ArrayAdapter is the most commonly used adapter in android. When you have a list of single type items which are stored in an array you can use ArrayAdapter. Likewise, if you have a list of phone numbers, names, or cities. ArrayAdapter has a layout with a single TextView. If you want to have a more complex layout instead of ArrayAdapter use custom ArrayAdapter. The basic syntax for ArrayAdapter is given as:

Context: the current context. This value cannot be null.

Resource: The resource ID for the layout file containing a layout to use when instantiating views.

textViewResourceId: The id of the TextView within the layout resource to be populated.

Objects: The objects to represent in the ListView. This value cannot be null.

Context is used to pass the reference of the current class. Here 'this' keyword is used to pass the current class reference. Instead of 'this' we could also use the getApplicationContext() method which is used for the Activity and the getApplication() method which is used for the Fragments.

public ArrayAdapter(this, int resource, int textViewResourceId, T[] objects)

resource: It is used to set the layout file(.xml files) for the list items.

public ArrayAdapter(this, R.layout.itemListView, int textViewResourceId, T[] objects)

textViewResourceId: It is used to set the TextView where you want to display the text data.

public ArrayAdapter(this, R.layout.itemListView, R.id.itemTextView, T[] objects)

objects: These are the array object which is used to set the array element into the TextView.

```
String courseList[] = {"Python", "C Programming", "Data Structure", "Database",
  "Java", " Compiler Design", "Android Development", "Operating System"};
ArrayAdapter arrayAdapter;
arrayAdapter = new ArrayAdapter(this, R.layout.itemLV, R.id.itemTView, courseList[]);
```

8.3. Example of Array Adapter

In this example, the list of courses is displayed using a simple array adapter. Note that we are going to implement this project using the Java language.

Step 1: Create a New Project

To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio. Note that select Java as the programming language.

Step 2: Working with the activity_main.xml file

Go to the layout folder and in activity_main.xml file change the ConstraintLayout to RelativeLayout and insert a ListView with id simpleListView. Below is the code for the activity_main.xml file.

```
<?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"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <ListView
        android:id="@+id/simpleListView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:divider="#000"
        android:dividerHeight="2dp"/>
</RelativeLayout>
```

Figure 8.2 Code for activity_main.xml file of the ArrayAdapter.

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

    <TextView
        android:id="@+id/textView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:padding="12dp"
        android:textColor="#000" />
</LinearLayout>
```

Figure 8.3 Code for list_item.xml file of the ArrayAdapter.

```
package gjust.dde.arrayadapter
import android.os.Bundle
import android.view.View
import android.widget.ArrayAdapter
import android.widget.ListView
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
  var simpleList: ListView? = null
  var courseList = arrayOf(
    "C", "Kotlin", "Java", "JavaScript", "Python",
    "C++", "Matlab", "JSON" )
  override fun onCreate(savedInstanceState: Bundle?)
 {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    simpleList = findViewById<View>(R.id.simpleListView) as ListView
    val arrayAdapter = ArrayAdapter(this, R.layout.list_item,
                                    R.id.textView, courseList)
    simpleList!!.adapter = arrayAdapter
  }
}
```

Figure 8.4 Code for MainActivity.kt file of the Array Adapter.

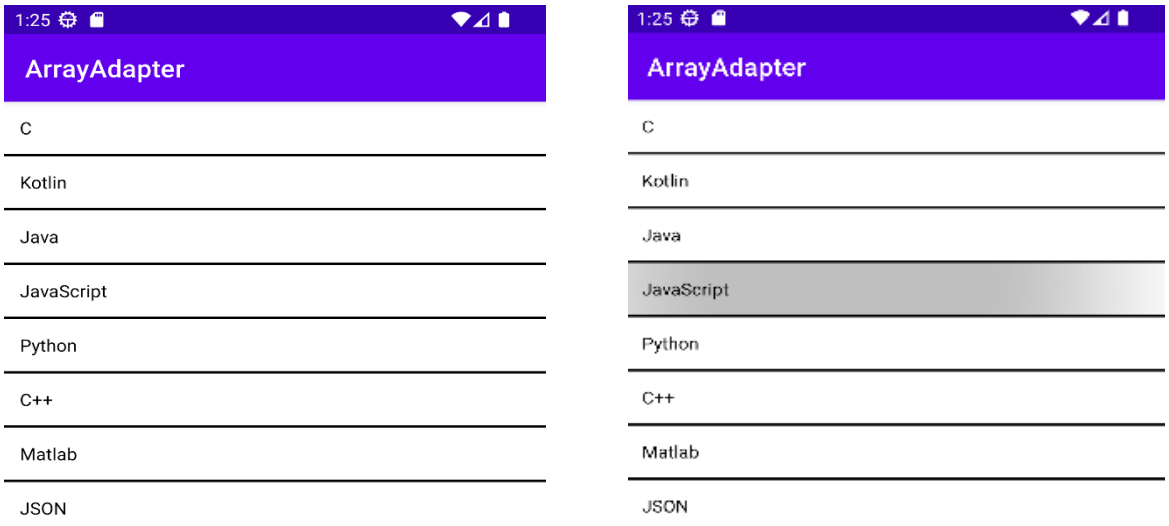


Figure 8.5 Screenshots of ArrayAdapter results.

8.4. Basic ArrayList

ArrayList is a dynamic data structure in which you can add or remove any number of elements and those elements are stored in ordered sequence. It may also contain duplicate values.

ArrayList are the implementations of List interface. The package you required to import the ArrayList is import java.util.ArrayList; In ArrayList, the items to be added are only of object type. No primitive data types (i.e., int, char etc) are allowed to insert in ArrayList.

Syntax of ArrayList:

It's very easy to use ArrayList, below is the syntax to declare it:

ArrayList arrayList = new ArrayList();

String ArrayList Declaration:

ArrayList<String> arrayList = new ArrayList<String>();

By using this syntax, an ArrayList is created which you can use to store characters. Here arrayList is the object name of this particular ArrayList.

8.5. ArrayList Adapter

ArrayList is a dynamic type advance data structure. It gives flexibility to add or remove any number of elements. It stored data/elements in sequential way and may have duplicate values. ArrayList are implemented with the List interface This class need the package to import is import java.util.ArrayList.

Only objects type of data are handled by this class. It doesn't support primitive data like int, float, char etc.

Syntax for ArrayList is:

ArrayList al=new ArrayLProg();

For string type of ArrayList

ArrayList<String> al=new ArrayLProg<String>();

AbstractList a base class for ArrayList and bother work under interface list. ArrayList is a subclass of AbstractList class and it implements List Interface. It has various methods that are defined and inherited from its parent class. Below is the list of those methods with example:

boolean add(Object o):

This function is sued to add element of object type which may be user defined. It adds an element of Specific Object type at the end of ArrayList as no index is mentioned in the method. It returns True if element is successfully added, and returns false if it is not.

void clear():

This method removes all the elements of the ArrayList. Below the example program clear() method shows the working of this method:

Object clone():

Clone function is used to make the same copy of an existing ArrayList. This method returns the exact same copy of the ArrayList object.

boolean contains(Object item):

This function is used to check whether the elements is available or not in ArrayList and return true/false as result.

As you know that we can design ArrayList data structure with existing data types such as Integer, Character, String, etc. However, user can also define their own object to hold data into ArrayList.

Few key points that should remember when you are working with this ArrayList:

It is a dynamic array means re-sizable that can shrink or grow at runtime.

Element can be deleted from any index.

It can work with any data object.

It uses List Iterator for traversing of elements in forward and backward.

```
<resources>
    <string
name="app_name">ArrayList2</string>
    <string name="name">Name</string>
    <string name="button">Add</string>
</resources>
```

Figure 8.6 Code for string.xml file of ArrayList Adapter.

It can store duplicate and null elements.

Most important when length of input objects are not confirmed then use this dynamic array.

```
<?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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <EditText
        android:id="@+id/Name"
        android:layout_width="266dp"
        android:layout_height="49dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:autofillHints=""
        android:ems="10"
        android:hint="Name"
        android:inputType="textPersonName"
        android:text="@string/name"
        app:layout_constraintEnd_toStartOf="@+id/button2"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/button2"
        android:layout_width="100dp"
        android:layout_height="48dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:text="@string/button"
 android:textAppearance="@android:style/TextAppearance.Material.Body1"
        android:visibility="visible"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <ListView
       android:id="@+id/ListofNames"
        android:layout_width="409dp"
        android:layout_height="0dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/Name" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Figure 8.7 Code for activity_main.xml file of ArrayList Adapter.

```
package gjust.dde.arraylist2;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
public class MainActivity extends AppCompatActivity {
Button btn;
EditText text;
ListView lv;
List<String> friends=new ArrayList<String>();
String[] elements={"Sunil","Verma","Ashok","Anshu","Arush","Rahul"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btn=findViewById(R.id.button2);
        text=findViewById(R.id.Name);
        lv=findViewById(R.id.ListofNames);
friends=new ArrayList<String>(Arrays.asList(elements));
        ArrayAdapter AD=new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1,friends);
  lv.setAdapter(AD);
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
String strname=text.getText().toString();
friends.add(strname);
        Collections.sort(friends);
AD.notifyDataSetChanged();
    }
});
lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
 public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
  Toast.makeText(MainActivity.this, "Pos="+position+" Name= "+
friends.get(position),Toast.LENGTH_SHORT).show();
    }
});
    }
}
```

Figure 8.8 Code for MainActivity.java file of ArrayList Adapter.

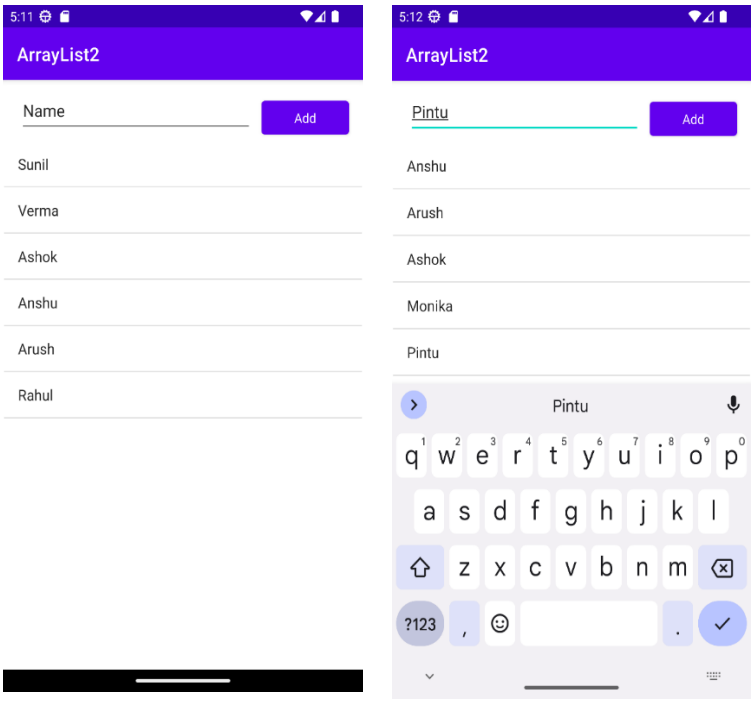


Figure 8.9 Screenshots to adding elements in ArrayList.

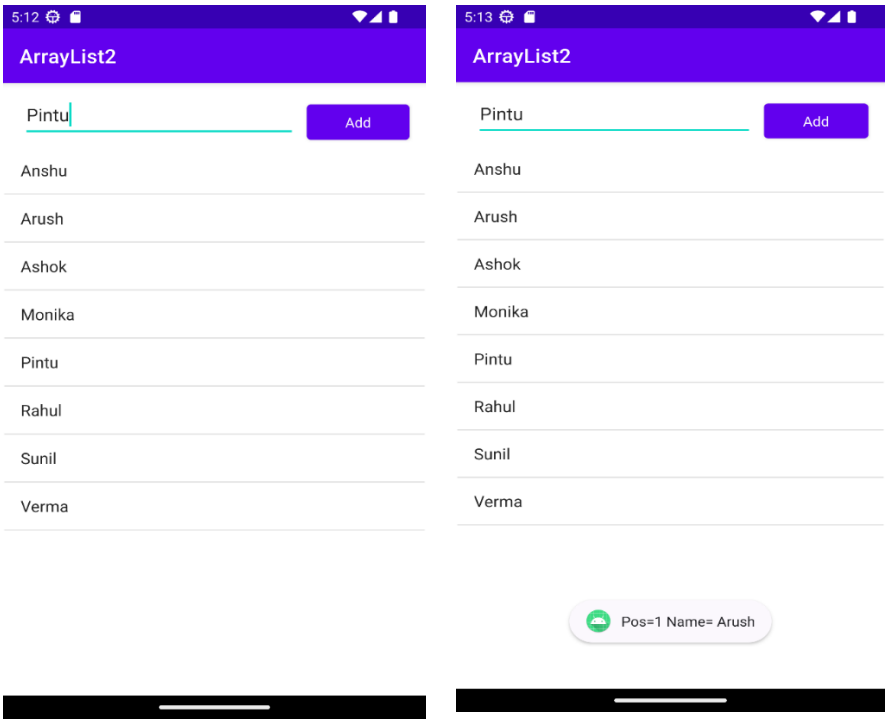


Figure 8.10 Screenshots of Sorting, Toast message of the ArrayList.

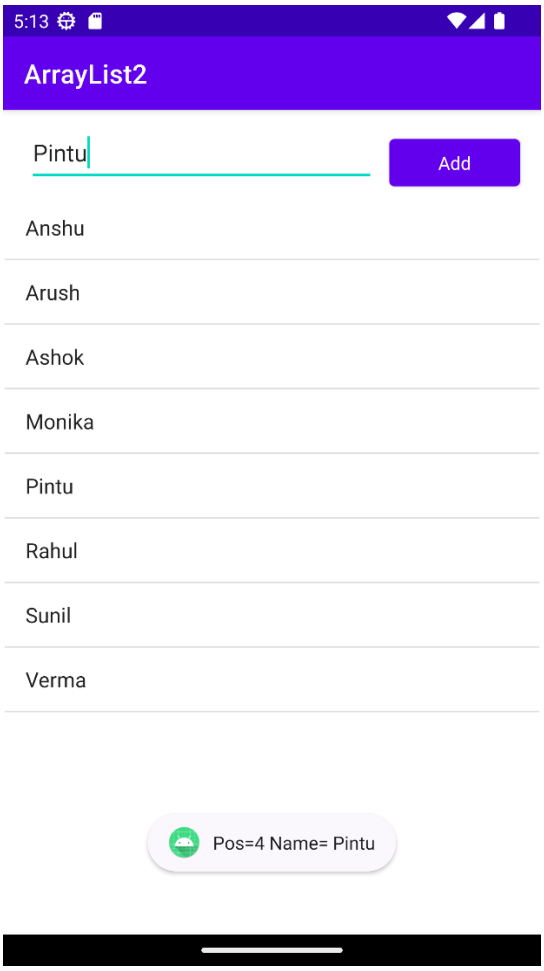


Figure 8.11 Screenshots of index and value of ArrayList.

8.6. Basic of BaseAdapter

In android programming, ArrayList is used to display elements in vertical way using scrollable collection. As we have read about ArrayAdapter, which is the simplest form of data filling and displayed with ListView using Adapter. The same thing can be processed through a collection which more advance as compare to normal Array Adapter. But both are using ListView to display on android screen. The data is populated through Adapter that is also a thread based notified and working as an intermediate between data objects unit and visualize unit. So, adapter, convert the objects into view that transfer to container like ListView. When we are working Adapter, always keep remember it is a bridge between Data Source and UI that fill data to UI modules.

What type of data you are using that decide the use of data source i.e ArrayAdapter. Example of few other data sources are CursorAdapter that working like a result set for embedded SQLite (a database type).

8.7. Base Adapter

As described in ArrayAdapter that it is a data holder and send to adapter view which later used by the different data viewers like as GridView, ListView, Spinner etc. Now, if you want to customize these existing viewing components then use base adapter.

BaseAdapter is a generic class that is existing at the top of specific adapter classes and it can directly work with ListView. Sometimes it is also common base class for general implantation of Adapter classes. For the customization of data in ListView and GridView we can design a new adapter class which will extends from Base Adapter class. By default, every existing adapter classes is implementing the base adapter class.

For example, this class is extending the BaseAdapter, where base class function must be override to make custom changes. This code snippet only explains two most common functions

```
public class CustomAdapter extends BaseAdapter {
@Override
public int getCount()
{
int count=arrayList.size();
return count;
}
@Override
public View getView(int i, View view, ViewGroup viewgroup)
{
view = inflter.inflate(R.layout.activity_gridview, null);
ImageView iconview = (ImageView) view.findViewById(R.id.iconview);
iconview.setImageResource(flags[i]);
return view;
}
```

The below code for creating a new project to show the custom adapter using BaseAdapter.

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

        <GridView
            android:id="@+id/simpleGridView"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:footerDividersEnabled="false"
            android:numColumns="3" />
    </LinearLayout>
```

Figure 8.12 Code for activity_main.xml of the BaseAdapter.

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

    <ImageView
        android:id="@+id/icon"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:scaleType="fitXY"
        android:layout_margin="5dp"
        android:layout_gravity="center_horizontal" />
</LinearLayout>
```

Figure 8.13 Code for activity_gridview.xml file of the BaseAdapter.

```
package gjust.dde.baseadapter;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
public class CustomAdapter extends BaseAdapter
{
        Context context;
        int animals[];
        LayoutInflater inflter;
        public CustomAdapter(Context applicationContext, int[] animals)
        {
            this.context = applicationContext;
            this.animals = animals;
            inflter = (LayoutInflater.from(applicationContext));
        }
        @Override
        public int getCount() {
            return animals.length;
        }
        @Override
        public Object getItem(int i) {
            return null;
        }
        @Override
        public long getItemId(int i) {
            return 0;
        }
        @Override
        public View getView(int i, View view, ViewGroup viewGroup) {
            view = inflter.inflate(R.layout.activity_gridview, null);
            ImageView icon = (ImageView) view.findViewById(R.id.icon);
            icon.setImageResource(animals[i]);
            return view;
        }
}
```

Figure 8.14 Code for CusomeAdapter.java file of BaseAdapter

```
package gjust.dde.baseadapter;
import android.app.Activity;
import android.os.Bundle;
import android.widget.GridView;

public class MainActivity extends Activity {

GridView simpleGrid;
int animals[] = {R.drawable.sparrow1, R.drawable.sparrow2,
R.drawable.sparrow3, R.drawable.sparrow4, R.drawable.sparrow5,
            R.drawable.sparrow6, R.drawable.sparrow7, R.drawable.sparrow8,
R.drawable.sparrow9,

R.drawable.sparrow10,R.drawable.sparrow11,R.drawable.sparrow12,R.drawable.s
parrow13,R.drawable.sparrow14,

R.drawable.sparrow15,R.drawable.sparrow16,R.drawable.sparrow17,R.drawable.s
parrow18,R.drawable.sparrow19,R.drawable.sparrow20};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        simpleGrid = (GridView) findViewById(R.id.simpleGridView);
        CustomAdapter customAdapter = new
CustomAdapter(getApplicationContext(), animals);
        simpleGrid.setAdapter(customAdapter);
    }
}
```

Figure 8.15 Code for MainActivity.java file of the BaseAdapter.

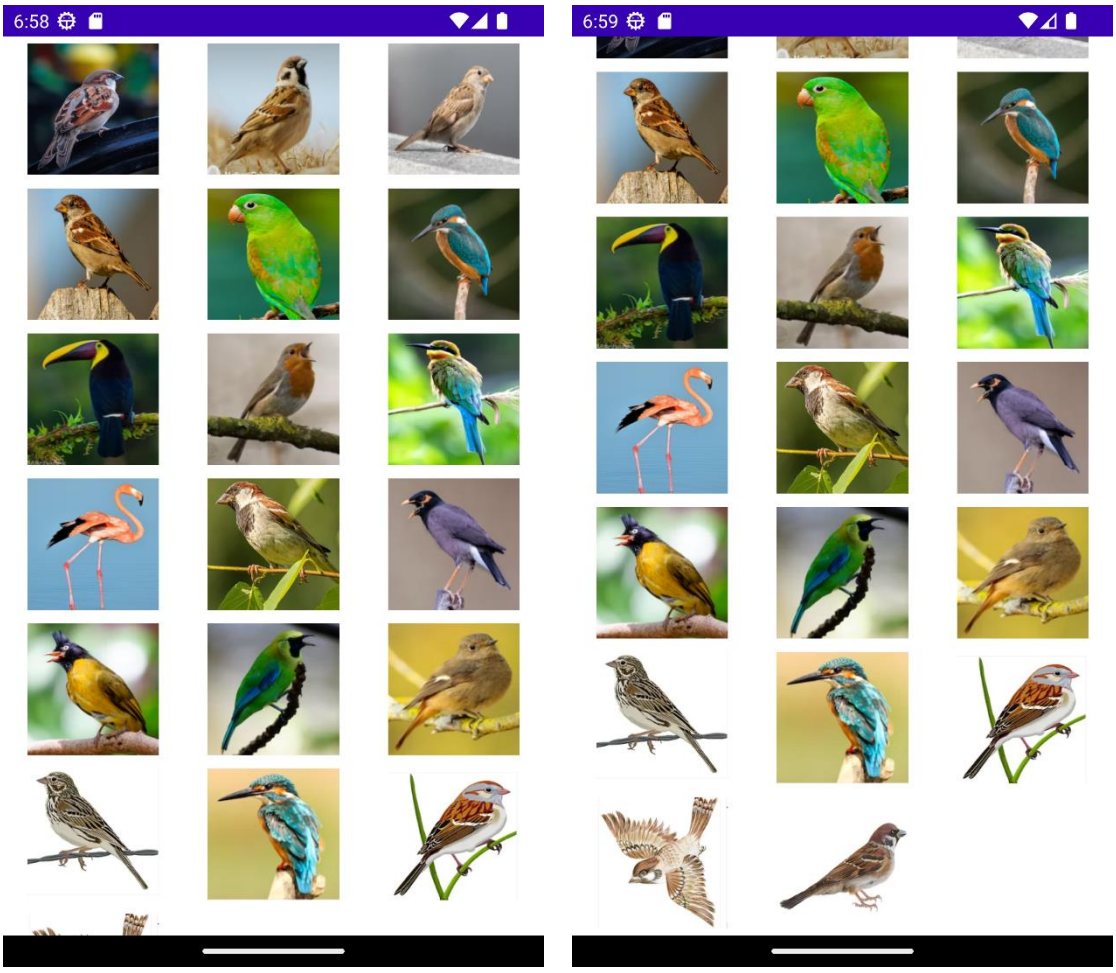


Figure 8.16 Screenshots of GridView using BaseAdapter.

8.8. Summary

The Adapter class is a fundamental component of the Android framework. It allows for the creation of custom user interfaces that are tailored to an app's specific needs. Adapters bridge the gap between data sources, such as databases or arrays, and views, like ListViews or Spinners, allowing developers to create complex UI elements without having to write a lot of code.

An adapter typically contains three main components: an array list containing data items; a layout file defining how each item should be displayed in its view; and an onCreateViewHolder() method which binds the two together. Adapters can also contain additional methods for manipulating their underlying data set and responding to user interaction with individual views within it.

8.9. Keywords

AdapterView: It is an Android class that serves as a base for classes such as ListView, GridView, and Spinner. It provides the ability to bind data from sources such as arrays or databases

ArrayAdapter: It is a type of adapter used in Android to provide views for an Adapter View, such as ListView or GridView, by populating the data from an array.

BaseAdapter: An abstract class that serves as the base for implementing a custom adapter.

ListAdapter: An interface in the Android SDK that provides a bridge between an Adapter View and the underlying data for that view.

RecyclerView: It is a class that serves as a bridge between the data to be displayed in the user interface and the UI components. This also support event handling for clicks and swipes.

8.10. Check Your Progress

1. A list of single type elements is data holder into _____.
2. _____ is used to display data of the DataAdapter
3. _____ is a bridge between UI interface and data source.
4. Array, HashMap and database are the example of _____.
5. ListView, GridView and Spinner are the _____.
6. _____is used to pass the reference of the current class.
7. ArrayList is the implementation of _____.
8. ArrayList is a _____ type data structure.
9. To traverse the elements of list _____ is used.
10. Base adapter is using a _____ based implementation.
11. BaseAdapter is a _____ class above the specific adapter classes.

8.11. Self-Assessment Test

1. What is Adapter? How android application can use it?
2. Explain all the functions of Adapter class.

3. Explain the different types of adapters used in android app designing.
4. What are the data viewing components? Explain.
5. Compare the BaseAdapter and ArrayAdapter class.
6. Write about the adapter generic class.
7. Discuss base adapter class. and its important functions.
8. Define ArrayList. Explain its uses with example.

8.12. Answers (Section 8.10)

1. Array
2. TextView
3. Adapter
4. Data Source
5. UI components
6. List Interface
7. dynamic
8. Iterator
9. thread
10. generic

8.13. References/ Suggested reading

1. Headfirst Android Development, Dawn Griffiths, 1st edition, .O′Reilly
2. Android Programming for Beginners, John Horton,2nd edition, .Packt
3. Android Programming with Kotlin for Beginners, John Horton, 1st edition, Packt.

SUBJECT: MOBILE APPLICATION DEVELOPEMENT

COURSE CODE: MCA-42	AUTHOR: Dr. Sunil Kumar Verma
LESSON NO. 9	
Android Content Views	

STRUCTURE

LEARNING OBJECTIVE

In this chapter, you will learn how to use special components for viewing the contents on the android screen. There is a list of various components that will be discussed in this chapter: GridView, WebView, ScrollView, SearchView, TabHost, Dynamic ListView, Expanded ListView.

Chapter 9. Content Views

9.1. Introduction

A View is a straightforward component of a user interface. Advance viewing components help to display a long list of items in android main activity. The data can be in any form to display. Each viewing components classes having a number function to interact with users. It is in charge of drawing and event handling.

9.2. GridView

GridView is a data viewing component that uses two-dimensional scrolling grid. Grid is combination of rows and columns. The item that you want add may differently numbers of object which is not necessary in advance the List Adapter or BaseAdapter automatically arrange the layout of grid for data.

It is also working like a bridge between UI component and source of data. Adapter can be used to supply the data to the spinner, list view, grid view etc. Adapter View have two subclasses ListView and GridView. GridView have lots of attribute to set on Main Activity. The spacing of grid can be in form of any unit likes: px, in, dp, sp, etc.

The BaseAdapter topic covered all the grid related concepts, you can take reading of this in Section 8.6 and 8.7, Chapter 8.

9.3. WebView

When you are working with huge own content then you can use any previously discussed components or collections. But when you want to directly access the data of web page in your application then it will not display correctly. The content may be distorted when you try to display. The solution of this problem is to use WebView component that helps in displaying a web page view in mobile view. As we use Chrome and other browsers which are specially design to view the web contents. The similar thing is achieved through WebView at some extent.

WebView allows to displays objects as apart of android activity layout. Sometime WebView is used to have more control on application with some advance configuration with some security parameters. A list of methods supported by this class are:

- **canGoBack():** to return back to the item history.
- **canGoForward():** to move forward based on item history.
- **clearHistory():** It clear both backward and forward history.
- **destroy():** It destroy the state of web page.
- **findAllAsync(String search):** To search the given string and highlight it .
- **getProgress():** to get the progress of loading page.
- **getTitle():** it return the title of web page.
- **getUrl():** return URL of active page.

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

android:usesCleartextTraffic="true"

```
<?xml version="1.0" encoding="utf-8"?>
<manifest android:targetSandboxVersion="1"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.INTERNET" />
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:usesCleartextTraffic="true"
        android:supportsRtl="true"
        android:theme="@style/Theme.WebView"
        tools:targetApi="31">

        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
        </activity>
    </application>

</manifest>
```

Figure 9.1 Code for AndroidManifest.xml of the WebView.

```
<?xml version="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:paddingLeft="10dp"
    android:paddingRight="10dp"
    android:paddingTop="10dp"
    android:paddingBottom="10dp" tools:context=".MainActivity">
    <TextView android:text="WebView" android:layout width="wrap content"
        android:layout_height="wrap_content"
        android:id="@+id/textview"
        android:textSize="35dp"
        android:layout alignParentTop="true"
        android:layout_centerHorizontal="true" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Dept. of Distance"
        android:id="@+id/textView"
        android:layout below="@+id/textview"
        android:layout_centerHorizontal="true"
        android:textColor="#ff7aff24"
        android:textSize="35dp" />
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/editText"
        android:hint="Enter Text"
        android:focusable="true"
        android:textColorHighlight="#ff7eff15"
        android:textColorHint="#ffff25e6"
        android:layout_marginTop="46dp"
        android:layout_below="@+id/imageView"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout alignRight="@+id/imageView"
        android:layout_alignEnd="@+id/imageView" />
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView"
        android:src="@drawable/gju"
        android:layout_below="@+id/textView"
        android:layout_centerHorizontal="true" />
    <Button
        android:layout_width="wrap_content"
        android:layout height="wrap content"
        android:text="Go"
        android:id="@+id/button"
        android:layout_alignTop="@+id/editText"
        android:layout_toRightOf="@+id/imageView"
        android:layout_toEndOf="@+id/imageView" />
    <WebView
        android:layout width="wrap content"
        android:layout_height="wrap_content"
        android:id="@+id/webView"
        android:layout_below="@+id/button"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true" />

</RelativeLayout>
```

Figure 9.2 Code for activity_main.xml file of WebView.

```
package gjust.dde.webview;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.Button;
import android.widget.EditText;


public class MainActivity extends Activity  {
    Button button1;
    EditText edittext1;

    private WebView webv1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        button1=(Button)findViewById(R.id.button);
        edittext1=(EditText)findViewById(R.id.editText);

        webv1=(WebView)findViewById(R.id.webView);
        webv1.setWebViewClient(new MyBrowser());

        button1.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                String url = edittext1.getText().toString();

                webv1.getSettings().setLoadsImagesAutomatically(true);
                webv1.getSettings().setJavaScriptEnabled(true);
                webv1.setScrollBarStyle(View.SCROLLBARS_INSIDE_OVERLAY);
                webv1.loadUrl(url);
            }
        });
    }
```

Figure 9.3 Code of MainActivity.java of the WebView.

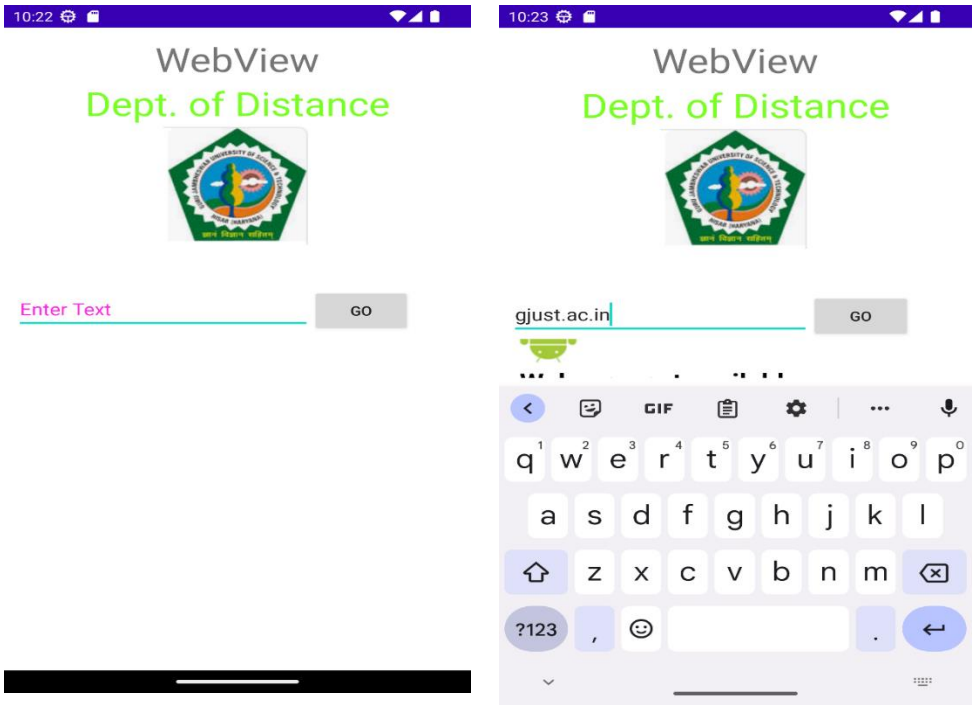


Figure 9.4 Screenshots of WebView Application.

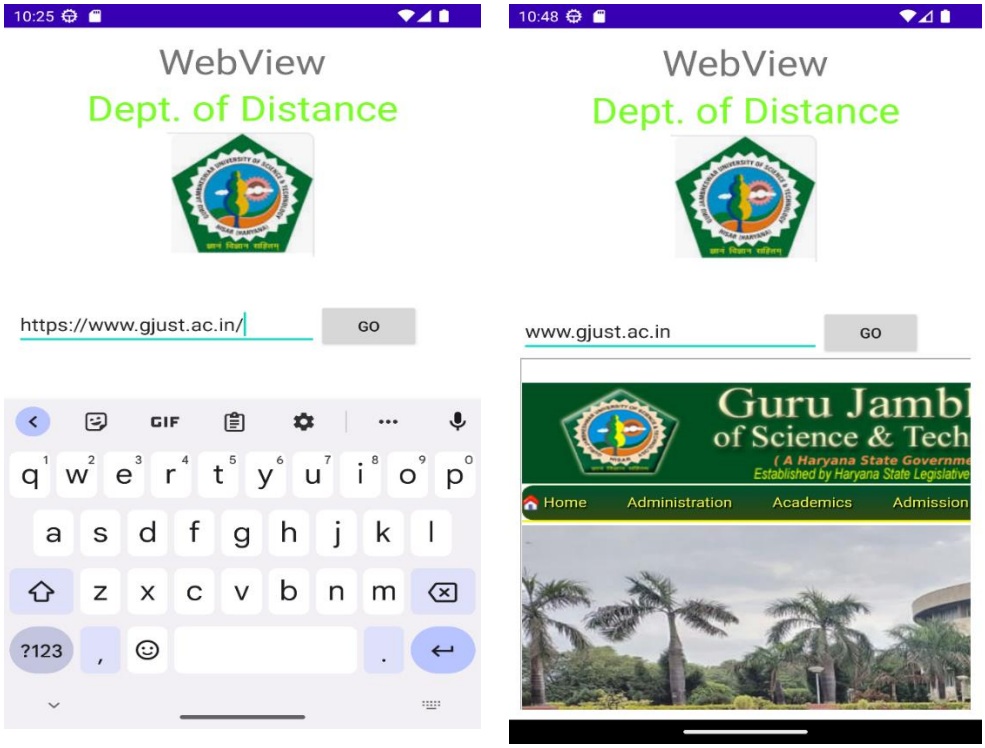


Figure 9.5 Screenshots of WebView for **www.gjust.co.in** web page.

9.4. ScrollView

A class inherited from android.widget.ScrollView is used to scroll the contents in application, if text side is larger. It only scrolls the current activity area where elements are available in form of text to UI components. By default, vertical scroll view supported by android and horizontal can be adjust manually.

A number of elements can be place in code to check the scrolling view. So, in example we have used buttons list. One important point to keep remember while using scroll view, only one direct child is support. For multiple types of components, you can use other types of layouts like Table, Relative, and Linear layout. We can specify individual layout_width and layout_height to setting on screen or use density pixel or other unit for hard structure then place them in standard layout and bind with the ScrollView to make it scrollable. Now, see the below android application based on ScrollView class components.

```
package gjust.dde.scrollview;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

Figure 9.6 Code for MainActivity.java of the ScrollView.

```
<?xml version="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="10dp"
    android:paddingLeft="10dp"
    android:paddingRight="10dp"
    android:paddingTop="10dp"
    tools:context=".MainActivity">
<TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:text="Scroll View for Buttons"
            android:id="@+id/textView"
            android:layout_gravity="center_horizontal"
            android:layout_centerHorizontal="true"
            android:layout_alignParentTop="true" />
        <ScrollView android:layout_marginTop="30dp"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:id="@+id/scrollView">
            <LinearLayout
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:orientation="vertical" >
        <Button
               android:layout_width="fill_parent"
               android:layout_height="wrap_content"
               android:text="One"/>
Repeat this button tags 20 time or more to see the scrolling and just
change the button text from one to twenty
  </LinearLayout>
        </ScrollView>
</RelativeLayout>
```

Figure 9.7 Code for activity_main.xml file of ScrollView.

Figure 9.8 Screenshots of ScrollView App.

9.5. SearchView

SearchView is a special widget used to set search interface where an app user can search query and submit their request for searching. Query is based on some suggestions.

XML tags for SearchView:

<SearchView

android:id="@+id/searchview"

android:layout_width="wrap_content"

android:layout_height="wrap_content" />

The SearchView class have a list of functions. few of methods are discussed below:

getQuery(): It is used to take data from the search view and return a value of CharSequence type.

A code segment for collecting request after initializing SearchView object and put into the request object of CharSequence

SearchView searchview = (SearchView) findViewById(R.id.searchview);

CharSequence request = searchview.getQuery();

getQueryHint(): It collect the text for hint from the text of SearchView and return value of CharSequence .

SearchView searchview = (SearchView) findViewById(R.id.searchview);

CharSequence hint = searchview.getQueryHint();

isIconfiedByDefault(): this get the default iconified state of text from SearchView and returns true or false.

boolean iconfiedstate=searchview.isIconfiedByDefault();

simpleSearchView.setIconifiedByDefault(false): to keep text visible set it false.

setOnQueryTextFocusChangeListener(OnFocusChangeListenerlistener): This function listen the every event of changing in text field. Apply this n SearchView object

The attributes of ScrollView are available in main_activity.xml file.

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <SearchView
        android:id="@+id/search"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:iconifiedByDefault="false">
        <requestFocus />
    </SearchView>
    <ListView
        android:id="@+id/listview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_below="@+id/search"
        />
</RelativeLayout>
```

Figure 9.9 Code for activity_main.xml file of SearchView.

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dp">

    <TextView
        android:id="@+id/nameLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Animal : " />

    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@+id/nameLabel" />

</RelativeLayout>
```

Figure 9.10 Code for listview_items.xml file of the SearchView.

```
package gjust.dde.searchview;

public class AnimalNames {
    private String animalName;

    public AnimalNames(String animalName) {
        this.animalName = animalName;
    }

    public String getAnimalName() {
        return this.animalName;
    }

}
```

Figure 9.11 Code for AnimalsName.java of the SearchView.

```
package gjust.dde.searchview;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
public class ListViewAdapter extends BaseAdapter {
    Context mContext;
    LayoutInflater inflater;
    private List<AnimalNames> animalNamesList = null;
    private ArrayList<AnimalNames> arraylist;
    public ListViewAdapter(MainActivity context, List<AnimalNames>
animalNamesList) {
        mContext = context;
        this.animalNamesList = animalNamesList;
        inflater = LayoutInflater.from(mContext);
        this.arraylist = new ArrayList<AnimalNames>();
        this.arraylist.addAll(animalNamesList);
    }
    public class ViewHolder {     TextView name;    }
    @Override
    public int getCount() {    return animalNamesList.size();    }
    @Override
    public AnimalNames getItem(int position) {
        return animalNamesList.get(position);    }
    @Override
    public long getItemId(int position) {    return position;    }
    public View getView(final int position, View view, ViewGroup parent) {
        final ViewHolder holder;
        if (view == null) {      holder = new ViewHolder();
            view = inflater.inflate(R.layout.listview_items, null);
            holder.name = (TextView) view.findViewById(R.id.name);
            view.setTag(holder);
        } else {           holder = (ViewHolder) view.getTag();     }
        holder.name.setText(animalNamesList.get(position).getAnimalName());
        return view;
    }
    public void filter(String charText) {
        charText = charText.toLowerCase(Locale.getDefault());
        animalNamesList.clear();
        if (charText.length() == 0) {
            animalNamesList.addAll(arraylist);
        } else {       for (AnimalNames wp : arraylist) {
if (wp.getAnimalName().toLowerCase(Locale.getDefault()).contains(charText))
{                    animalNamesList.add(wp);                }
            }        }
 notifyDataSetChanged();
  }}
```

Figure 9.12 Code for ListViewAdapter.java of the SearchView.

```
package gjust.dde.searchview;
import android.os.Bundle;
import android.widget.ListView;
import android.widget.SearchView;
import androidx.appcompat.app.AppCompatActivity;
import java.util.ArrayList;
public class MainActivity extends AppCompatActivity implements
SearchView.OnQueryTextListener {
    ListView list;
    ListViewAdapter adapter;
    SearchView editsearch;
    String[] animalNameList;
    ArrayList<AnimalNames> arraylist = new ArrayList<AnimalNames>();
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        animalNameList = new String[]{"Lion", "Tiger", "Dog",
                "Cat", "Tortoise", "Rat", "Elephant", "Fox",
                "Cow","Donkey","Monkey"};
        list = (ListView) findViewById(R.id.listview);
        for (int i = 0; i < animalNameList.length; i++) {
            AnimalNames animalNames = new AnimalNames(animalNameList[i]);
            arraylist.add(animalNames);        }
        adapter = new ListViewAdapter(MainActivity.this, arraylist);
        list.setAdapter(adapter);
        editsearch = (SearchView) findViewById(R.id.search);
        editsearch.setOnQueryTextListener(this);    }
    @Override
    public boolean onQueryTextSubmit(String query) {  return false;    }
    @Override
    public boolean onQueryTextChange(String newText) {
        String text = newText;
        adapter.filter(text);        return false;
    }}
```

Figure 9.13 Code for MainActivity.java file of the SearchView.

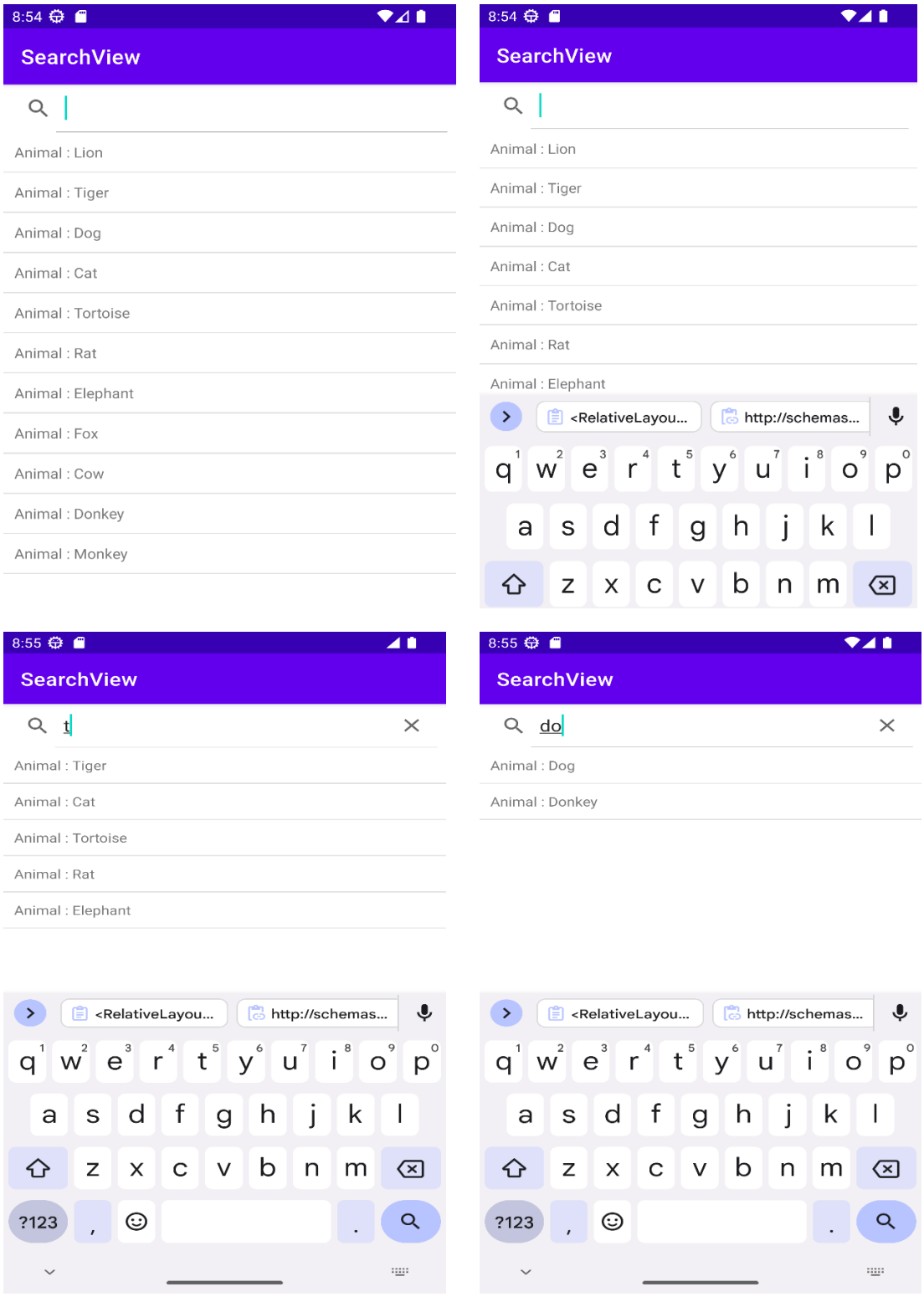


Figure 9.14 Screenshots of result of the SearchView.

9.6. TabHost

TabHost is a work like a container that holds multiple activity as view through tabs. This component uses two subsection one is set of tabs and other one is frame layout, where tab is used to click and open a particular tab and frame layout is used to display the of same clicked tab.

We can assign lot of data to one activity to display on screen responding tab. So, this is one of the best methods to use multiple tabs in mobile application interface through TabHost. One section may have tab list and other having information of every tabs. Here all process looks like a holding multiple pages on one screen without opening a new page.

For the implementation of this TabHost, we need to extend the TabActivity class into our MainActivity. A tab name is specifying the page content. and tag is used to track the content. Tab can have an indicator and icon for proper labeling. TabContentFactory class helps to create the content view and intent launches an Activity.

Some other classes used with TabHost are TabSpec, CharSequence, Drawable. CharSequence is sued to set type value for a label at tab. To display icon on tab, drawable is used. The function of each class has their own role which are explainable through the program that you can see below with various file names.

```
<resources>
    <string name="app_name">TabHost</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="title_activity_my">All
Activity</string>
    <string name="title_activity_about">About</string>
    <string
name="title_activity_contact">Contact</string>
</resources>
```

Figure 9.15 Code for string.xml file of the TabHost.

```
<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="10dp"
    android:paddingLeft="10dp"
    android:paddingRight="10dp"
    android:paddingTop="10dp"
    tools:context="gjust.dde.tabhost.Contact">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="Contact
Registrar, Guru Jambheshwar University of
Science  Technology, Hisar - 125001, Haryana (India)
Academic Support to teachers/students of study centres
Sh. Vinod Goyal,01662-263571
Dr. Vizender Singh,01662-263574,
Dr. Sunaina,01662-263158"
        android:textColor="#00f"
        android:textSize="25sp"
        android:textStyle="bold" />

</RelativeLayout>
```

Figure 9.16 Code for contact.xml file of the TabHost

```
<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="10dp"
    android:paddingLeft="10dp"
    android:paddingRight="10dp"
    android:paddingTop="10dp"
    tools:context="gjust.dde.tabhost.Contact">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:textSize="25sp"
        android:textColor="#f00"
        android:textStyle="bold"
        android:text=" Home                                The study material
has been prepared by the Directorate with the association of specialists
in the respective areas and the same is updated as and when the changes
are brought about in the course curriculum as per need of the market. In
order to have the interest of the students closely watched, the
Directorate has appointed teachers/course coordinators in the respective
discipline for each of the programme." />

</RelativeLayout>
```

Figure 9.17 Code for home.xml file of TabHost.

```
<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="10dp"
    android:paddingLeft="10dp"
    android:paddingRight="10dp"
    android:paddingTop="10dp"
    tools:context="gjust.dde.tabhost.About">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:textSize="25sp"
        android:textColor="#0f0"
        android:textStyle="bold"
        android:text="GENERAL
        The Guru Jambheshwar University of Science
        Technology, Hisar, was established on October 20, 1995 by an Act
         of the Legislature of the State of Haryana with the objectives
         'to facilitate and promote studies and research in emerging areas
of higher
        education with focus on new frontiers of technology, pharmacy,
         environmental studies, non-conventional energy sources and
         management studies and also to achieve excellence in these and
           " /></RelativeLayout>
```

Figure 9.18 Code for about.xml file of the TabHost.

Now create three files for about.java, home.java and contact.java in the main application folder.

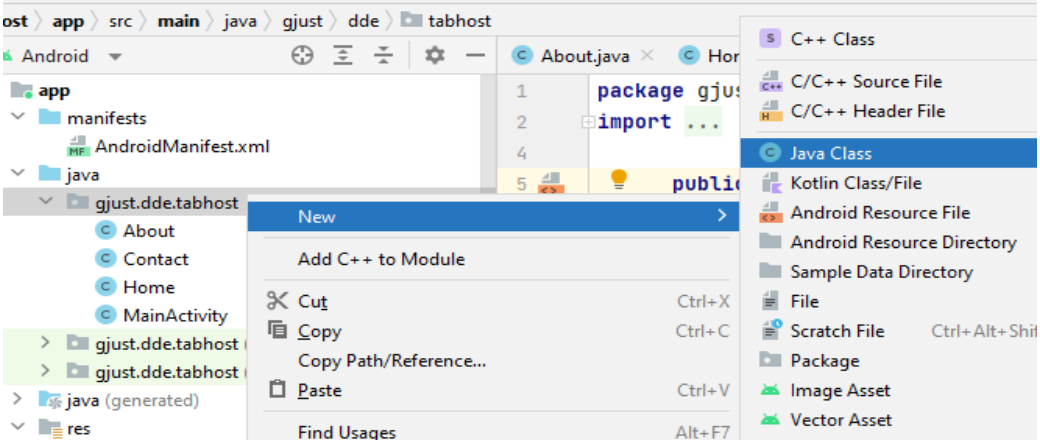


Figure 9.19 Option to create new these three java files.

```
package gjust.dde.tabhost;

import android.app.TabActivity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.TabHost;
import android.widget.Toast;
public class MainActivity extends TabActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TabHost tabHost = (TabHost) findViewById(android.R.id.tabhost);
        TabHost.TabSpec spec;
        Intent intent;
       spec = tabHost.newTabSpec("Home"); spec.setIndicator("HOME");
      intent = new Intent(this, Home.class);
              spec.setContent(intent);
              tabHost.addTab(spec);
              spec = tabHost.newTabSpec("Contact");
              spec.setIndicator("CONTACT");
              intent = new Intent(this, Contact.class);
                spec.setContent(intent);
                tabHost.addTab(spec);

                spec = tabHost.newTabSpec("About");
       spec.setIndicator("ABOUT");
              intent = new Intent(this, About.class);
              spec.setContent(intent);
              tabHost.addTab(spec);
              tabHost.setCurrentTab(1);
              tabHost.setOnTabChangedListener(new
       TabHost.OnTabChangeListener() {
                  @Override
                  public void onTabChanged(String tabId) {
              Toast.makeText(getApplicationContext(), tabId,
       Toast.LENGTH_SHORT).show();
                  }
              });
        }
       }
```

Figure 9.20 Code for MainActivity.java file of the TabHost.

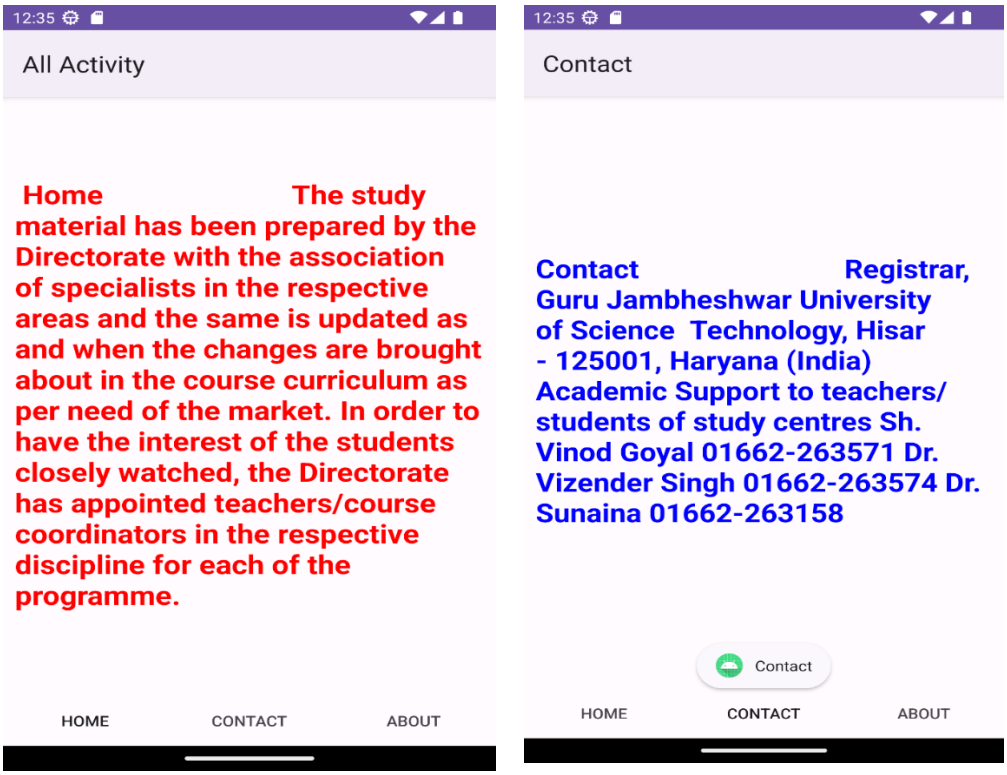


Figure 9.21 Screenshots of home and contact tab.

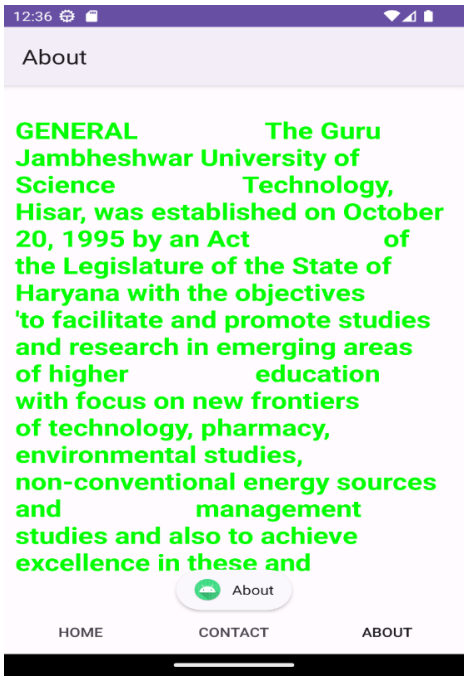


Figure 9.22 Screenshot of the about tab.

9.7. Dynamic ListView

ListView is a special UI component that is used in almost android application. It is used to display an array of data using list view. ListView provides number of functions by using them we can perform dynamic operation on list. This example explains how to add elements in list dynamically. ListView already covered in previous section 8.5 Chapter 7. here this topic will cover only the dynamic list designing process.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/idRLContainer"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <EditText
        android:id="@+id/idEditName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="4dp"
        android:layout_toLeftOf="@id/idBtnAdd"
        android:hint="Enter element" />
    <Button
        android:id="@+id/idBtnAdd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_margin="4dp"
        android:text="Add"
        android:textAllCaps="false" />
    <ListView
        android:id="@+id/idDLV"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@id/idEditName" />
</RelativeLayout>
```

Figure 9.23 Code for activity_main.xml file of DynamicListView.

```
package gjust.dde.dynamiclistview;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import androidx.appcompat.app.AppCompatActivity;
import java.util.ArrayList;
public class MainActivity extends AppCompatActivity {
    private ListView DLV;
    private Button Btn;
    private EditText item;
    private ArrayList<String> DList;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);.
        DLV = findViewById(R.id.idDLV);
        Btn = findViewById(R.id.idBtnAdd);
        item = findViewById(R.id.idEditName);
        DList = new ArrayList<>();
        DList.add("Apple");
        DList.add("Grapes");
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                                android.R.layout.simple_list_item_1, DList);
        DLV.setAdapter(adapter);
        Btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String items = item.getText().toString();
                if (!items.isEmpty()) {
                    DList.add(items);
                    adapter.notifyDataSetChanged();
                }
            }
        });
    }}
```

Figure 9.24 Code for MainActivity.java file of DynamicListView.

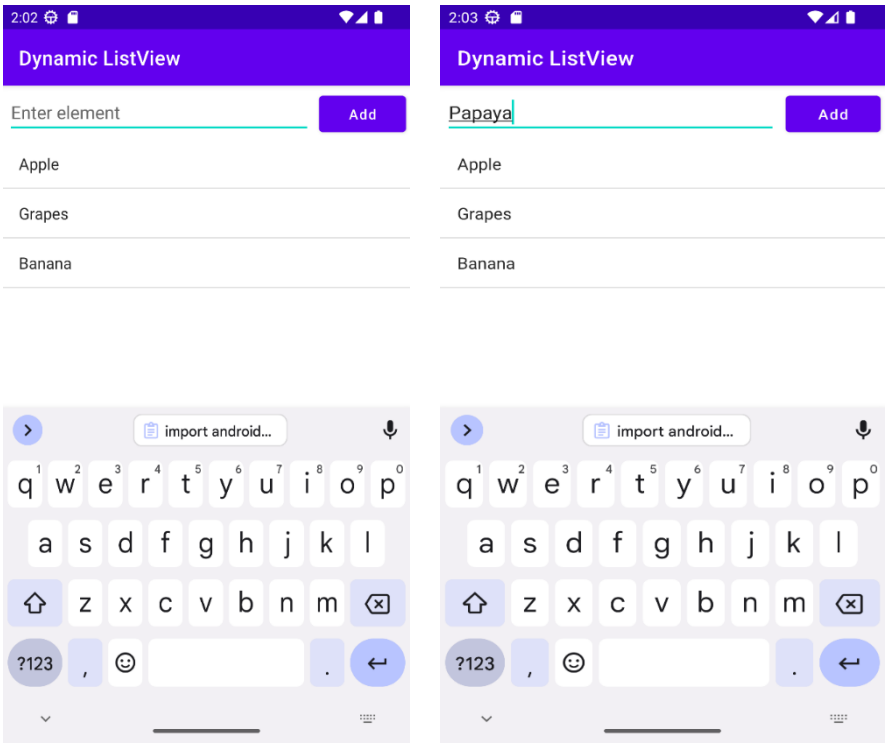


Figure 9.25 Screenshots of dynmic ListView.

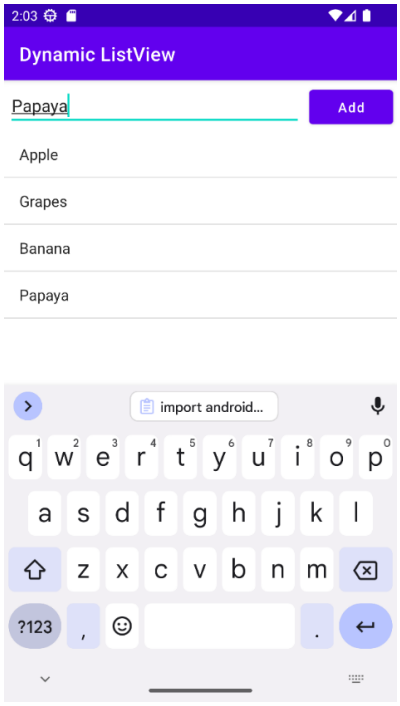


Figure 9.26 After adding content in Dynamic ListView.

9.8. Expanded ListView

It is an expansion of list with two levels. which scrolls vertically. this is somewhat different from simple ListView where every individual group are accessible to look their children It is an expand and collapse list.

The listener code can be written for every individual item or group to explore or assign any activity in better way. Adapter has their own role to control the supply of data this expansion list. When we are working expansion list there is an important role played by adapter. Because adapter is a supplier of data to children and group which is viewable and this can be in three ways.

- ExpandableListAdapter
- BaseExpandableListAdapter
- SimpleExpandableListAdapter

ExpandableListAdapter: this work for underlying data and customization.

BaseExpandableListAdapter: is a base class of all adapters for custom adapter preparing.

SimpleExpandableListAdapter: it is used to bind the static data from XML file.

Source of data can be specified separately as group in form of List of Maps. Each group is stored in an ArrayList separately for the Expandable List.

We use BaseExpandableListAdapter as base class for CustomAdapter. Here we write code for both type of function setOnGroupClickListener() and setOnChildClickListener() to display a toast message for child or group items.

```
<?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"
    android:orientation="vertical">
    <ExpandableListView
        android:id="@+id/simpleExpandableListView"
        android:layout_width="match_parent"
        android:layout_height="fill_parent"
        android:divider="#F8E42E"
        android:childDivider="#FF006F"
        android:dividerHeight="2dp" />
</RelativeLayout>
```

Figure 9.27 Code for activity_main.xml of Expandable ListView.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout android:layout_width="match_parent"
android:layout_height="match_parent"
    android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/android">
    <TextView
        android:id="@+id/sequence"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:paddingLeft="35sp"
        android:background="#F8EB7E"
        android:textColor="#4478F1"
      android:textAppearance="?android:attr/textAppearanceMedium" />
    <TextView
        android:id="@+id/subItem"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_toRightOf="@id/sequence"
       android:textAppearance="?android:attr/textAppearanceMedium" />
</RelativeLayout>
```

Figure 9.28 Code for subitems.xml of Expendable ListView

```
package gjust.dde.expendablelistview;
public class ChildInfo {
    private String sequence = "";
    private String name = "";
    public String getSequence()
         {
         return sequence;
         }
    public void setSequence(String sequence)
        {
        this.sequence = sequence;
        }
    public String getName()
         {
        return name;
        }
    public void setName(String name)
         {
        this.name = name;
        }
}
```

Figure 9.29 Code for ChildInfo.java of Expandable ListView.

```
package gjust.dde.expendablelistview;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseExpandableListAdapter;
import android.widget.TextView;
import java.util.ArrayList;
public class CustomAdapter extends BaseExpandableListAdapter {
    private Context context;    private ArrayList<GroupInfo> deptList;
    public CustomAdapter(Context context, ArrayList<GroupInfo> deptList) {
        this.context = context;
        this.deptList = deptList;    }
    @Override    public Object getChild(int groupPosition, int
childPosition) { ArrayList<ChildInfo> productList =
deptList.get(groupPosition).getProductList();
        return productList.get(childPosition);    }
    @Override    public long getChildId(int groupPosition, int
childPosition) {
        return childPosition;    }
    @Override  public View getChildView(int groupPosition, int
childPosition, boolean isLastChild,View view, ViewGroup parent) {
 ChildInfo detailInfo = (ChildInfo) getChild(groupPosition, childPosition);
        if (view == null) {   LayoutInflater infalInflater =
(LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            view = infalInflater.inflate(R.layout.subitems, null);        }
        TextView sequence = (TextView) view.findViewById(R.id.sequence);
        sequence.setText(detailInfo.getSequence().trim() + ". ");
        TextView childItem = (TextView) view.findViewById(R.id.subItem);
        childItem.setText(detailInfo.getName().trim());
        return view;    }
    @Override public int getChildrenCount(int groupPosition) {
        ArrayList<ChildInfo> productList =
deptList.get(groupPosition).getProductList();
        return productList.size();    }
    @Override public Object getGroup(int groupPosition) {
        return deptList.get(groupPosition);    }
    @Override public int getGroupCount() {
        return deptList.size();    }
    @Override    public long getGroupId(int groupPosition) {
        return groupPosition;    }
    @Override    public View getGroupView(int groupPosition, boolean
isLastChild, View view,ViewGroup parent) {
        GroupInfo headerInfo = (GroupInfo) getGroup(groupPosition);
        if (view == null) {LayoutInflater inf = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            view = inf.inflate(R.layout.groupitems, null); }
        TextView heading = (TextView) view.findViewById(R.id.heading);
        heading.setText(headerInfo.getName().trim());
        return view;    }
    @Override public boolean hasStableIds() {        return true;    }
    @Override public boolean isChildSelectable(int groupPosition, int
childPosition) {        return true;    }    }
```

Figure 9.30 Code for CustomAdapter of the Expandable ListView.

```
package gjust.dde.expendablelistview;
import android.os.Bundle;
import android.view.View;
import android.widget.ExpandableListView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import java.util.ArrayList;
import java.util.LinkedHashMap;
public class MainActivity extends AppCompatActivity {
    private LinkedHashMap<String, GroupInfo> subjects = new
LinkedHashMap<String, GroupInfo>();
    private ArrayList<GroupInfo> deptList = new ArrayList<GroupInfo>();
    private CustomAdapter listAdapter;
    private ExpandableListView simpleExpandableListView;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
loadData();
simpleExpandableListView = (ExpandableListView)
findViewById(R.id.simpleExpandableListView);
listAdapter = new CustomAdapter(MainActivity.this, deptList);
simpleExpandableListView.setAdapter(listAdapter);
expandAll();
simpleExpandableListView.setOnChildClickListener(new
ExpandableListView.OnChildClickListener() {
    @Override    public boolean onChildClick(ExpandableListView parent,
View v, int groupPosition, int childPosition, long id) {
        GroupInfo headerInfo = deptList.get(groupPosition);
   ChildInfo detailInfo =  headerInfo.getProductList().get(childPosition);
            //display it or do something with it
            Toast.makeText(getBaseContext(), " Clicked :: " +
headerInfo.getName()
                    + "/" + detailInfo.getName(),
Toast.LENGTH_LONG).show();
            return false;
        }
    });
    simpleExpandableListView.setOnGroupClickListener(new
ExpandableListView.OnGroupClickListener() {
        @Override
        public boolean onGroupClick(ExpandableListView parent, View v, int
groupPosition, long id) {
         GroupInfo headerInfo = deptList.get(groupPosition);
            Toast.makeText(getBaseContext(), " Header is :: " +
headerInfo.getName(),Toast.LENGTH_LONG).show();

            return false;
        }
    });
}
```

Figure 9.31 Code for MainActivity.java part-1 of the Expandable ListView.

```
private void expandAll() {
        int count = listAdapter.getGroupCount();
        for (int i = 0; i < count; i++){
            simpleExpandableListView.expandGroup(i);
        }
    }
    private void collapseAll() {
        int count = listAdapter.getGroupCount();
        for (int i = 0; i < count; i++){
            simpleExpandableListView.collapseGroup(i);
        }
    }
    private void loadData(){
        addProduct("GJUS&T Hisar","DDE");
        addProduct("GJUS&T Hisar","ECE");
        addProduct("GJUS&T Hisar","CSE");
       addProduct("CSE","AIML");
        addProduct("CSE","Btech");
        addProduct("CSE","MCA");
    }
    private int addProduct(String department, String product){
        int groupPosition = 0;
        GroupInfo headerInfo = subjects.get(department);
        if(headerInfo == null){
            headerInfo = new GroupInfo();
            headerInfo.setName(department);
            subjects.put(department, headerInfo);
            deptList.add(headerInfo);
        }
        ArrayList<ChildInfo> productList = headerInfo.getProductList();
        int listSize = productList.size();
        listSize++;
        ChildInfo detailInfo = new ChildInfo();
        detailInfo.setSequence(String.valueOf(listSize));
        detailInfo.setName(product);
        productList.add(detailInfo);
        headerInfo.setProductList(productList);
       groupPosition = deptList.indexOf(headerInfo);
        return groupPosition;
    }
}
```

Figure 9.32 Code for MainActivity.java part-2 of the Expandable ListView.

9.9. Summary

GridView is an Android ViewGroup that displays items in a two-dimensional, scrollable grid. It is used to display data in a structured way. WebView is permit the user to display web content within your application. It is used to display web pages, HTML content, and other web-based content. ScrollView allows the user to scroll through a list of items. It is used to display long lists of items that may not fit on the screen.

SearchView allows the user to search for items within an application. It is used to provide a search interface for users to quickly find what they are looking for. TabHost allows the user to switch between different tabs. It is used to provides a tabbed interface for users to easily navigate between different sections of an application. DynamicListView provides the user to dynamically add and remove items from a list. It is used to provide a dynamic list of items that can be easily modified. Expanded ListView is used to expand and collapse items in a list. It is used to provide a hierarchical view of data, allowing users to easily navigate between different levels of information.

9.10. Keywords

GridView: is an Android ViewGroup that displays items in a two-dimensional, scrollable grid. It is commonly used to display data in a structured format.

WebView: is a system component powered by Chrome that allows Android apps to display web content. It is commonly used in hybrid mobile applications,

ScrollView: is a type of Android ViewGroup that allows the user to scroll through a list of views.

SearchView: is a user interface element that allows users to search for information within an app.

TabHoast: is a class used to create a tabbed user interface.

DynamicListView: It is a library that allows developers to create dynamic lists of items that can be scrolled through and interacted with.

ExpandedListView: is a widget that displays a two-level list, where each level can be expanded or collapsed to show or hide its contents.

9.11. Check Your Progress

1. A view is a user _____.

2. Viewing components can handle _____ through various function.
3. GridView is a data viewing component that uses _____ scrolling grid.
4. BaseAdapter automatically arrange the layout of _____ for data.
5. View components are bridge between _____ and source of data.
6. WebView component helps in displaying a _____ view in mobile view.
7. For ScrollView, a class inherited from _____.
8. Multiple types of components use _____ like Table, Relative, and Linear layout.
9. SearchView is a _____ where an app user can search query.
10. TabHost is a_____ that holds multiple activity as view through tabs.
11. TabHost need to import the _____ class.
12. It is used to display an _____ of data using list view.
13. ExpandedListView is an expansion of list with _____.
14. Adapter is a supplier of data to _____ and group which is viewable.
15. We use BaseExpandableListAdapter as base class for _____.

9.12. Self-Assessment Test

1. What is GridView? Explain in detail.
2. How and when to use WebView for android app?
3. Discuss the various base class as well as current class function used for ScrollView.
4. What do you mean by content view? explain.
5. Define WebView and its various attributes.
6. How the SearchView is searching data for user in App?
7. Explain in brief about TabHost components.
8. What id DymanicListView? explain its functions in details.
9. Define ExpandedListView and explain with attributes and function in details.

9.13. Answers (Section 9.11)

1. interface
2. events
3. two-dimensional

4. grid
5. UI component
6. web page
7. android.widget.ScrollView
8. layouts
9. search interface
10. container
11. TabActivity
12. array
13. two levels
14. children
15. CustomAdapter

9.14. References/ Suggested reading

1. Headfirst Android Development, Dawn Griffiths, 1st edition, .O′Reilly
2. Android Programming for Beginners, John Horton,2nd edition, .Packt
3. Android Programming with Kotlin for Beginners, John Horton, 1st edition, Packt.
4. Head-First Kotlin, Dawn Griffiths, 1st edition, O'Reilly.
5. Android App Development, Michael Burton, 3rd edition.

SUBJECT: MOBILE APPLICATION DEVELOPEMENT

COURSE CODE: MCA-42	AUTHOR: Dr. Sunil Kumar Verma
LESSON NO. 10	
SQLite Database	

STRUCTURE

LEARNING OBJECTIVE

The objective of learning about SQLite is to gain an understanding of how to use the SQLite database engine to store, query, and manipulate data. This includes learning how to create tables, insert data into them, update existing data, delete data, and query the database for specific information.

Additionally, students should learn how to use SQLite's built-in functions and commands to perform more complex operations. Finally, students should understand the basics of database security and how to protect their data from unauthorized access.

Chapter 10. SQLite Database

10.1. Introduction

SQLite is a small, standalone and open-source database. It stores the file locally in the form of text file instead of a special database tool on computer. Android studio package come with SQLite database.

This database also supports relational database management features. As you know that in other database, we need to connect with them through a connection string or data provider such as ODBC for window, JDBC, oracle, SQL-Server etc.

The complete SQLite is implemented in C library that is must fast to save data directly with creating a bridge between DB tools and IDE package and is considered lightweight disk-based database. It minimizes the communication overhead for database connectivity to server or some other medium. One important thing the SQLite data saved in disk can be later imported or browse through the database tools like SQLite DB Browser.

A new SQLite module sqlite3 was designed by Gerhard Haring. It provides an interface as per the specification of DB-API 2.0. Once you decided to design a database, first create a structure of database how it will look then start implementation of the database with proper name and then create tables. An example of creating and deleting a table or simple database is discussed in next section:

The SQLite database library lies under the database package (android.database.sqlite) which have number of classes to manage the database created by you. SQLite library has number of classes with different use as described in next section.

10.2. SQLite Database

To create a new database, you need to use openOrCreateDatabase method with specified parameters like name database name and mode of database. This function will return a object of your database as shown in example.

SQLiteDatabase mydb = openOrCreateDatabase("Database name", MODE_PRIVATE, null);

Next function uses a cursor factory class that works like a result set. Similarly, there is different version of same function with various types of parameters.

openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags)

10.3. Tables

After creating database, we need to create a table to insert data. For this purpose, there is a method class exec SQL which is available in SQLiteDatabase class. The syntax for this function is:

mydb.execSQL("CREATE TABLE Exam(firstname VARCHAR, lastname VARCHAR);");

mydb.execSQL("INSERT INTO Exam VALUES('sunil','verma');");

This will insert some values into our table in our database. It requires some addition parameter according to the columns mentioned in create table function.

execSQL(String sql, Object[] bindArgs)

This function is used to insert data as well as to update or modify the already existing record in table using second arguments.

10.4. Fetching Record

To fetch data from the database a Cursor class object is used. For this process we may get a number of records which can be hold into result set and a function rawQuery is called for this process. The cursor can move forward to get the next record. Here, in this example, rs is a result set and object of Cursor class.

*Cursor rs = mydb.rawQuery("Select * from student", null);*

rs.moveToFirst();

String firstname = rs.getString(0);

String lastname = rs.getString(1);

List of functions having some other uses:

getColumnCount(): it returns total column of a table.

getColumnIndex(String colName): it returns an index of column by column name.

getColumnName(int colIndex): It returns the name of column based on index

getColumnNames(): It returns result as array for all the columns of table.

getCount(): It returns total rows of result set in the cursor

getPosition(): it returns current position of the cursor in table

isClosed(): It returns true if the cursor is closed and return false otherwise

10.5. SQLite Classes

SQLiteOpenHelper: this class provides a list of useful APIs for creating and managing a database. This class perform operation on demand instead of running or accessing database with application startup. The two most important functions are: getWritableDatabase() and getReadableDatabase().

This class can be inherited by any new subclass where the two functions must be override. the name of functions are: onCreate() and onUpgrade() method also known for callback.

Few important points:

For large data processing SQLite is more preferable.

It locally stored on device, later can be synchronized with main database.

The path for database is: /data/data/name of package/databases/ in android application. You can change this path for your convenience and change the path accordingly. It will work fine without any issues.

10.6. SQLite Spinner

SQLiteOpenHelper and SQLiteDatabase are used to use various function like onCreate(), onUpgrade() getWritableDatabase(). For Spinner is used to collect element entered in text box and click listener added to the spinner elements that display a toast message on click.

You can see the various files are creating for this application.

MainActivity.java perform all operation belonging to the App activities.

DetailActivity.java a custom file to handle activity separately for DbHandler class. It updates the list Adapter. SbHandler.java include all implementation related to database activities using SQLiteOpenHelper and SQLiteDatabase class.

list_row.xml file to list the items of adapter view.

detais.xml include two components list and back button.

activity_main.xml this file has main screen UI components like button and text boxes.

```
<?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="gjust.dde.sqlspinner.MainActivity">
    <EditText
        android:id="@+id/input_label"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="46dp"
        android:hint="Add data"
        android:ems="10" />
    <Button
        android:id="@+id/btn_add"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/input_label"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="67dp"
        android:text="Add" />
    <Spinner
        android:id="@+id/spinner"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/btn_add"
        android:layout_marginTop="70dp" />
</RelativeLayout>
```

Figure 10.1 Code for activity_main.xml file of SQLite Spinner.

```
package gjust.dde.sqlspinner;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import java.util.ArrayList;
import java.util.List;
public class DatabaseHandler extends SQLiteOpenHelper {
    private static final int DATABASE_VERSION = 1;
    private static final String DATABASE_NAME = "spinnerExample";
    private static final String TABLE_NAME = "labels";
    private static final String COLUMN_ID = "id";
    private static final String COLUMN_NAME = "name";
    public DatabaseHandler(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);    }
    @Override
  public void onCreate(SQLiteDatabase db) {
        String CREATE_ITEM_TABLE = "CREATE TABLE " + TABLE_NAME + "("
        + COLUMN_ID + " INTEGER PRIMARY KEY," + COLUMN_NAME + " TEXT)";
        db.execSQL(CREATE_ITEM_TABLE);
   }
  @Override
  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
    }
    public void insertLabel(String label){
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(COLUMN_NAME, label);
        db.insert(TABLE_NAME, null, values
        db.close();
    }
    public List<String> getAllLabels(){
        List<String> list = new ArrayList<String>();
       String selectQuery = "SELECT  * FROM " + TABLE_NAME;
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery(selectQuery, null);
        if (cursor.moveToFirst()) {
            do {
                list.add(cursor.getString(1));
            } while (cursor.moveToNext());
        }
        cursor.close();
        db.close();
```

Figure 10.2 Code for DatabaseHandler.java file of SQLite Spinner.

```
package gjust.dde.sqlspinner;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import java.util.List;
public class MainActivity extends AppCompatActivity implements
AdapterView.OnItemSelectedListener {
    Spinner spinner;
    Button btnAdd;
    EditText inputLabel;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        spinner = findViewById(R.id.spinner);
        btnAdd =  findViewById(R.id.btn_add);
        inputLabel = findViewById(R.id.input_label);
        spinner.setOnItemSelectedListener(this);
        loadSpinnerData();
  btnAdd.setOnClickListener(new View.OnClickListener() {
  @Override
 public void onClick(View arg0) {
String label = inputLabel.getText().toString();
if (label.trim().length() > 0) {
DatabaseHandler db = new DatabaseHandler(getApplicationContext());
db.insertLabel(label);
inputLabel.setText("");
InputMethodManager imm = (InputMethodManager)
 getSystemService(Context.INPUT_METHOD_SERVICE);
imm.hideSoftInputFromWindow(inputLabel.getWindowToken(), 0);
                    loadSpinnerData();} else {
Toast.makeText(getApplicationContext(), "Please enter label name",
                            Toast.LENGTH_SHORT).show();     } } });    }
    private void loadSpinnerData() {
        DatabaseHandler db = new DatabaseHandler(getApplicationContext());
        List<String> labels = db.getAllLabels();
        ArrayAdapter<String> dataAdapter = new
ArrayAdapter<String>(this,android.R.layout.simple_spinner_item, labels);
dataAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdow
n_item);        spinner.setAdapter(dataAdapter);    }
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int
position,long id) {
        String label = parent.getItemAtPosition(position).toString();
        Toast.makeText(parent.getContext(), "You selected: " + label,
                Toast.LENGTH_LONG).show();   }
    @Override
    public void onNothingSelected(AdapterView<?> arg0) {
           }}
```

Figure 10.3 Code for MainActivity.java file of SQLite Spinner.

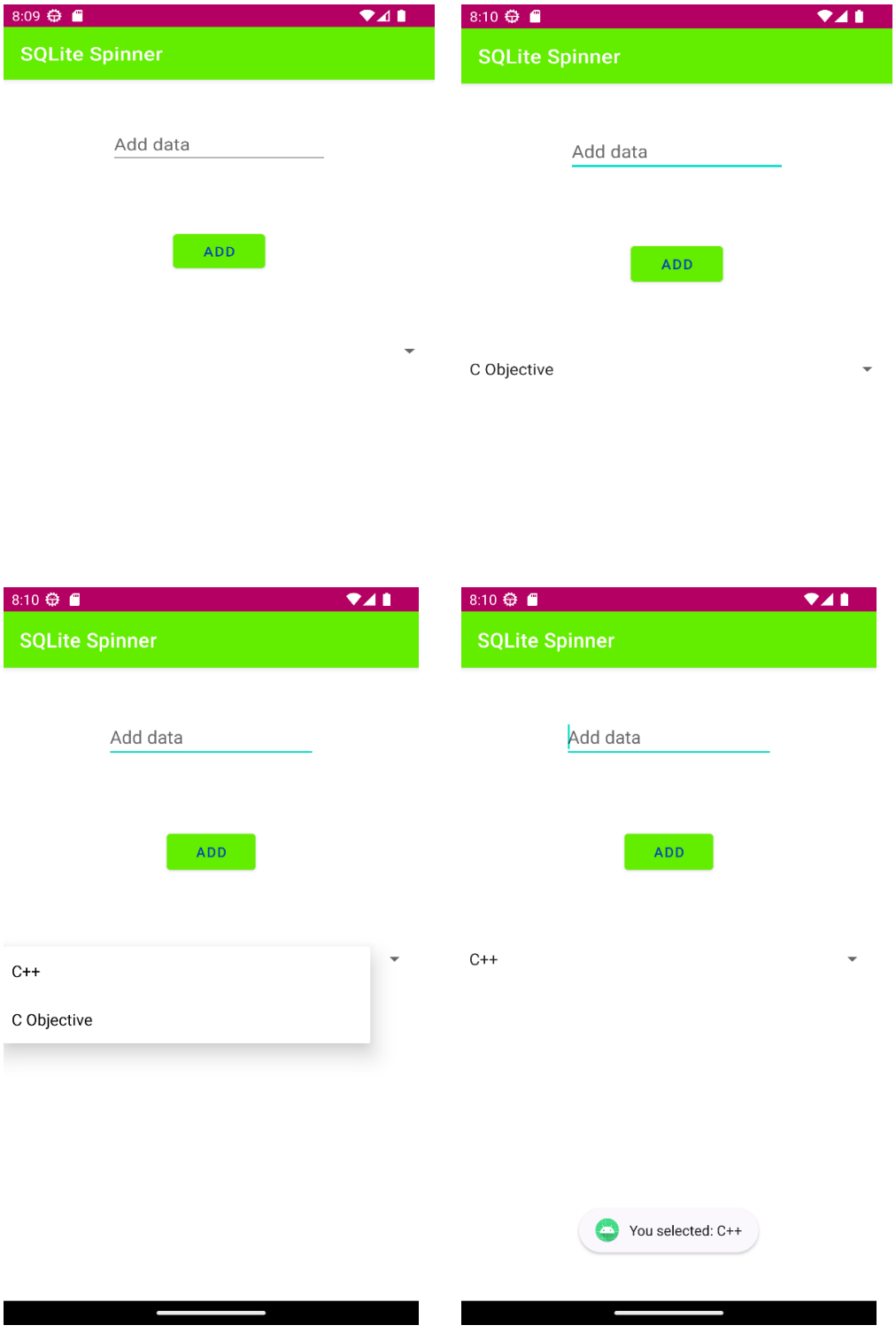


Figure 10.4 SQLite Spinner screenshots.

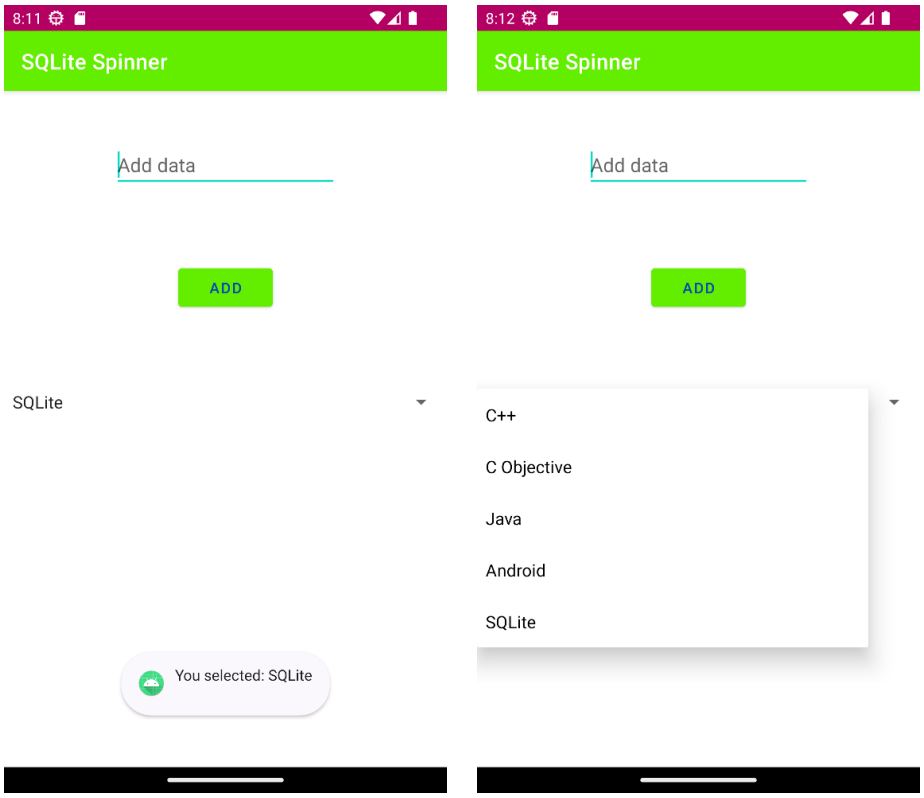


Figure 10.5 Screenshots of the SQLite Spinner.

10.7. SQLite ListView

SQLite is a RDBMS to apply various operations like storing, updating and retrieving record from the database. ListView is a way to view data from data adapter. It may use shared option for internal and external memory to maintain records entries. To dealing huge amount of records SQLite database is preferable for structured format.

The current example will show the various methods used for dealing with data using SQLite Database and custom ListView to display the records. SQLiteOpenHelper class is used to work with database. The application name is SQLite ListView.

List of files that used for designing this app is mentioned below with their purpose.

DbHandler.java: It include implementation details of SQLite database activities

activity_main.xml: file is available at path \res\layout in app that have all activity components design.

details.xml: is also in same path in \res\layout path to customize the listview

list_row.xml: is design to view the record in listview.

MainActivity.java: write code as shown in below file.

DetailsActivity.java: to handle the inserted data view.

AndroidManifest.xml: configure the few lines as written in this file.

```
package gjust.dde.sqlitelistview;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import java.util.ArrayList;
import java.util.HashMap;
public class DbHandler extends SQLiteOpenHelper {
    private static final int DB_VERSION = 1;
    private static final String DB_NAME = "usersdb";
    private static final String TABLE_Users = "userdetails";
    private static final String KEY_ID = "id";
    private static final String KEY_NAME = "name";
    private static final String KEY_LOC = "location";
    private static final String KEY_DESG = "designation";
    public DbHandler(Context context){
        super(context,DB_NAME, null, DB_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db){
        String CREATE_TABLE = "CREATE TABLE " + TABLE_Users + "("
                + KEY_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," + KEY_NAME
+ " TEXT,"
                + KEY_LOC + " TEXT,"
                + KEY_DESG + " TEXT"+ ")";
        db.execSQL(CREATE_TABLE);
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion){
         db.execSQL("DROP TABLE IF EXISTS " + TABLE_Users);
         onCreate(db);
    }
    void insertUserDetails(String name, String location, String
designation){
           SQLiteDatabase db = this.getWritableDatabase();
          ContentValues cValues = new ContentValues();
        cValues.put(KEY_NAME, name);
        cValues.put(KEY_LOC, location);
        cValues.put(KEY_DESG, designation);
           long newRowId = db.insert(TABLE_Users,null, cValues);
        db.close();
    }
     public ArrayList<HashMap<String, String>> GetUsers(){
        SQLiteDatabase db = this.getWritableDatabase();
        ArrayList<HashMap<String, String>> userList = new ArrayList<>();
        String query = "SELECT name, location, designation FROM "+
TABLE_Users;
        Cursor cursor = db.rawQuery(query,null);
```

Figure 10.6 Code for DbHandler.java of SQLite List View Part-1

```
while (cursor.moveToNext()){
            HashMap<String,String> user = new HashMap<>();

user.put("name",cursor.getString(cursor.getColumnIndex(KEY_NAME)));

user.put("designation",cursor.getString(cursor.getColumnIndex(KEY_DESG)));

user.put("location",cursor.getString(cursor.getColumnIndex(KEY_LOC)));
            userList.add(user);
        }
        return  userList;
    }
    // Get User Details based on userid
    public ArrayList<HashMap<String, String>> GetUserByUserId(int userid){
        SQLiteDatabase db = this.getWritableDatabase();
        ArrayList<HashMap<String, String>> userList = new ArrayList<>();
        String query = "SELECT name, location, designation FROM "+
TABLE_Users;
        Cursor cursor = db.query(TABLE_Users, new String[]{KEY_NAME,
KEY_LOC, KEY_DESG}, KEY_ID+ "=?",new String[]{String.valueOf(userid)},null,
null, null, null);
        if (cursor.moveToNext()){
            HashMap<String,String> user = new HashMap<>();

user.put("name",cursor.getString(cursor.getColumnIndex(KEY_NAME)));

user.put("designation",cursor.getString(cursor.getColumnIndex(KEY_DESG)));

user.put("location",cursor.getString(cursor.getColumnIndex(KEY_LOC)));
            userList.add(user);
        }
        return  userList;
    }
    // Delete User Details
    public void DeleteUser(int userid){
        SQLiteDatabase db = this.getWritableDatabase();
        db.delete(TABLE_Users, KEY_ID+" = ?",new
String[]{String.valueOf(userid)});
        db.close();
    }
    // Update User Details
    public int UpdateUserDetails(String location, String designation, int
id){
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues cVals = new ContentValues();
        cVals.put(KEY_LOC, location);
        cVals.put(KEY_DESG, designation);
        int count = db.update(TABLE_Users, cVals, KEY_ID+" = ?",new
String[]{String.valueOf(id)});
        return  count;
    }
}
```

Figure 10.7 Code for DbHandler.java of SQLite List View Part-2.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/fstTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:layout_marginTop="150dp"
        android:background="@color/yellow"
        android:text="Name" />
    <EditText
        android:id="@+id/txtName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:ems="10"/>
    <TextView
        android:id="@+id/secTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Location"
        android:background="@color/yellow"
        android:layout_marginLeft="100dp" />
    <EditText
        android:id="@+id/txtLocation"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:ems="10" />
    <TextView
        android:id="@+id/thirdTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Designation"
        android:background="@color/yellow"
        android:layout_marginLeft="100dp" />
    <EditText
        android:id="@+id/txtDesignation"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:ems="10" />
    <Button
        android:id="@+id/btnSave"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:text="Save Record"
        android:background="#FAE313"/>
</LinearLayout>
```

Figure 10.8 Code for activity_main.xml of SQLite List View.

```
<?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" >
    <ListView
        android:id="@+id/user_list"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:dividerHeight="1dp" />
    <Button
        android:id="@+id/btnBack"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="20dp"
        android:text="Back" />
</LinearLayout>
```

Figure 10.9 Code for detail.xml of SQLite List View.

```
<?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="wrap_content"
    android:orientation="horizontal"
    android:padding="5dip" >
    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textStyle="bold"
        android:textSize="17dp" />
    <TextView
        android:id="@+id/designation"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_marginTop="7dp"
        android:textColor="#343434"
        android:textSize="14dp" />
    <TextView
        android:id="@+id/location"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/designation"
        android:layout_alignBottom="@+id/designation"
        android:layout_alignParentRight="true"
        android:textColor="#343434"
        android:textSize="14dp" />
</RelativeLayout>
```

Figure 10.10 Code for list_row.xml of SQLite List View.

```
package gjust.dde.sqlitelistview;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    EditText name, loc, desig;
    Button saveBtn;
    Intent intent;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        name = (EditText)findViewById(R.id.txtName);
        loc = (EditText)findViewById(R.id.txtLocation);
        desig = (EditText)findViewById(R.id.txtDesignation);
        saveBtn = (Button)findViewById(R.id.btnSave);
        saveBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String username = name.getText().toString()+"\n";
                String location = loc.getText().toString();
                String designation = desig.getText().toString();
                DbHandler dbHandler = new DbHandler(MainActivity.this);
                dbHandler.insertUserDetails(username,location,designation);
                intent = new
Intent(MainActivity.this,DetailsActivity.class);
                startActivity(intent);
                Toast.makeText(getApplicationContext(), "Record Inserted
Successfully",Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

Figure 10.11 Code for DetailActivity.java of SQLite List View.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tutlane.sqliteexample">
    <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/Theme.AppCompat.Light">
        <activity android:name="gjust.dde.sqlitelistview.MainActivity"
            android:exported="true">
            <intent-filter>
             <action android:name="android.intent.action.MAIN" />
             <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="gjust.dde.sqlitelistview.DetailsActivity"
android:label="SQLite ListView"></activity>
    </application>
</manifest>
```

Figure 10.12 Code for AndroidManifest.xml of SQLite ListView.

```
package gjust.dde.sqlitelistview;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
    EditText name, loc, desig;
    Button saveBtn;
    Intent intent;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        name = (EditText)findViewById(R.id.txtName);
        loc = (EditText)findViewById(R.id.txtLocation);
        desig = (EditText)findViewById(R.id.txtDesignation);
        saveBtn = (Button)findViewById(R.id.btnSave);
        saveBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String username = name.getText().toString()+"\n";
                String location = loc.getText().toString();
                String designation = desig.getText().toString();
                DbHandler dbHandler = new DbHandler(MainActivity.this);
          dbHandler.insertUserDetails(username,location,designation);
          intent = new Intent(MainActivity.this,DetailsActivity.class);
          startActivity(intent);
Toast.makeText(getApplicationContext(), "Record Inserted
Successfully",Toast.LENGTH_SHORT).show();
            }
        });
    }}
```

Figure 10.13 Code for MainActivity.java of SQLite List View.

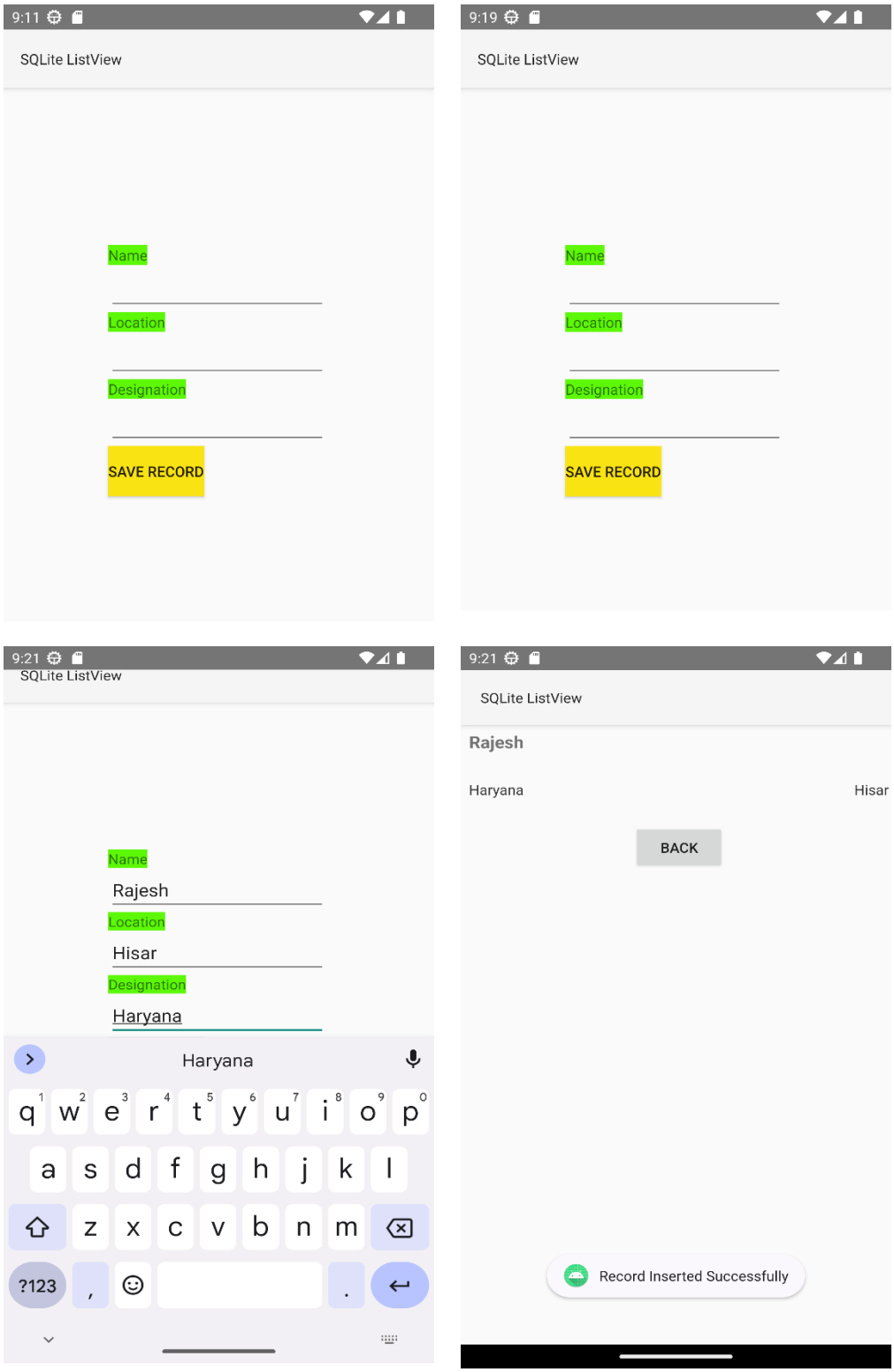


Figure 10.14 Screenshots of the SQLite ListView.

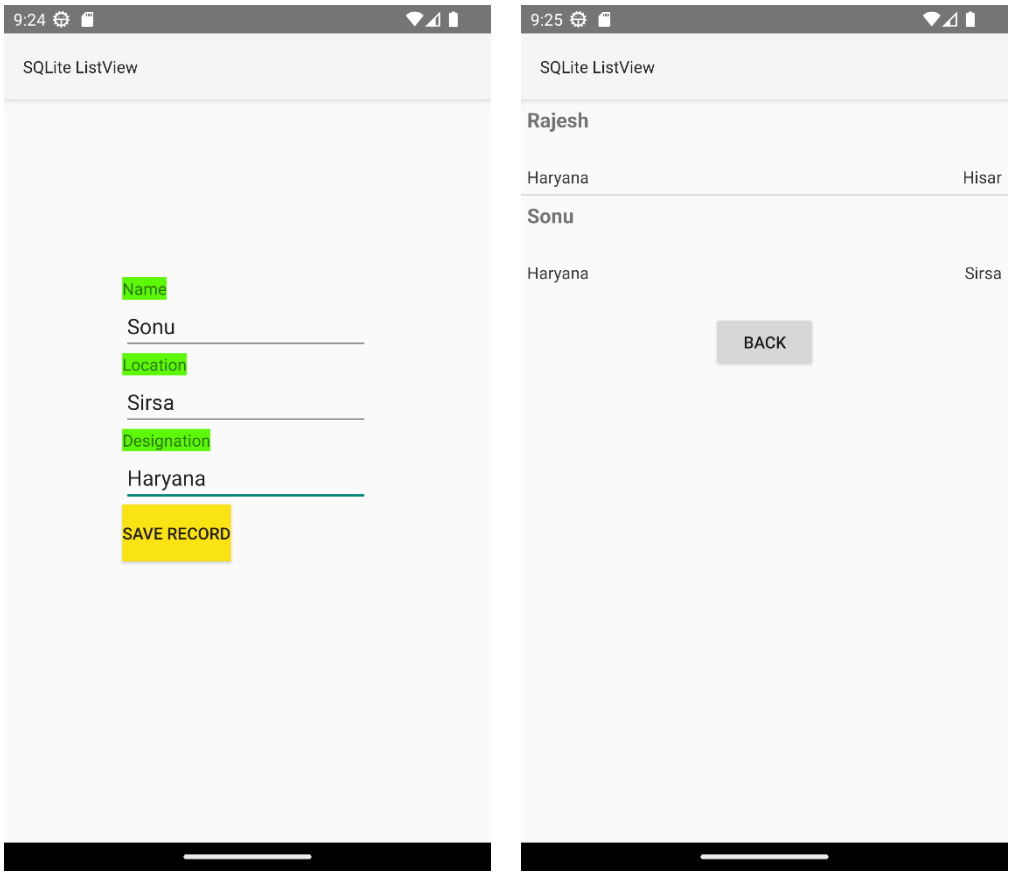


Figure 10.15 Screenshots of the SQLite ListView.

10.8. Summary

Android SQLite is a lightweight, open-source, relational database management system (RDBMS) designed specifically for Android devices. It is based on the SQL language and provides a powerful and efficient way to store, manage, and query data. It is used to store data locally on the device, making it an ideal choice for applications that need to store data on the device without relying on a server. It is also used to store application settings and user preferences. Android SQLite is highly secure, reliable, and efficient, making it a popular choice for mobile applications. This chapter has discussed the various classes SQLite Database, SQLiteOpenHelper, Cutsor, Query and data viewing components like Tables, Spinner and ListView.

10.9. Keywords

Database: The database is a collection of data that is organized in such a way that it can be easily accessed, managed, and updated. It is a structured set of data stored in a computer system.

Table: An SQLite table is a data structure used to store information in a SQLite database. It consists of columns and rows, where each column stores a specific type of data, and each row contains a single record.

Query: SQLite queries are commands used to interact with the database

Cursor: An SQLite Cursor is an object used to traverse the records in a database table. It is used to iterate over the rows of a result set, one row at a time. The cursor can be used to retrieve, add, update, and delete records from the database table.

ContentValues: ContentValues is a class in the Android SDK that is used to store key-value pairs. It is typically used when inserting or updating data in a SQLite database.

SQLiteOpenHelper: The SQLiteOpenHelper class is a helper class to manage database creation and version management. It creates a database the first time it is accessed, and provides methods to access and configure the database

SQLiteDatabase: SQLiteDatabase is an object that provides an interface to the SQLite database engine, allowing you to perform various operations such as creating, updating, and querying the database. It is a lightweight, self-contained, serverless, zero-configuration, transactional SQL database engine.

10.10. Check Your Progress

1. SQLite is a small, standalone and _____ database.
2. SQLite stores file locally in the form of _____ instead of a special database tool.
3. _____ is implemented in C library.
4. SQLite minimizes the _____ overhead for database connectivity to server.
5. To create a new database _____ method required.
6. Cursor factory class works like a _____.
7. To execute query/SQL command _____ method is used.
8. To fetch data from the database a _____ class object is used

9. _____ class provides a list of useful APIs to manage DB.
10. Spinner is used to collect element entered in _____.

10.11. Self-Assessment Test

1. What is SQLite?
2. What are the advantages of using SQLite?
3. How does SQLite compare to other database systems?
4. What are the different types of data types supported by SQLite?
5. How do you create a database in SQLite?
6. What are the different commands used in SQLite?
7. How do you query data from a SQLite database?
8. What are the best practices for using SQLite?
9. What are the different classes of SQLite database?
10. How do you optimize an SQLite database?

10.12. Answers (Section 10.10)

1. open-source
2. text file
3. SQLite
4. communication
5. openOrCreateDatabase
6. result set
7. mydb.execSQL
8. cursor
9. SQLiteOpenHelper
10. text box

10.13. References/ Suggested reading

1. Headfirst Android Development, Dawn Griffiths, 1st edition, .O′Reilly
2. Android Programming for Beginners, John Horton,2nd edition, .Packt
3. Android Programming with Kotlin for Beginners, John Horton, 1st edition, Packt.
4. Android App Development, Michael Burton, 3rd edition.

SUBJECT: MOBILE APPLICATION DEVELOPEMENT

COURSE CODE: MCA-42	AUTHOR: Dr. Sunil Kumar Verma
LESSON NO. 11	

XML & PARSING

STRUCTURE

LEARNING OBJECTIVE

To learn XML, you should have some basic knowledge of HTML. The learning objective of XML and Parsing is to understand the structure of XML documents, how to create them, and how to use a parser to extract data from them. Additionally, students should be able to identify common XML elements and attributes, as well as understand the differences between different types of parsers.

Chapter 11. XML and Parsing

11.1. Introduction

Extensible Mark-up Language is a one of the widely and popularly used data exchanging language. It is used to share/ exchange data between client and server machines. XML is an extensible markup language that is much similar to HTML. it is designed to handle data for transmission or holding for local in text format like tag based. Everything stored in xml is self-descriptive

for example

```
<book>
  <name>Android</name>
  <price>200</price>
  <pages>400</pages>
  <body>Best book for beginning</body>
</book>
```

This code explains every basic of XML as you can see there is no predefined tag for this. and it can be designed as per the requirement to holding data. The basic different between XML and HTML can be evaluated by you also, if you have basic knowledge. XML is using user defined tag whereas HTML predefined. XML is used to carry data and find what data is. HTML is focused on how data should be looks.

XML tags are according to above example: <book>, <price>, </price>

HTML tags are like <p>, <h1>, <table>.

XML is also useful to define tags and structure of documents also. Extensible of XML indicate that we can add new data at any time in XML existing file. Means the XML file support information like insert, delete and update.

There is option in XML file that it can have metadata. Metadata is the information of XML file which can be reference as DTD (Document Type Definition) or XSD (XML schema definition) for data validation. DTD file helps to manage the entities, element types and attributes. The DTD is used to set some constraint of XML file.

XML Schema is a language for describing and constraining the structure of XML documents. It is used to define the legal building blocks of an XML document, including elements, attributes, and data types. XPath is a language for navigating and selecting nodes in an XML document. It is used to locate elements, attributes, and other nodes in an XML document.

XSLT is a language for transforming XML documents into other formats such as HTML or plain text. It is used to convert XML documents into other formats such as HTML, PDF, or plain text. XML Namespaces are used to provide a unique name for elements and attributes in an XML document. They are used to avoid name collisions between elements and attributes from different sources**.** XML Security is a set of technologies used to secure XML documents. It includes encryption, digital signatures, and access control.

In android application, three types of XML parsers are mainly used to retrieved the needed information

- SAX Parser
- DOM Parser
- StAX (XMLPullParser)

11.2. SAX Parser

In android application, it is a simple API for XML which is focused on parsing the XML documents. It is latest parser than SAX and DOM. It stands for StAX (Streaming API for XML). SAX is an application programming interface to parse XML documents. It parses the document based on event generation while reading. A callback calling approach is sued to receive SAX events through custom handlers. API activated when an event dropped and after getting notification of callbacks method. It is the smarter form of managing documents.

SAX is also known for event based streaming parser. It starts process from the starting of document in sequential order and end when it found the root elements. It reads document from top to bottom and recognizes token as they appear in parsing of documents. When token found in parsing it call the callback method with relevant information. SAX is preferable because DOM consume more memory as compared to SAX. One more point SAX event occurs on availability of data in XML file or stream.

SAX parser is a programing interface between code and XML for data sharing for XML file through events.

The following events are related to SAX for XML:

Text Nodes

Starting and ending of Element

Processing Instructions

Comments

The properties of SAX Parser are depicted below as follows:

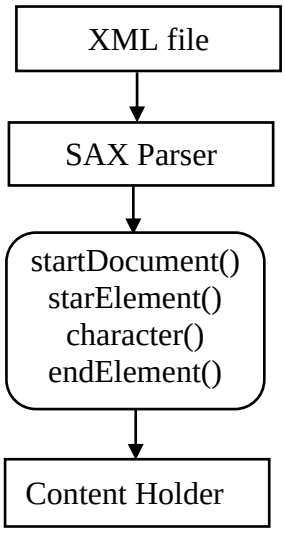


Figure 11.1 Structure of SAX Parser.

```
<?xml version="1.0" encoding="utf-8"?>
    <records>
        <employee>
            <name>Anju</name>
            <salary>20000</salary>
        </employee>
        <employee>
            <name>Bhushan</name>
            <salary>40000</salary>
        </employee>
        <employee>
            <name>Kuldeep</name>
            <salary>45000</salary>
        </employee>
    <employee>
        <name>Makhan</name>
        <salary>48000</salary>
    </employee>
    </records>
```

Figure 11.2 Code for employee.xml of the XMLSAX Parser.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
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"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:textAlignment="center"
    tools:context=".MainActivity">

        <TextView
            android:id="@+id/textView1"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:textAlignment="center"
            android:layout_alignParentStart="true"
            android:layout_alignParentTop="true"
            android:background="#FFFFFF"
            android:text="@string/textview"
            android:textSize="22dp"
            android:textColor="@color/purple_500"
            tools:ignore="MissingConstraints" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Figure 11.3 Code for activity_main.xml of the XML SAX parser.

```
package gjust.dde.xmlsaxparser;
import java.io.IOException;
import java.io.InputStream;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends Activity {    TextView tv;
    @Override  public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tv= findViewById(R.id.textView1);
  try {     SAXParserFactory factory = SAXParserFactory.newInstance();
            SAXParser saxParser = factory.newSAXParser();
            DefaultHandler handler = new DefaultHandler() {
                boolean name = false;
                boolean salary = false;
public void startElement(String uri, String localName,String qName,
   Attributes attributes) {
 if (qName.equalsIgnoreCase("name")) { name = true;      }
                    if (qName.equalsIgnoreCase("salary"))
                    {         salary = true;          }
                }
public void endElement(String uri, String localName, String qName) { }
                @SuppressLint("SetTextI18n")
                public void characters(char[] ch, int start, int length) {
if (name) {
  tv.setText(tv.getText()+"\n\n Name : " + new String(ch, start, length));
                    name = false;                }
if (salary) {
  tv.setText(tv.getText()+"\n Salary : " + new String(ch, start, length));
                    salary = false;                  }
                }           };
            InputStream is = getAssets().open("employee.xml");
            saxParser.parse(is, handler);
        }
        catch (ParserConfigurationException | IOException | SAXException e)
{
            e.printStackTrace();
        }
    }
}
```

Figure 11.4 Code for MainActivity.java file of the XML SAX Parser.

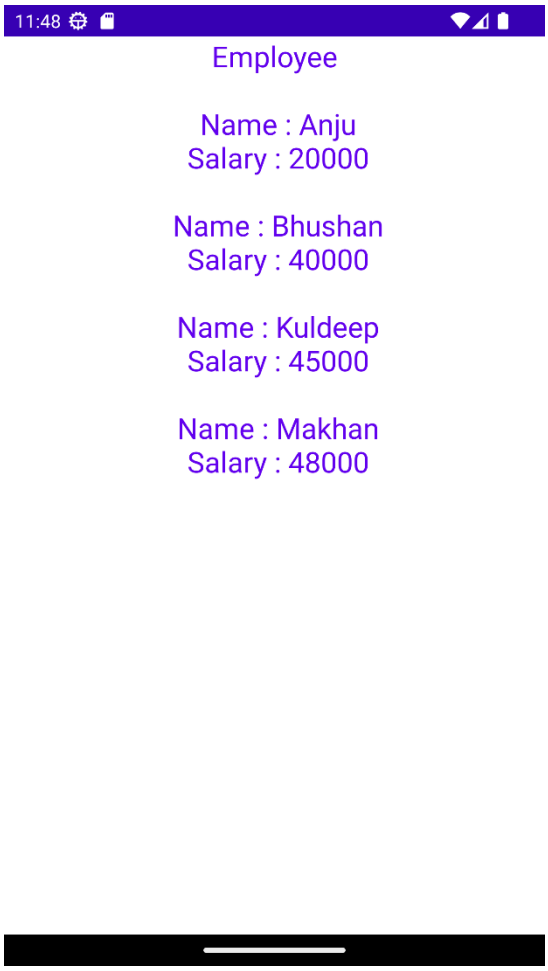


Figure 11.5 Result of XML SAX Parser.

11.3. DOM Parser

DOM parser is a Document Object Model based on object approach to access/parse the xml files. As it read about DOM process and loads the file in memory to parse the XML documents. So, the storing file in memory take lots of space, because it will read documents from starting to ending. Now we can understand the structure of XML file. DOM is not dependent on the event occurring. DOM parser needs to load complete XML file in memory then start to parse it. So, in comparison SAX memory-efficient than DOM. DOM parser support XPath that means it easily operate on the complete XML file in one go from loaded area.

- Prolog is an initial information about XML file
- Events are the occurring of every starting and ending tags of XML file
- Text is a content among the tag element in XML tag elements.
- Attributes are some additional information about tag. Also know for tag value.

```
<?xml version="1.0" encoding="utf-8"?>
<users>
    <user>
        <name>Ramesh</name>
        <designation>Technician</designation>
        <location>DHBVN</location>
    </user>
    <user>
        <name>Narender</name>
        <designation>Professor</designation>
        <location>Gurugram</location>
    </user>
    <user>
        <name>Pawan</name>
        <designation>Programmer</designation>
        <location>Hisar</location>
    </user>
    <user>
        <name>Rajkumar</name>
        <designation>Clerk</designation>
        <location>Hisar</location>
    </user>
    <user>
        <name>Sunil</name>
        <designation>Developer</designation>
        <location>Faridabad</location>
    </user>
</users>
```

Figure 11.6 Code for userdetails.xml file of XML DOM Parser.

```
<?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" >
    <ListView
        android:id="@+id/user_list"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:dividerHeight="1dp" />
</LinearLayout>
```

Figure 11.7 Code for activity_main.xml file of XML DOM Parser.

```
<?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="wrap_content"
    android:orientation="horizontal"
    android:padding="5dip" >
    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textStyle="bold"
        android:textSize="17dp" />
    <TextView
        android:id="@+id/designation"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_marginTop="7dp"
        android:textColor="#343434"
        android:textSize="14dp" />
    <TextView
        android:id="@+id/location"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/designation"
        android:layout_alignBottom="@+id/designation"
        android:layout_alignParentRight="true"
        android:textColor="#343434"
        android:textSize="14dp" />
</RelativeLayout>
```

Figure 11.8 Code for list_row.xml of XML DOM Parser.

```
package gjust.dde.xmldomparser;
import android.os.Bundle;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import androidx.appcompat.app.AppCompatActivity;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.HashMap;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
public class MainActivity extends AppCompatActivity {
    @Override protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    try{  ArrayList<HashMap<String, String>> userList = new ArrayList<>();
            ListView lv = findViewById(R.id.user_list);
            InputStream istream = getAssets().open("userdetails.xml");
 DocumentBuilderFactory builderFactory =
DocumentBuilderFactory.newInstance();
DocumentBuilder docBuilder = builderFactory.newDocumentBuilder();
            Document doc = docBuilder.parse(istream);
            NodeList nList = doc.getElementsByTagName("user");
            for(int i =0;i<nList.getLength();i++){
                if(nList.item(0).getNodeType() == Node.ELEMENT_NODE){
                    HashMap<String,String> user = new HashMap<>();
                    Element elm = (Element)nList.item(i);
                    user.put("name", getNodeValue("name",elm));
        user.put("designation", getNodeValue("designation",elm));
                    user.put("location", getNodeValue("location",elm));
                    userList.add(user);
                }       }
            ListAdapter adapter = new SimpleAdapter(MainActivity.this,
userList, R.layout.list_row,new String[]{"name","designation","location"},
new int[]{R.id.name, R.id.designation, R.id.location});
            lv.setAdapter(adapter);        }
catch (IOException e) {            e.printStackTrace();        }
catch (ParserConfigurationException e) {   e.printStackTrace(); }
catch (SAXException e) {     e.printStackTrace();    }    }
    protected String getNodeValue(String tag, Element element) {
        NodeList nodeList = element.getElementsByTagName(tag);
        Node node = nodeList.item(0);
if(node!=null){   if(node.hasChildNodes()){
   Node child = node.getFirstChild();
while (child!=null){    if(child.getNodeType() == Node.TEXT_NODE){
                        return  child.getNodeValue();
  }         }     }
   }        return "";
    }  }
```

Figure 11.9 Code for MainActivity.xml of XML DOM Parser.

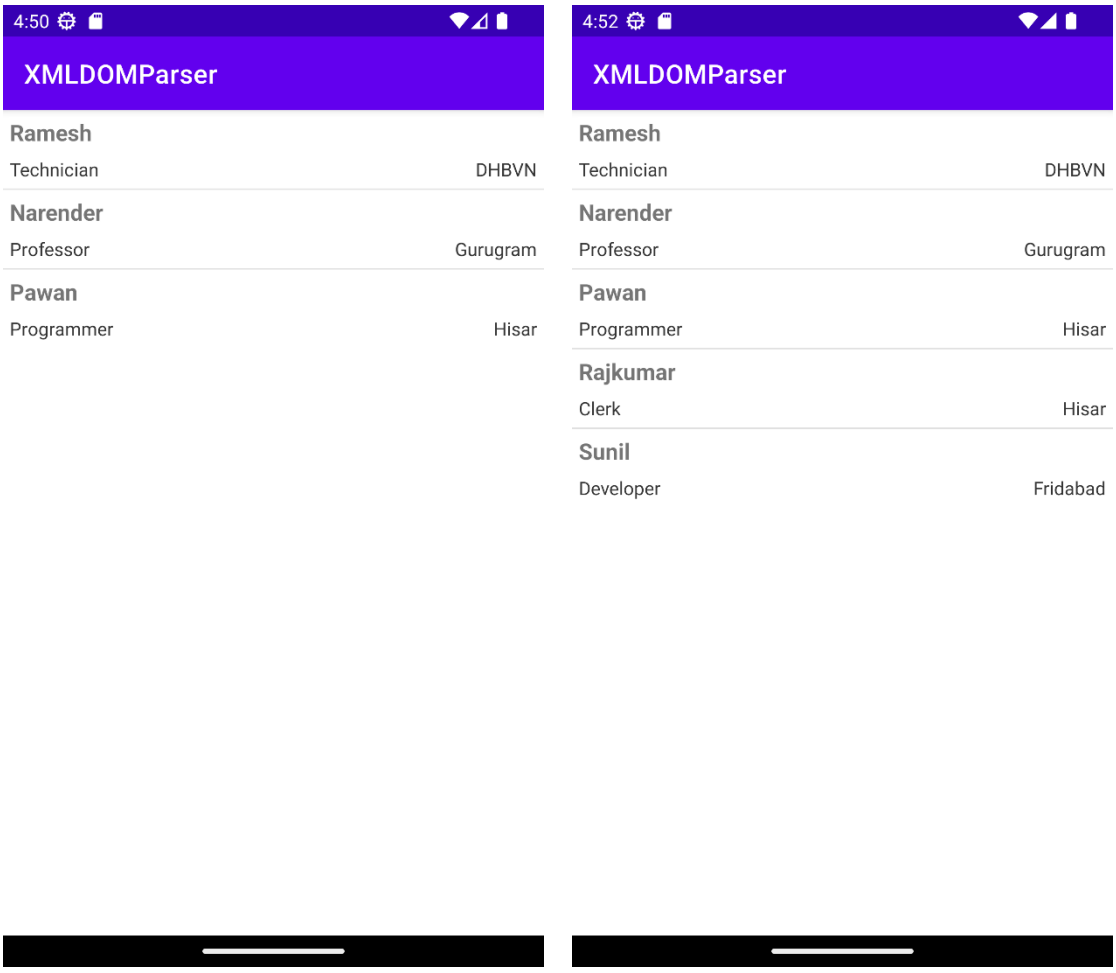


Figure 11.10 An output of XM DOM Parser App.

11.4. XML Pull Parser

StAX is a pull parser type API where a user or client has to ask the for information. It means it is on demand information retrieval process to get data. After getting data from XML, it will notify the client that your requested data is available. This pull parser is capable to read and write the XML file. Every parser has their merits and demerits to get data from XML file. Therefore, programmer should decide the use of parser based on their problem and requirements.

Few classes that are used to process the XML file elements:

- XMLEventReader
- XMLEventWriter

- XMLStreamReader
- XMLStreamWriter

For more details of these class, you can check in Android IDE by creating objects.

ContentHandler : It is a class that provides callback methods which is used to notify about the finding of elements in XML to the program.

```
<?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="wrap_content"
    android:orientation="horizontal"
    android:padding="5dip">
    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="17dp"
        android:textStyle="bold" />
    <TextView
        android:id="@+id/designation"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_marginTop="7dp"
        android:textColor="#343434"
        android:textSize="14dp" />
</RelativeLayout>
```

Figure 11.11 Code for list_row.xml of XML Pull Parser.

```
<?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" >
    <ListView
        android:id="@+id/user_list"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:dividerHeight="1dp" />
</LinearLayout>
```

Figure 11.12 Code for activity_main.xml of XML Pull Parser.

```xml
<?xml version="1.0" encoding="utf-8"?>
<users>
    <user>
        <name>Suresh</name>
        <designation>Scientist</designation>
    </user>
    <user>
        <name>Ravi</name>
        <designation>Officer</designation>
    </user>
    <user>
        <name>Kapil</name>
        <designation>Student</designation>
    </user>
</users>
```

Figure 11.13 Code for userdetails.xml of the XML Pull Parser.

```
package gjust.dde.xmlpullparser;
import android.os.Bundle;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import androidx.appcompat.app.AppCompatActivity;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.HashMap;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
   try{ ArrayList<HashMap<String, String>> userList = new ArrayList<>();
            ListView lv = findViewById(R.id.user_list);
            InputStream istream = getAssets().open("userdetails.xml");
            DocumentBuilderFactory builderFactory =
DocumentBuilderFactory.newInstance();
DocumentBuilder docBuilder = builderFactory.newDocumentBuilder();
            Document doc = docBuilder.parse(istream);
            NodeList nList = doc.getElementsByTagName("user");
```

Figure 11.14 Code for MainActivity.java of the XML Pull Parser Part-1.

```
    for(int i =0;i<nList.getLength();i++){
                if(nList.item(0).getNodeType() == Node.ELEMENT_NODE){
                    HashMap<String,String> user = new HashMap<>();
                    Element elm = (Element)nList.item(i);
                    user.put("name", getNodeValue("name",elm));
                    user.put("designation",
getNodeValue("designation",elm));
                    user.put("location", getNodeValue("location",elm));
                    userList.add(user);
                }
            }
            ListAdapter adapter = new SimpleAdapter(MainActivity.this,
userList, R.layout.list_row,new String[]{"name","designation","location"},
new int[]{R.id.name, R.id.designation});
            lv.setAdapter(adapter);
        }
        catch (IOException e) {
            e.printStackTrace();
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        } catch (SAXException e) {
            e.printStackTrace();
        }
    }
    protected String getNodeValue(String tag, Element element) {
        NodeList nodeList = element.getElementsByTagName(tag);
        Node node = nodeList.item(0);
        if(node!=null){
            if(node.hasChildNodes()){
                Node child = node.getFirstChild();
                while (child!=null){
                    if(child.getNodeType() == Node.TEXT_NODE){
                        return  child.getNodeValue();
                    }
                }
            }
        }
        return "";
    }
}
```

Figure 11.15 Code for MainActivity.java of the XML Pull Parser Part-2.

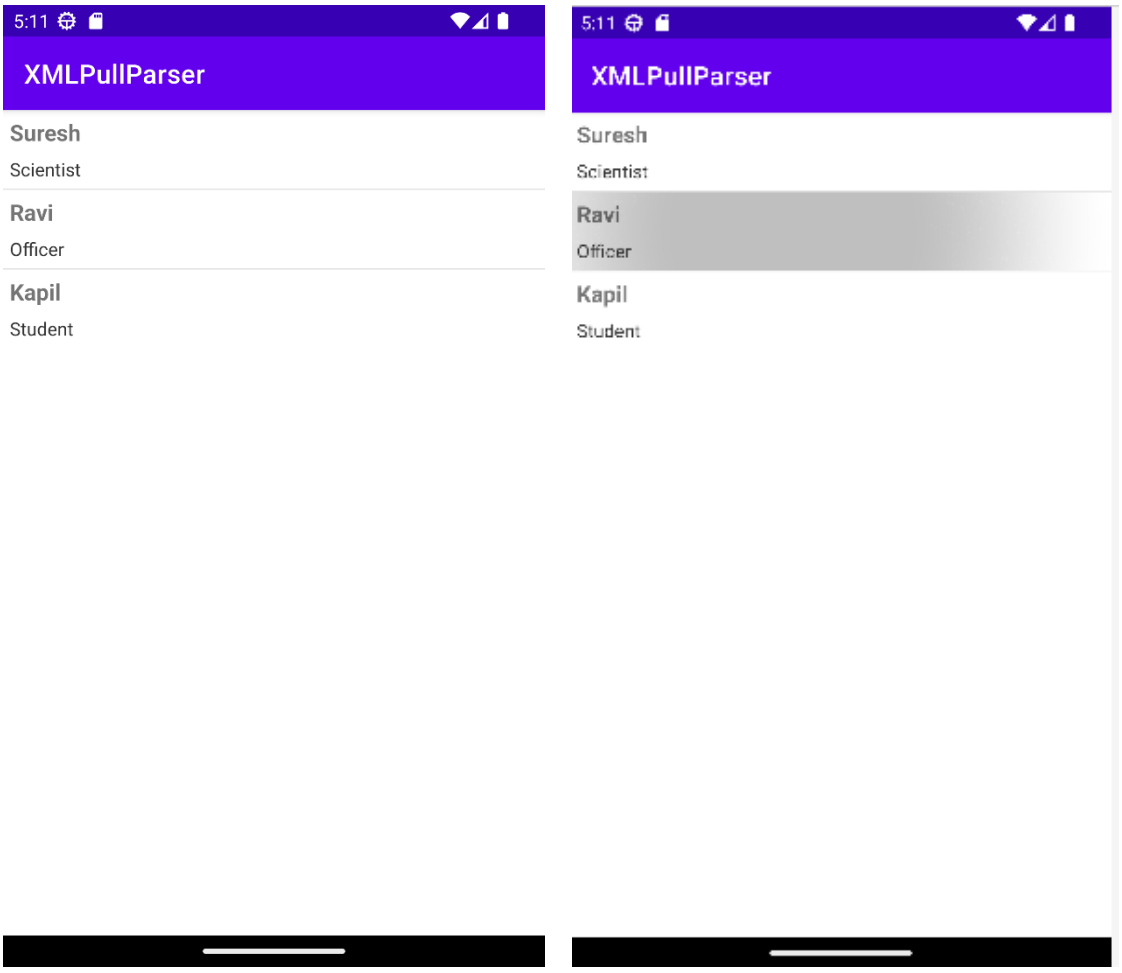


Figure 11.16 Screenshots of XML Pull Parser.

11.5. Summary

This chapter has discussed about the basics of XML and parsing techniques. XML (Extensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. XML is used to store and transport data, and is designed to be self-descriptive. Parsing is the process of analyzing a string of symbols, either in natural language or in computer languages, according to the rules of a formal grammar. Parsing is used to interpret and analyze the structure of a given text or sentence. XML parsing is the process of analyzing an XML document and breaking it into pieces to access the data it contains. XML parsers are used to read XML documents and make their content available to programs.

11.6. Keywords

XML: is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

Parsing: is the process of analyzing the structure and content of an XML document.

DOM: is a programming interface for HTML and XML documents.

SAX: is an event-driven, serial access parser API for XML.

ElementTree: is a class that allows developers to parse and create XML documents.

Namespaces: are used to provide a unique name to an element or attribute in an XML document.

Validation: is the process of checking an XML document against a schema to ensure that it is valid.

11.7. Check Your Progress

1. XML is a markup language that defines a set of rules for_____ documents.
2. XML is representation of data in a format that is both human-readable and_____.
3. XML parsing is the process of taking XML code and _____ it into an understandable format.
4. For parsing a variety of methods are used like _____ and SAX.
5. DOM is a tree-based parsing method that reads the entire XML document into _____.
6. DOM creates a _____ tree structure of the elements.
7. SAX is an _____ parsing method that reads the XML document sequentially
8. SAX _____ events when it encounters certain elements.
9. XML parsing can also be done using XML _____.
10. Schema of XML is a language for describing the _____ of an XML document.

11.8. Self-Assessment Test

1. What is XML?
2. What are the benefits of using XML?
3. How does XML differ from HTML?
4. What is an XML Parser?

5. What are the different types of XML Parsers?
6. How do you validate an XML document?
7. What is the purpose of an XML Schema?
8. What is the difference between DTD and XML Schema?
9. How do you parse an XML document using DOM Parser?
10. What are the advantages of using SAX Parser over DOM Parser?

11.9. Answers (Section 11.7)

1. encoding
2. machine-readable
3. transforming
4. DOM
5. memory
6. hierarchical
7. event-based
8. triggers
9. Schema
10. structure

11.10. References/ Suggested reading

1. Headfirst Android Development, Dawn Griffiths, 1st edition, .O′Reilly
2. Android Programming for Beginners, John Horton,2nd edition, .Packt
3. Android Programming with Kotlin for Beginners, John Horton, 1st edition, Packt.
4. Android App Development, Michael Burton, 3rd edition.

SUBJECT: MOBILE APPLICATION DEVELOPEMENT	
COURSE CODE: MCA-42	AUTHOR: Dr. Sunil Kumar Verma
LESSON NO. 12	
JSON & Parsing	

STRUCTURE

LEARNING OBJECTIVE

The learning objective of JSON and Parsing is to understand how to use the JavaScript Object Notation (JSON) format to store and exchange data, as well as how to parse JSON data into a usable

format. This includes understanding the syntax of JSON, how to create valid JSON objects, and how to use various tools and libraries to parse JSON data. Additionally, students should be able to identify when and why JSON is the best format for storing and exchanging data.

Chapter 12. JSON & Parsing

12.1. Introduction

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write and for machines to parse and generate. JSON is a text-based, human-readable format for representing simple data structures and associative arrays (called objects).

Parsing is the process of analyzing a string of symbols, either in natural language or in computer languages, according to the rules of a formal grammar. It involves resolving a string into its component parts and describing their syntactic roles. Parsing is used to validate that strings are syntactically. JSON is used in different area - social media, Online Shopping, Video Streaming, Cyber Security, Artificial Intelligence, Cloud Computing, Big Data, Internet of Things (IoT), Augmented Reality (AR).

JSON is stands for Java Script Object Notation which is a lightweight text file and easy to read. this is more useful for data interchanging. JSON object was specially designed by Douglas Crockford. The JSON file extension is. json and media type is mention by application\json. JSON is used by various languages like PERL, PHP, Python, Java, Ruby, JavaScript, Ajax, C#, etc.

JSON is more advance than XML and supports different simple and complex data type like: array, object, string, number and values.

12.2. Example of JSON

A basic example of JSON explains how can you design a file:

File: record.json

```
{"Students":[
   {"name" : "Sunil", "email" : "ermaihack@gmail.com"},
   {"name" : "Kumar", "email" : "ermaihack1@gmail.com"},
```

{"name" : "Sk", "email" : "ermaihack2@gmail.com"}

]}

12.3. Characteristic of JSON

- It is popular for data transition
- JSON is basically categorized in two structural entities, first is name-value pairs and second is ordered list of values.
- JSON is become a universal data structure for almost programming language.
- The name value pairs are recognized by an object, struct, dictionary, record, etc.
- The ordered list is recognized as an array, list etc.

12.4. JSON Components

Following notation are used to represent the components.

Array ([): Square bracket ([) indicates an array of JSON. The values of array can be a combination JSON Objects. An array can be unique collecting of Booleans, Doubles, Integers, Longs, Strings, null or NULL etc.

Objects ({): Here curly bracket is used to represent an object. It holds the information in key-value pair. JSONObject can also be a collection of different objects like Booleans, Doubles, Integers, Longs, Strings, null or NULL, JSON Arrays and Objects

key: key is an object attribute and it is string type.

Value: Every key contains a value which is a primitive datatype (e.g., integer, double, string etc.).

12.5. Working with JSON in Android

JSON stands for JavaScript Object Notation. It is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. In Android, JSON is used to store and exchange data between the application and the server. It is also used to parse data from web services and APIs.

To work with JSON in Android, you need to use the JSONObject class. This class provides methods for creating, parsing, and manipulating JSON objects. You can also use the JSONArray class to create and manipulate arrays of JSON objects. To parse a JSON string, you can use the JSONObject.parse() method. This method takes a string as an argument and returns a JSONObject. You can then use the methods of the JSONObject class to access and manipulate the data in the object.

To create a JSON string, you can use the JSONObject.toString() method. This method takes a JSONObject as an argument and returns a string representation of the object. To send data to a server, you can use the HttpURLConnection class. This class provides methods for sending data to a server using HTTP requests. You can use the HttpURLConnection.setRequestProperty() method to set the request headers, and the HttpURLConnection.getOutputStream() method to write the data to the request body.

To receive data from a server, you can use the HttpURLConnection.getInputStream() method. This method returns an InputStream object, which you can use to read the data from the server. JSON is a powerful and versatile data format that can be used to store and exchange data between applications and servers. With the help of the JSONObject and JSONArray classes, you can easily parse and manipulate JSON data in Android.

JSON Classes

- JSONObject: This class is used to create a JSON object. It contains methods for creating, parsing, manipulating and encoding JSON data.
- JSONArray: This class is used to create a JSON array. It contains methods for creating, parsing, manipulating and encoding JSON data.
- JSONTokener: This class is used to parse a JSON string. It contains methods for parsing, manipulating and encoding JSON data.

JSON Class Methods

- optString(): This method is used to get the string value associated with a given key.
- optBoolean(): This method is used to get the boolean value associated with a given key.
- optInt(): This method is used to get the integer value associated with a given key.
- optLong(): This method is used to get the long value associated with a given key.

- optDouble(): This method is used to get the double value associated with a given key.
- optJSONObject(): This method is used to get the JSONObject value associated with a given key.
- optJSONArray(): This method is used to get the JSONArray value associated with a given key.
- put(): This method is used to add a name/value pair to the JSONObject.
- remove(): This method is used to remove a name/value pair from the JSONObject.
- toString(): This method is used to convert the JSONObject into a String.
- toJSONArray(): This method is used to convert the JSONObject into a JSONArray.
- toJSONObject(): This method is used to convert the JSONObject into a JSONObject.

12.6. Parsing JSON in Android

Parsing JSON in Android can be done using the JSONObject and JSONArray classes. The JSONObject class is used to parse a JSON string into an object, while the JSONArray class is used to parse a JSON array into an array of objects. To parse a JSON string, first create a JSONObject object and then use the getString() method to retrieve the value of a particular key. To parse a JSON array, first create a JSONArray object and then use the getJSONObject() method to retrieve each element in the array.

12.7. Storing and Retrieving JSON Data

Storing JSON Data

- Use SharedPreferences: SharedPreferences is a key-value store that allows you to store primitive data types such as strings, booleans, and integers. To store JSON data, you can convert it into a string and then save it in SharedPreferences.
- Use SQLite Database: You can also store JSON data in an SQLite database. To do this, you need to create a table with a column that can store JSON data. Then, you can use the INSERT statement to insert the JSON data into the table.

Retrieving JSON Data

- Use SharedPreferences: To retrieve JSON data from SharedPreferences, you can use the getString() method to get the string representation of the JSON data and then use a JSON parser to convert it back into a JSON object.
- Use SQLite Database: To retrieve JSON data from an SQLite database, you can use the SELECT statement to query the table and then use a JSON parser to convert the result set into a JSON object.

12.8. JSON Parser

This example explains how to parse the JSON file and used in android application.

```
<?xml version="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"
    tools:context="gjust.dde.jsonparser.MainActivity">
    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="50dp"
        android:text="Name"
        android:textColor="#000"
        android:textSize="20sp" />
    <TextView
        android:id="@+id/salary"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="80dp"
        android:text="Salary"
        android:textColor="#000"
        android:textSize="20sp" />
</RelativeLayout>
```

Figure 12.1 Code for activity_main.xml file of JSON Parser.

```
package gjust.dde.jsonparser;
import android.os.Bundle;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
import org.json.JSONException;
import org.json.JSONObject;
public class MainActivity extends AppCompatActivity {
    String JSON_STRING =
"{\"employee\":{\"name\":\"Priyanka\",\"salary\":90000}}";
    String name, salary;
    TextView employeeName, employeeSalary;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // get the reference of TextView's
        employeeName = (TextView) findViewById(R.id.name);
        employeeSalary = (TextView) findViewById(R.id.salary);
        try {
            JSONObject obj = new JSONObject(JSON_STRING);
            JSONObject employee = obj.getJSONObject("employee");
            name = employee.getString("name");
            salary = employee.getString("salary");
            employeeName.setText("Name: "+name);
            employeeSalary.setText("Salary: "+salary);
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}
```

Figure 12.2 Code or MainActivity.java file of JSON Parser.

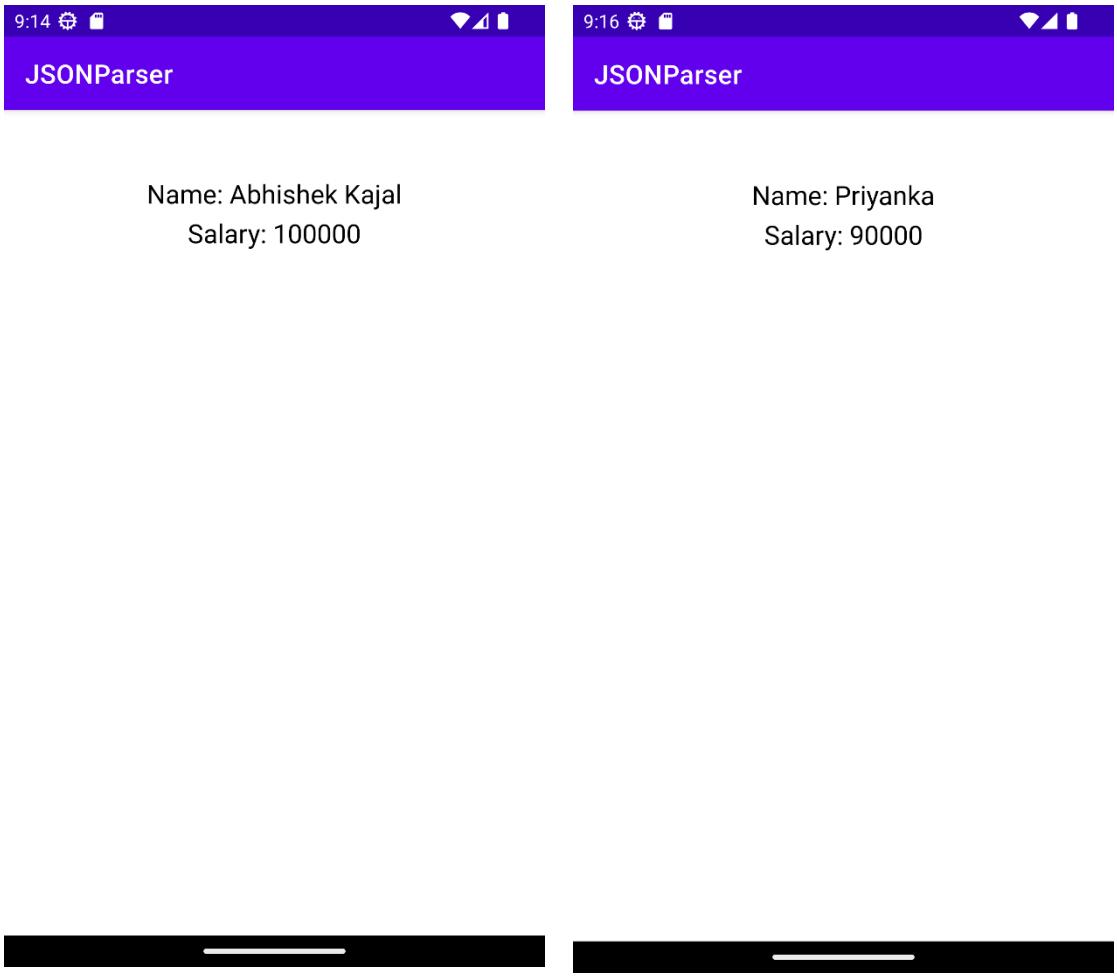


Figure 12.3 Screenshots of JSON parser.

12.9. Summary

JSON is a lightweight data-interchange format. It is a text-based, human-readable format for representing simple data structures and associative arrays (called objects). JSON is often used for serializing and transmitting structured data over a network connection. Parsing is the process of taking a string of JSON

In Android, parsing JSON data can be done using the JSONObject and JSONArray classes. The JSONObject class is used to parse a JSON string into an object, while the JSONArray class is used to parse a JSON array into an array of objects. To parse a JSON string, first create a new instance of the JSONObject class and then call the getString() method to get the value of a particular key. To parse a

JSON array, first create a new instance of the JSONArray class and then call the getJSONObject() method to get the value of a particular element.

Parsing is the process of taking a string of data and breaking it down into its component parts. This can be done using various methods, such as regular expressions, string manipulation, or a library like JSON.parse. The result of parsing is usually an object or array that can data and breaking it down into its component parts. This can be done using various methods, such as regular expressions, string manipulation, or a library like JSON.parse. The result of parsing is usually an object or array that can be used to access the data in a structured way.

12.10. Keywords

JSONObject : is a class in the Java programming language that is used to store and process data in the form of key-value pairs

JSONArray: A JSON array is an ordered collection of values. It is enclosed in square brackets and contains a comma-separated list of values. Each value can be a string, number, object, array, true, false or null.

JsonParser: is a software library that provides functions for parsing and manipulating data in the JavaScript Object Notation (JSON) format. It is commonly used to exchange data between web applications and servers.

JsonElement: represents an element of JSON data. It can be either a JsonObject, a JsonArray, It provides methods to access and modify the data it contains.

12.11. Check Your Progress

1. JSON.parse() is a method used to convert a _____ into a JavaScript object.
2. JSON.stringify() converts a _____ into a JSON string.
3. XML is a _____ language that defines a set of rules for encoding documents.
4. JSON is used to store and _____ data between a server and a web application.
5. XML is used to store and exchange data between _____ applications.
6. JSON is a _____ format.
7. JSON is easier to read and write than XML, and it is also more_____.

8. JSON is the preferred format for data interchange between _____.
9. JSON is a language _____ data format.

12.12. Self-Assessment Test

1. How to parse a JSONArray from a String in Android?
2. List the differences between JSON and XML?
3. How do you parse JSON in Android?
4. What is the syntax of JSON?
5. How do you access an element of a JSONArray in Android?
6. How to convert a JSONArray to an ArrayList in Android?
7. What are the benefits of using JSON?
8. How do you convert a JSONArray to a List in Android?
9. Define JSON and why is it used?
10. What are the advantages of using JSON over XML?
11. What is the difference between a JSONObject and a JSONArray?
12. How do you convert a JSONObject to a String in Android?

12.13. Answers (Section 12.11)

1. JSON string
2. JavaScript object
3. markup
4. exchange
5. different
6. text-based
7. lightweight
8. web services
9. independent

12.14. References/ Suggested reading

1. Headfirst Android Development, Dawn Griffiths, 1st edition, .O′Reilly
2. Android Programming for Beginners, John Horton,2nd edition, .Packt
3. Android Programming with Kotlin for Beginners, John Horton, 1st edition, Packt.
4. Android App Development, Michael Burton, 3rd edition.

NOTES