International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2020 IJSRCSEIT | Volume 6 | Issue 2 | ISSN : 2456-3307
DOI : https://doi.org/10.32628/CSEIT206249

Android XML Layouts – A study of Viewgroups and Views for UI Design in Android Application

Rahul Shah*, Kumari Chettri
Department of Information Technology, ICFAI University Sikkim, Gangtok, Sikkim, India

ABSTRACT

The main objective of this paper is to tell about the Android platform, which is designed for mobile devices, and android based mobile app development, which helps developers to develop various applications, which are now being used by the user all across the world. Android developers make use of layout with the help of which users can get a good user interface for an android application. This paper will mostly focus on the types of layouts and views that developers use to design the user interface of an android application. This paper will also discuss extensible mark-up language (XML) which is the lightweight mark-up language used for designing the layouts.
Keywords : Android, XML Layouts, Smartphones, User Interface

I. INTRODUCTION

Android, which is based on Linux kernel, is an operating system, which was developed for cameras but later shifted to mobile phones because of the low market for cameras. Initially, it was founded by Android Incorporation but later acquired by Google Inc. in 2005. Later in the year 2007, Google announced the development of an android operating system along with which they introduced OHA (Open Handset Alliance) which consists of 84 companies committed to providing smartphones using the android platform. As smartphones are becoming very popular, all of us are getting addicted to android application for all our works from ordering food, booking movie tickets, watching videos to scrolling and reading news in various application but while using these applications users expect the design of the application to be user friendly and attractive. An application, which has a good design, is quite popular as compared to the ones that have a poor user interface. This paper will discuss the design of android applications through various android XML layouts.

II. ANDROID XML LAYOUTS

The layout is used to provide a good design for your application. Android provides a variety of widgets that a programmer uses to create a desired layout and interface. Android XML layout is a file that defines the different widgets to be used in the user interface. Android treats the layout file as resources. Hence, the layout is kept in the res layout folder.

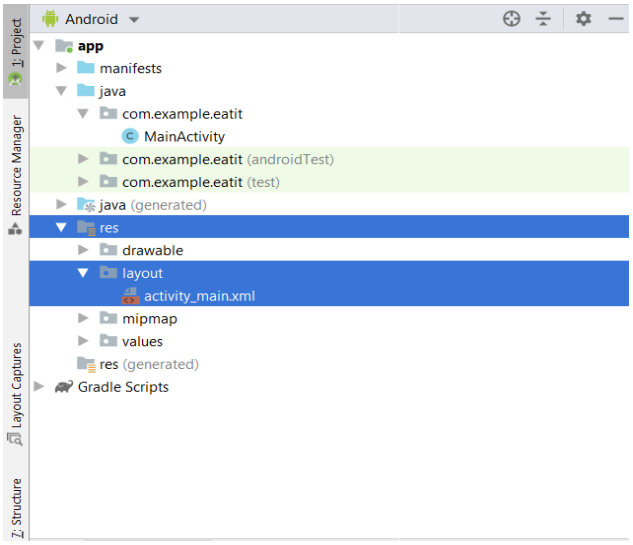

Figure 1. res – layout folder

The User Interface in Android is a hierarchy of viewgroups and views. The viewgroups will be intermediate nodes in the hierarchy, and the views will be terminal nodes. Each layout file must contain exactly one root element, once you have defined the root element, you can add additional layout objects or widgets as child elements to gradually build your layout.

For example, in the below main.xml file, the Linear Layout is a viewgroup and the TextView is a view.

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

Android provides the following standard layouts (viewgroups) that can be used in your Android application:

- Constraint Layout
- AbsoluteLayout
- FrameLayout
- LinearLayout
- RelativeLayout
- TableLayout

Now, we are going to explore each one of them in detail.

A. Constraint Layout

A Constraint Layout is a view group, which allows you to position and size widgets in a flexible way. This layout is similar to Relative Layout, but with more power. The main motive behind Constraint Layout is to provide better performance to applications. Android developer tries to build complex UI by creating deep nesting of view group so, that it will hamper the performance of an application. Due to the use of nesting view group it hard to maintain the layout. Using constraint layout, we can improve the performance of the applications by removing the nested views with a flat and flexible design and it is easier to maintain. A view inside the Constraint Layout has handles (or anchor points) on each side which are used to assign the constraints.

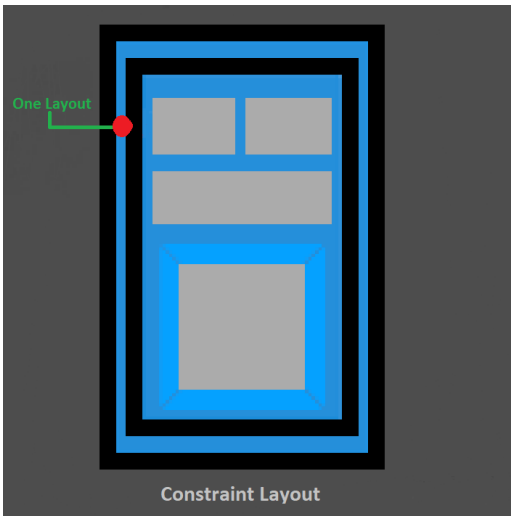

Figure 2. Constraint Layout

XML Code for Constraint Layout

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

    <TextView
      android:id="@+id/textview1"
```

```
        android:layout_width ="wrap_content"
        android:layout_height ="wrap_content"
        android:text="Hello "/>

    <TextView
        android:id ="@+id/textview2"
        android:layout_width ="wrap_content"
        android:layout_height ="wrap_content"
    app:layout_constraintLeft_toRightOf
    ="@id/textview1"
        android:text ="hello"/>
</android.support.constraint.ConstraintLayout>
</ConstraintLayout>
```

Here are some important attributes specific to Constraint Layout they are as follows:

TABLE I

ATTRIBUTES OF CONSTRAINT LAYOUT

SL. No	Attributes	Description
1	android:layout_constraintTop_toTopOf	It aligns the **top** of the desired view to the **top** of another.
2	android:layout_constraintTop_toBottomOf	It aligns the **top** of the desired view to the **bottom** of another.
3	android:layout_constraintBottom_toTopOf	It aligns the **bottom** of the desired view to the **top** of another.
4	android:layout_constraintBottom_toBottomOf	It aligns the **bottom** of the desired view to the **bottom** of another.
5	android:layout_constraintLeft_toTopOf	It aligns the **left** of the desired view to the **top** of another.
5	android:layout_constraintLeft_toBottomOf	It aligns the **left** of the desired view to the **bottom** of another.
6	android:layout_constraintLeft_toLeftOf	It aligns the **left** of the desired view to the **left** of another.
7	android:layout_constraintLeft_toRightOf	It aligns the **left** of the desired view to the **right** of another.
8	android:layout_constraintRight_toTopOf	It aligns the **right** of the desired view to the **top** of another.
9	android:layout_constraintRight_toBottomOf	It aligns the **right** of the desired view to the **bottom** of another.
10	android:layout_constraintRight_toLeftOf	It aligns the **right** of the desired view to the **left** of another.
11	android:layout_constraintRight_toRightOf	It aligns the **right** of the desired view to the **right** of another.

A. Absolute Layout

It enables you to specify the exact location of its children. It is used when we reposition the views and when there is a change in the screen while rotation. Absolute layout is depreciated as we had to set the exact location of each component based on x(top) and y(left) coordinates and that positioning is very difficult as android has various screen sizes.

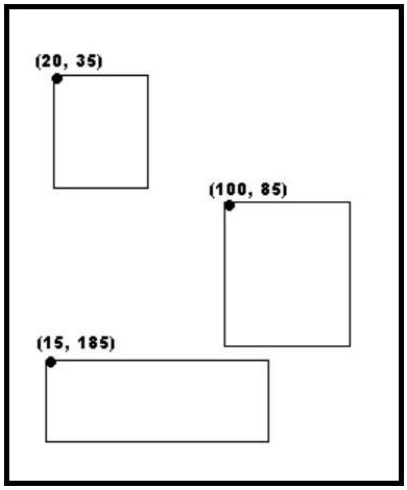

Figure 3. Absolute Layout

XML Code for Absolute Layout

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

Here are some important attributes specific to Absolute Layout they are as follows:

TABLE II

ATTRIBUTES OF ABSOLUTE LAYOUT

SL. No	Attributes	Descriptions
1	android:id	This is the id, which uniquely identifies the view.
2	android:layout_x	This specifies the x-coordinate of the view.
3	android:layout_y	This specifies the y-coordinate of the view.

B. Frame Layout

It is a placeholder on a screen that can be used to display a single view. They are used to block an area so that the child view can be adjusted in it. We can add multiple views to a Frame Layout.

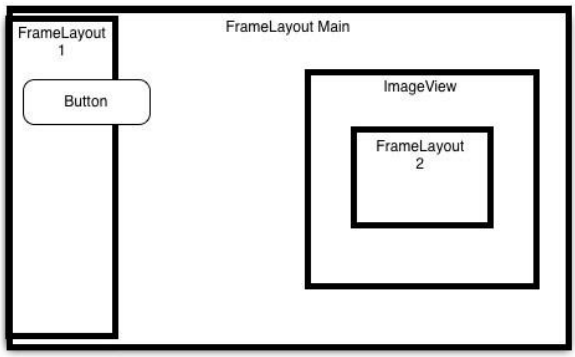

Figure 4. Frame Layout

XML Code for Frame Layout

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

<ImageView
   android:layout_width="250px"
   android:layout_height="250px"
   android:src="@drawable/ic_launcher"
   android:layout_height="fill_parent"/>

</FrameLayout>
```

Here are some important attributes specific to Frame Layout they are as follows:

TABLE III

ATTRIBUTES OF FRAME LAYOUT

SL. No	Attributes	Descriptions
1	android.id	This is the id, which uniquely identifies the view.
2	android:foreground	This defines the drawable to draw over the content and possible values may be a color value.
3	android:foregroundGravity	Defines the gravity to apply to the foreground drawable. The gravity defaults to fill. Possible values are top, bottom, left, right, center, center_vertical, center_horizontal etc.
4	android:visibility	This is used to make the view visible, invisible or gone.

5	android:measureAllChildren	Determines whether to measure all children or just those in the visible or invisible state when measuring defaults to false.

C. Linear Layout

A layout that organizes all children in a single row either horizontally or vertically. If the length of the window exceeds the length of the screen. It will create a scrollable window in our view.

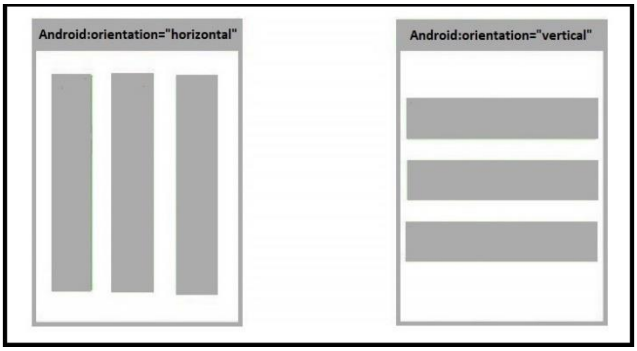

Figure 5. Linear Layout

XML Code for Linear Layout

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

</Linear Layout>
```

Here are some important attributes specific to Linear Layout they are as follows:

TABLE IV
ATTRIBUTES OF LINEAR LAYOUT

SL. No	Attributes	Descriptions
1	android:id	This is the id, which uniquely identifies the view.
2	android: divider	This is used as a vertical divider between the buttons.
3	android:baselineAligned	This must be a Boolean value, either “true” or “false” and prevents the layout from aligning its children’s baselines.
4	android:baselineAlignedChildIndex	When a linear layout is part of another layout that is baseline aligned, it can specify which of its children to baseline align.
5	android:gravity	This specifies how child views are positioned.
6	android:orientation	This specifies the direction of arrangement and you will use horizontal for a row and vertical for a column. The default is horizontal.
7	android:layout_weight	This specifies value to view in terms of how much space it should occupy on screen.
8	android:weight-Sum	Sum of the child weight.

D. Relative Layout

It displays all child views in relative positions and is considered as the most flexible layout as it allows positioning the component anywhere we want. The position of each view can be specified as relative to sibling elements or relative to the parent. It is considered as the second most used layout after Linear Layout.

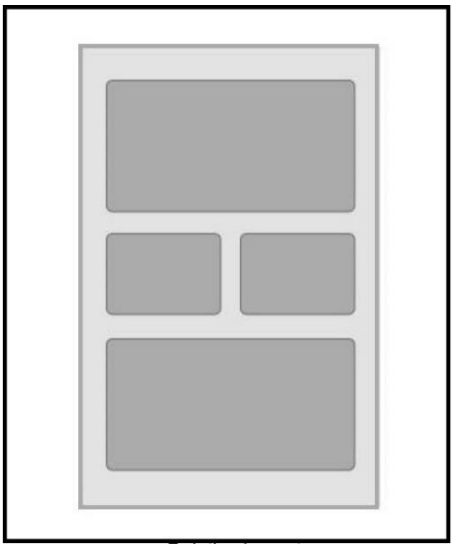

Figure 6. Relative Layout

XML Code for Relative Layout

```
xml version="1.0" encoding="utf-8"?>
<Relative Layout
xmlns:android="http://schemas.android.com/apk/res/
android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:paddingLeft="16dp"
android:paddingRight="16dp”>
</Relative Layout>
```

Here are some important attributes specific to Relative Layout they are as follows:

TABLE V

ATTRIBUTES OF RELATIVE LAYOUT

SL. No	Attributes	Descriptions
1	android.id	This is the id that uniquely identifies the view.
2	android:gravity	This specifies how child views are positioned.
3	android:ignoreGravity	This indicates what view should not be affected by gravity
4	android:layout_above	It is used to position the view above another view. ID must be used as a reference.
	android:layout_below	It is used to position the view below another view. ID must be used as a reference.
	android:layout_toRightOf	It is used to position the view to the right of another view. ID must be used as a reference.
	android:layout_toLeftOf	It is used to position the view to the left of another view. ID must be used as a reference.
5	android:layout_alignTop	It is used to position the view to match the top edge of another

		view. ID must be used as a reference.
	android:layout_alignBottom	It is used to position the view to match the bottom edge of another view. ID must be used as a reference.
	android:layout_alignLeft	It is used to position the view to match the left edge of another view. ID must be used as a reference.
	android:layout_alignRight	It is used to position the view to match the right edge of another view. ID must be used as a reference.
6	android:layout_centerInParent	It is used to make the view aligned to the center of the parent. It should be specified as “true”.
	android:layout_centerHorizontal	It is used to make the view horizontally aligned to the center of the parent. It should be specified as “true”.
	android:layout_centerVertical	It is used to make the view vertically aligned to the center of the parent. It should be specified as “true”.
7	android:layout_alignParentTop="true"	It is used to make a view stick to the top of its parent.
	android:layout_alignParent Bottom="true"	It is used to make a view stick to the bottom of its parent.
	android:layout_alignParent Left="true"	It is used to make a view stick to the left of its parent.
	android:layout_alignParent Right="true"	It is used to make a view stick to the right of its parent.

E. Table Layout

A layout that group views into rows and columns. Each row has zero or more cells, each cell can hold one view object like Text View, Image View, Buttons etc. <Table Row> element is used to create a row in a table and its width is determined by its parent container.

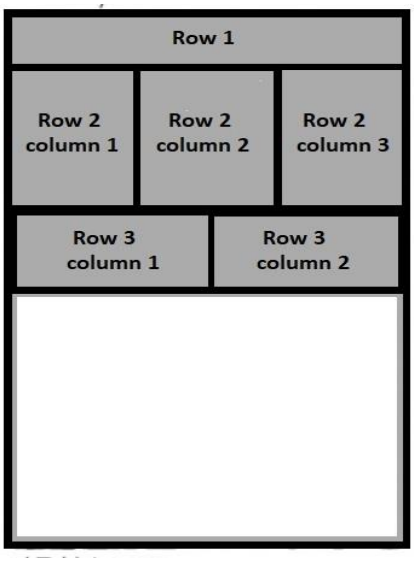

Figure 7. Table Layout

XML Code for Table Layout

```
xml version="1.0" encoding="utf-8"?>
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/
android"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
   <TableRow
      android:layout_width="fill_parent"
      android:layout_height="fill_parent">
   <TextView
      android:text="Hello Table Layout"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent">
   </TableRow>

</TableLayout>
```

Here are some important attributes specific to Tables Layout they are as follows:

TABLE VI

ATTRIBUTES OF TABLE LAYOUT

SL. No	Attributes	Descriptions
1	android.id	This is the id which uniquely identifies the view.
2	android:collapseColumns	This specifies the zero-based index of the columns to collapse. The column indices must be separated by a comma.
3	android:shrinkColumns	The zero-based index of the columns to shrink. The column indices must be separated by a comma.
4	Android:stretchColumns	The zero-based index of the columns to stretch. The column indices must be separated by a comma.

III. LAYOUT ATTRIBUTES

Every type of layout has attributes that define the way its elements appear. There are both common attributes that all layouts share, and attributes specific to some of the layout types.

The following are attributes that apply to all layouts:

TABLE VII

ANDROID LAYOUT ATTRIBUTES

SL. No	Attributes	Description
1	android:id	This is the ID, which uniquely identifies the view.
2	android:layout_width	This is the width of the layout. (required for every view)
3	android:layout_height	This is the height of the layout. (required for every view
4	android:layout_margin	This is the extra space outside of the view. For example, if you give android:marginLeft= 10dp, then the view will be arranged after 10dp from left
5	android:layout_marginBottom	Extra space on the bottom of the layout.
6	android:layout_marginLeft	Extra space to the left of the layout.
7	android:layout_marginRight	Extra space to the right of the layout.
8	android:layout_weight	Specifies how much of the extra space in the layout should be allocated to the view.
9	android:layout_padding	This is similar to android:layout_margin except that it specifies the extra space inside the view
10	android:paddingLeft	Padding to the left of the view.
11	android:paddingRight	Padding to the right of the view.
12	android:paddingTop	Padding at the top of the view.
12	android:paddingBottom	Padding at the bottom of the view.
14	android:layout_gravity	This specifies how child Views are positioned
15	android:layout_x	This specifies the x-coordinate of the layout
16	android:layout_y	This specifies the y-coordinate of the layout

IV.VIEWS USED FOR ANDROID APP DESIGN

The Android applications are a combination of Viewgroups and View. A View in android helps in designing and creating an attractive design for an android application. It is used to create components like TextView, Edit Text, Radio Button, etc. It helps in taking input from the user and in return, gives output to the user.

Android provides the following views that can be used in your Android application.

- Text View
- EditText

- Button
- ImageView
- ImageButton
- CheckBox
- Radio button
- RadioGroup
- ListView
- Spinner
- AutoCompleteTextView

Now, we are going to explore each one of them in detail.

A. Text View

It is used to display text on our application screen. It also allows the user to edit text programmatically.

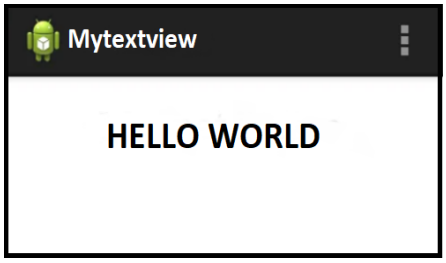

Figure 8. Text View

XML Code for Text View

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

   <TextView
   android:id="@+id/mytxtview"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:padding="12dp"
   android:layout_margin="20dp"
   android:text="HELLO WORLD"
   android:textSize="20dp"
   android:textColor="#0000000"
   android:gravity="center"/>

   </LinearLayout>
```

B. Edit Text View

It makes text to be editable in an application. It helps in building the data interface taken from any user, also contains certain features through which we can hide the confidential data. E.g. password, CVV number, etc.

Figure 9. Edit Text View

XML Code for Edit Text View

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

<EditText
android:id="@+id/myEdittext"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:padding="10dp"
android:layout_margin="20dp"
```

```
android:textSize="20dp"
android:textStyle="bold"
android:gravity="center"
android:hint="Enter a Number"
android:singleLine="true"
android:inputType="textPassword" />

</LinearLayout>
```

C. Button View

It is used to perform event handling on button click. Android provides different types of button such as Radio Button, Toggle Button, Compound Button, etc.

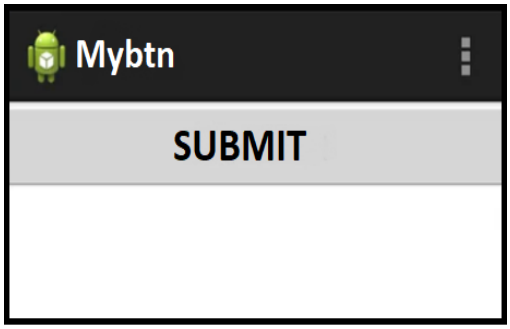

Figure 10. Button View

XML Code for Button View

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

<Button
android:id="@+id/button1"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="SUBMIT />

</LinearLayout>
```

D. Image View

Image view helps to display an image in an android application. We just have to paste an image in a drawable folder from where we can access it.

Figure 11. Image View

XML Code for Image View

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

<ImageView
android:id="@+id/myimageview"
android:layout_width="110dp"
android:layout_height="110dp"
android:layout_margin="20dp"
android:padding="10dp"
android:gravity="center"
```

```
android:src="@drawable/ic_launcher" />

</LinearLayout>
```

E. Image Button View

It works as a button but the only difference is that it has an image in it. It has both features as we can display an image with a button. When users click on the image at that time, a certain event will occur.

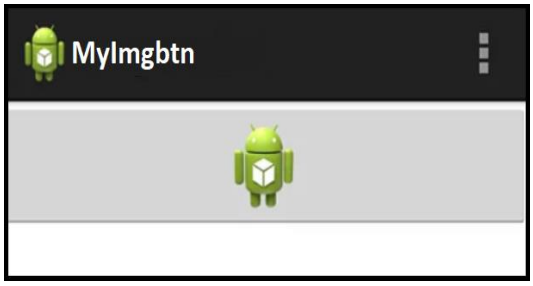

Figure 12. Image Button View

XML Code for Image Button 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" >

<ImageButton
android:id="@+id/MyimageBtn"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:src="@drawable/ic_launcher" />

</LinearLayout>
```

F. Checkbox View

It is either a type of two state button checked or unchecked.it can be used to check multiple things at a one time. For example, hobbies. Android Checkbox class is the subclass of the Compound Button class.

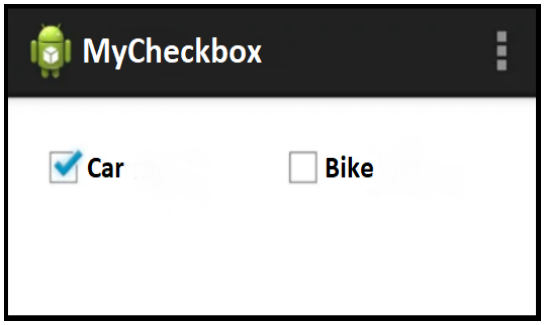

Figure 13. Checkbox View

XML Code for Checkbox View

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

<CheckBox
android:id="@+id/MycheckBox1"
android:layout_width="110dp"
android:layout_height="wrap_content"
android:layout_margin="20dp"
android:text="Car"
android:checked="true" />

<CheckBox
android:id="@+id/MycheckBox2"
android:layout_width="100dp"
```

```
android:layout_height="wrap_content"
android:layout_margin="20dp"
android:text="Bike" />

</LinearLayout>
```

G. Radio Button View

Radio button is like a checkbox, but there is a slight difference between them. In the Radio button, we can select only one option out of them e.g. Gender whereas in checkbox we can select more than two options at one time. E.g. hobbies.

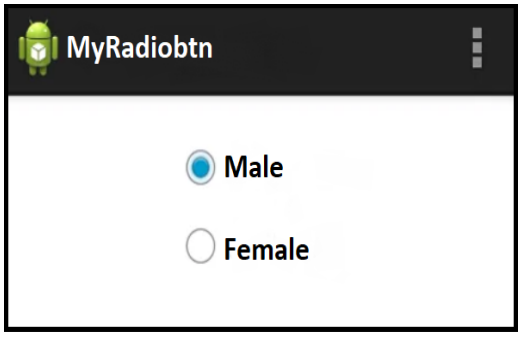

Figure 14. Radio Button View

XML Code for Radio Button View

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

<RadioButton
android:id="@+id/MyradioButton1"
android:layout_width="100dp"
android:layout_height="wrap_content"
android:layout_margin="20dp"
android:text="Male"
android:checked="true" />

<RadioButton
android:id="@+id/MyradioButton2"
android:layout_width="100dp"
android:layout_height="wrap_content"
android:layout_margin="20dp"
android:text="Female" />

</LinearLayout>
```

H. Radio Group View

It is used to group Radio Buttons in Android. If we check one radio button that belongs to a radio group, it automatically unchecks any previously checked radio button within the same group.

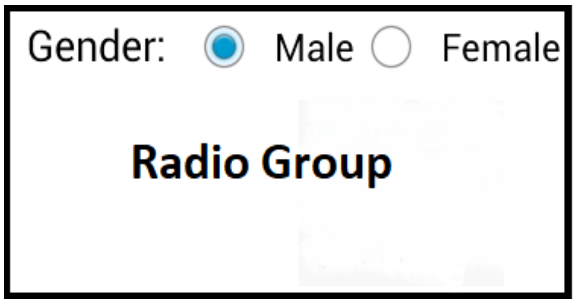

Figure 15. Radio Group View

XML Code for Radio Group View

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

```
<RadioGroup
    android:id="@+id/radioGroup"
    android:layout_width="fill_parent"
    android:layout_height="90dp">

 <RadioButton
     android:layout_width="wrap_content"
     android:layout_height="55dp"
     android:text="Male"
     android:id="@+id/radioButton"
     android:layout_gravity="center_horizontal"
     android:checked="false"
     android:textSize="25dp" />

  <RadioButton
     android:id="@+id/radioButton2"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Female"
     android:layout_gravity="center_horizontal"
     android:checked="false"
     android:textSize="25dp" />
  </RadioGroup>

</RealtiveLayout>
```

I. List View

It is a view, which groups several items and displays in a scrollable list. In the list view, we do not have to mention the scroll view because the list view is by default scrollable. List View uses Adapter classes, which add the content from the data source (such as string array, array, database, etc.) to List View.

Figure 16. List View

XML Code for List View

```
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=".MainActivity">

   <ListView
      android:id="@+id/MylistView"
      android:layout_width="match_parent"
      android:layout_height="fill_parent" />

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

J. Spinner View

Spinner view displays multiple options, but only one can be selected at a time. Android spinner is associated with Adapter View. Therefore, you need to use one of

the adapter classes with spinner. Spinner class is the subclass of the AsbSpinner class.

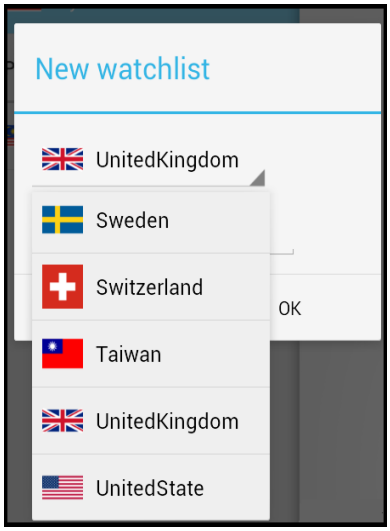

Figure 17. Spinner View

XML Code for Spinner View

```
xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android=http://schemas.android.com/apk/res/a
ndroid
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical"
android:gravity="center_horizontal" >
    <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dip"
    android:text=Select :"
    android:layout_marginBottom="5dp"/>

  <Spinner
    android:id="@+id/spinner"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:prompt="@string/spinner_title"/>
</LinearLayout>
```

K. AutoComplete Text View

Android AutoComplete Text View is an editable text field, it displays a list of suggestions in a drop down menu from which user can select only one suggestion or value. Android AutoComplete Text View is the subclass of the Edit Text class. The Multi AutoComplete Text View is the subclass of the AutoComplete Text View class.

Figure 18. AutoComplete Text View

XML Code for Auto Complete Text View

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

 <AutoCompleteTextView
    android:id="@+id/autoCompleteTextView"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:layout_marginLeft="92dp"
    android:layout_marginTop="144dp"
    android:text=" "/>

 <RelativeLayout/>
```

V. XML AND IMPORTANCE OF XML BASED LAYOUT

XML stands for Extensible mark-up language. It is designed to store and transport data. It encodes documents in a format that is both human and machine readable. It doesn't depend on any software or hardware as it is platform and programming language independent.

XML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
<to>Kumari Chettri</to>
<from>Rahul Shah</from>
<msg>Welcome to Techcoders.in</msg>
</message>
```

Now let us see some benefits of XML-Based Layouts:

- XML is used widely and quite popular so many developers feel comfortable using it.
- Android has a drag and drop UI tools so generating code for XML is easier than writing the entire code.
- Android provides separate UI for XML, which provides flexibility to change one code without affecting the other one.
- XML-based layouts are very helpful if you know the UI components at the time of compiling. If run-time UI components are needed, then those can be added using the XML code.

VI. CONCLUSION

Android is a truly open, free development platform based on Linux operating system for mobile devices. The Past few years android is growing rapidly because of Its User friendly UI, Beautiful Design and many more features. Android Smartphone is in the hype in the 21st century. The scope of android applications is increasing day by day and its development and design have become an essential part of today's programming curriculum. It provides various types of layouts and widgets to design dynamic UI for the android application. The study shows details reading the use of various layouts along with their attributes and also tells how these layouts have their way of providing an exceptional user interface to the user. XML based layouts and its UI elements are helping developers to design their application more appropriately.

VII. REFERENCES

[1] Wei-Meng Lee, "Beginning Android Application Development", Wiley Publishing, Inc. Pg. - 81 – 95 and 126 – 166.

[2] Prof. Rajkumar A. Soni, "A STUDY PAPER ON ANDROID UI", International Journal of Enterprise Computing and Business Systems, Volume 2 Issue 1 January 2013.

[3] Android Tutorial, https://www.javatpoint.com/android-tutorial.

[4] Know Your Layouts in Android, https://www.sitepoint.com/know-your-layouts-in-android.

[5] Android Developers official website, https://developer.android.com/guide/topics/ui/declaring-layout.

[6] Android Layout Tutorial, https://www.learnhowtoprogram.com/android/introduction-to-android/introduction-to-xml-and-android-layouts.

[7] Constraint Layout, https://medium.com/exploring-android/exploring-the-new-android-constraintlayout-eed37fe8d8f1

[8] Android User Interface, https://www.tutorialspoint.com/android/android_user_interface_layouts.htm.

AUTHORS PROFILE

Mr. Rahul Shah has received his Master of Computer Application (MCA) degree from Shri Ramaswamy Memorial University, Sikkim in 2017. He is currently working as a Lecturer at ICFAI University, Sikkim. He has 3+ years of teaching experience for both undergraduate and postgraduate students. His current field of interest includes Android application development, Artificial Intelligence, Cloud computing, and Web Engineering.

Ms. Kumari Chettri has received her Bachelor of Computer Application (BCA) degree from ICFAI University, Sikkim in 2018. She is currently pursuing her Masters of Computer Application (MCA) from ICFAI University, Sikkim. She has worked as a Trainee at SIBIN (Department of Information Technology). Her areas of expertise include Cloud Computing, Android application development, and DBMS.

Cite this article as:

Rahul Shah, Kumari Chettri, "Android XML Layouts - A study of Viewgroups and Views for UI Design in Android Application", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 6 Issue 2, pp. 184-200, March-April 2020. Available at doi : https://doi.org/10.32628/CSEIT206249
Journal URL : http://ijsrcseit.com/CSEIT206249