Theory Assignment – 01

B.Sc.(H) CS, Spring 2022

4th Semester, Android Programming

SUBMITTED BY

Name: Nishant Kumar Giri
Roll No: AC-1254

2022

Department of Computer Science

ACHARYA NARENDRA DEV COLLEGE

Q1. Write the important features (at least 5) of each Android OS versions form Cupcake to Android 12.

1. **Android 1.5: Cupcake**

   Released on April 30, 2009.

   Features:

   - On-screen keyboard and search function
   - Uploading videos and images
   - Copy and paste facility and video recordings
   - Support for MPEG4 and 3GP formats

2. **Android 1.6: Donut**

   Released on September 15, 2009

   Features:

   - Power Control widget for handling Wi-Fi, Bluetooth, GPS, etc.
   - Gallery and Camera quick toggling features
   - WVGA screen resolution speed
   - Technology support for CDMA/EVDO, 802.1x, VPNs, and a text-to-speech engine
   - Speed improvements for camera and searching applications
   - Quick Search Box

3. **Android 2.0: Eclair**

Released on December 3, 2009

Features:

- Update UI
- Support Bluetooth 2.1
- Improve Google map
- Minor API Changes
- Support Live and animated Wallpaper
- Ability to add contact's photo and select to call, message or email

4. **Android 2.2: Froyo**

Released on May 20, 2010

Features:

- Support for Animated GIF and multiple keyboard languages
- Speed and performance improvements
- Upload file support in the browser
- Support numeric & alphanumeric passwords to enhance security
- Increased Compatibility with car kits and headsets
- Wi-Fi Support Hotspot functionality

5. **Android 2.3: Gingerbread**

Released on December 6, 2010

Features:

- Improve Copy and Paste Facility
- Updated UI design
- Support for VP8 and WebM video format
- Video calling and Social Networking Supports
- Easy to use a keyboard with faster and intuitive typing

6. **Android 3.0: Honeycomb**

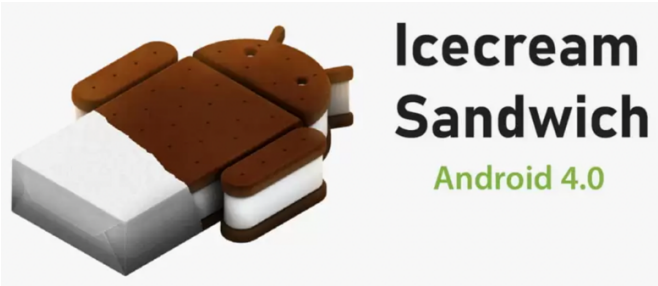
Released on February 22, 2011

Features:

- Gmail, contacts, camera and gallery improvements
- Support for passwords with complex characters
- Encrypted storage and updated 3D UI
- Supports multiprocessors and recent apps for easy visual multitasking
- Media Sync from SD Card
- Action bar for application control
- System bar for global status and notifications

7. **Android 4.0: Icecream Sandwich**

Released on October 18, 2011

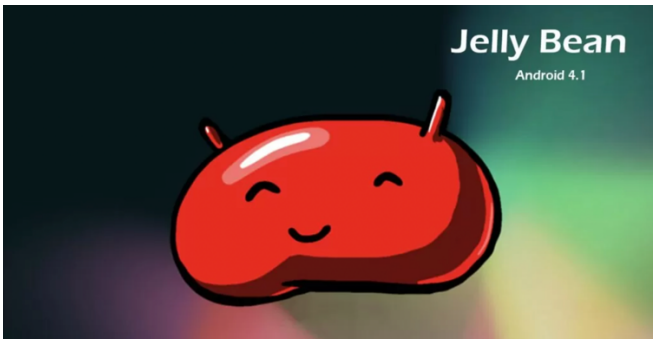


Features:

- Spelling check feature
- Wi-Fi direct
- Photo Decor facility and on-screen buttons
- Unlocking with face-fixing.
- Card-like appearance for app-switching
- Improved video recording with high resolution
- Better Camera performance
- Ability to open up to 16 tabs in the web browser

8. **Android 4.1: Jelly Bean**

Released on July 9, 2012



Features:

- Voice search and typing
- Panorama
- Project Butter
- Expandable notifications
- Daydream as a screensaver
- Power control
- Support USB audio
- Improved camera app
- Security improvements

9. Android 4.4: Kitkat

Released on October 31, 2013

Features:

- Screen Recording
- Contact Prioritization
- GPS Support
- Smarter Caller ID
- Offline music support
- UI updates for alarm and google map navigations
- Cartoonish ideograms and emojis to the Google keyboard
- KitKat has 'OK Google' feature that allows access to Google to the users without touching your smartphones.

10. Android 5.0: Lollipop

Released on November 12, 2014

Features:

- Support ART
- Better device protection
- Notifications can be flicked away from the lock screen
- Better and improved UI
- Built-in battery saver feature
- New material design
- Revamped navigation bar

11. Android 6.0: Marshmallow

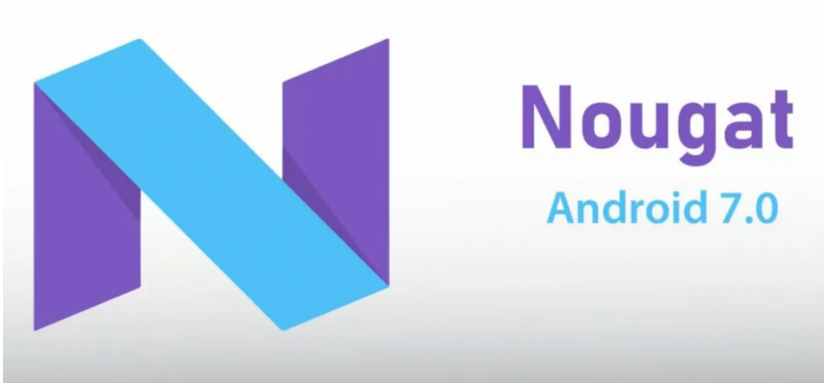
Released on October 5, 2015

Features:

- Support for Fingerprint readers
- Type C USB support
- Multi-window experience
- 'Sleep Mode' for saving battery life
- Clear permission system
- Custom google tabs and improved Copy-pasting

12. Android 7.0: Nougat

Released on August 22, 2016

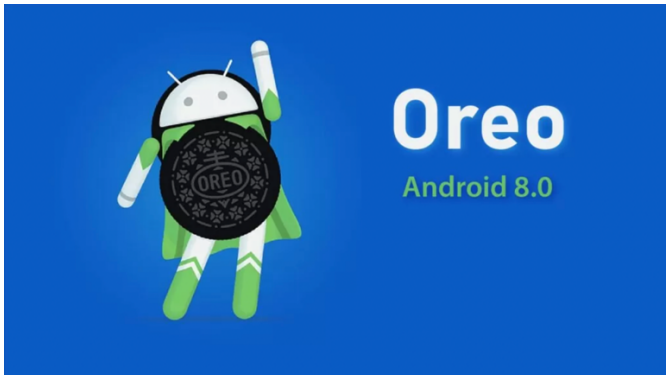


Features:

- Provide multitasking and split-screen mode
- Storage manager enhancements
- Quick setting toggles
- Display touch enhancements
- Better setting application
- Inline reply to messages and notifications without opening applications

13. Android 8.0: Oreo

Released on August 21, 2017



Features:

- Password autofill
- Auto-enable Wi-Fi
- Downloadable fonts
- Multi-display support
- Support Picture-in-Picture
- Notification channels and snooze notification
- Google Play support and new emoji styling
- Adaptive icons and smart text selection

14. Android 9.0: Pie

Released on August 6, 2018

Features:

- Sound amplifier with select to speak options
- Artificial intelligence (AI) compatibility
- Adaptive Battery and Brightness with background restrictions
- Multi-camera support with the external camera compatibility
- New Gesture Navigation and App Actions
- New Screenshot Shortcut key and accessibility menu
- Easier Screen Rotation and edge to edge screens support
- Volume and Sound enhancements
- Selectable Dark Mode

15. Android 10.0: Android Q

Released on September 3, 2019

Features:

- Support for foldable smartphones with flexible displays
- Dark mode for eyes comfortability
- Navigation control over gesture quicker and intuitive ever
- Sound amplifier with more clear sound
- Smart reply suggestions for all messaging apps
- Live caption for media playing on a smartphone
- Undo app removal
- Better notification control with many options

16. Android 11.0: Android R

Released on September 8, 2020

Features:

- Airplane mode doesn’t kill Bluetooth anymore
- Face Unlock will require you to open your eyes in pixel 4
- App pinning in the share menu
- Native screen recording
- Muting notifications during video
- Increase touch sensitivity
- Notification History
- Auto revoke app permissions

17. Android 12.0: Android 12 (Snow Cone)

Released on October 4, 2021

Features:

- Colour Extraction
- Enhanced Notifications
- Wi-Fi Sharing
- Handy Tiles / Quick tools
- Motion and Animations

Features for Developers:

- Media Transcoding
- Multi-channel Audio
- Improved Performance
- Big Display Optimization
- AVIF Image
- Debugging Toggleable Changes
- Foreground Services

Q2. Differentiate among the following:

a) Android OS vs iOS

Parameters	iOS	Android
Developers	Apple Incorporation developed the iOS operating system, and it is also the owner.	Open Handset Alliance and Google developed the Android OS, and Google LLC owns it.
Kernel Type	iOS has a hybrid kernel type.	Android has a Linux- based kernel type.
Language Used	assembly language, Objective-C, Swift, C, and C++ languages.	We mainly write Android in Java, C, C++, and various other languages.
Default Internet Browser	The devices that are iOS-based have Safari as their default browser for the Internet.	The devices that are Android-based have Chrome as their default browser for the Internet.
Preferred License	iOS has a preferred license of GNU GPL, APSL, and Proprietary.	Android has a preferred license of GNU GPLv2 and Apache.
Languages Available (in Devices)	The iOS devices are available in the market in about 34 languages.	The Android devices in the market are available in about 100+ languages.
Voice Assistant	Siri is the Voice Assistant for iOS.	Google Assistance is the Voice Assistant for Google.

b) Android OS vs Windows OS

Parameters	Android	Windows
Developer	Android was developed and owned by Google LLC.	Windows was developed and owned by Microsoft Inc.
Kernel Type	In Android, the Kernel type is Linux-based.	In Windows, the Kernel type is Hybrid with modules here.
Language Used	Android OS uses Java, C, C++ languages.	Windows OS uses C, C++ languages.
Root Support	All Android OS are supported for root access.	Currently, any windows phone OS does not support root access.
Cost	Android apps are free of cost.	Windows apps are charged for the original version.
Supported Platforms	Android OS supports ARM, MIPS, x86, IMX.	Windows OS only supports x86.

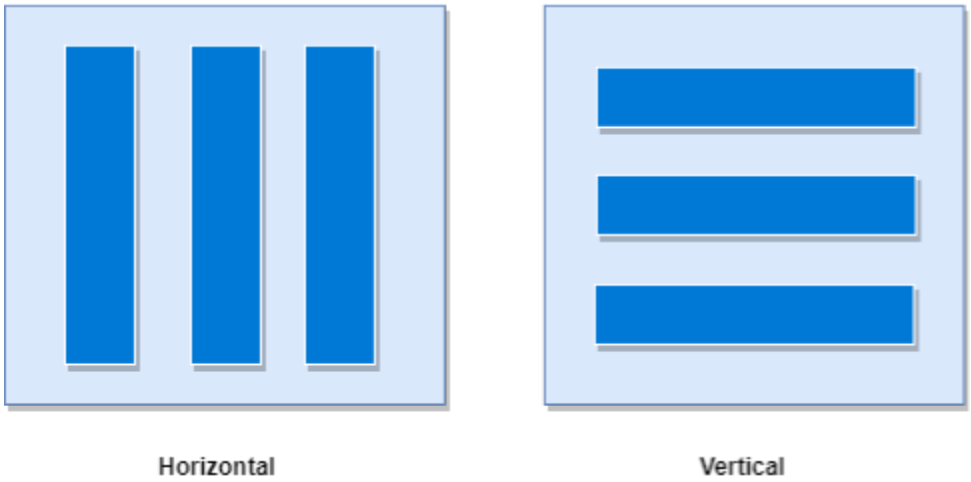
Q3. Explain in detail different layouts (Linear, Constraint, Grid, Relative, Table, Frame, List, etc.) available in Android OS with suitable examples (pictorial representation of layout and its implementation).

1. Linear Layout

In android, **LinearLayout** is a **ViewGroup** subclass which is used to render all child **View** instances one by one either in **Horizontal** direction or **Vertical** direction based on the **orientation** property.

In android, we can specify the linear layout orientation using **android:orientation** attribute.

Following is the pictorial representation of linear layout in android applications.



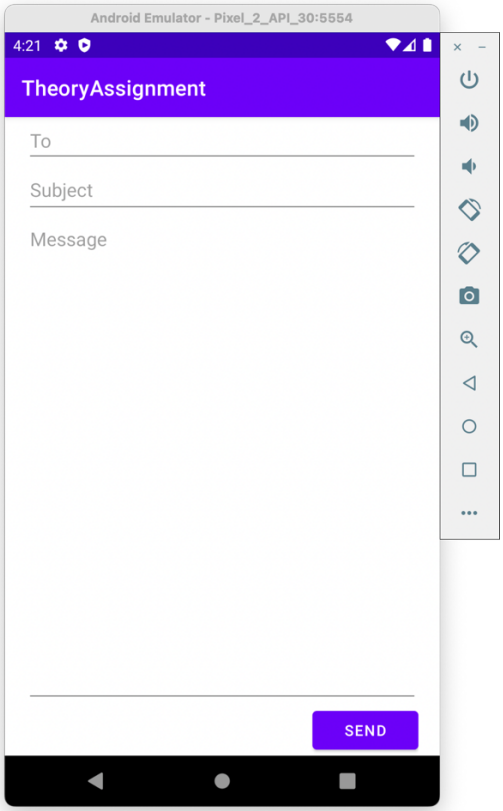
In **LinearLayout**, the child **View** instances arranged one by one, so the horizontal list will have only one row of multiple columns and vertical list will have one column of multiple rows.

Implementation:

```
<?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" >
    <!-- Add Child Views Here -->
</LinearLayout>
```

Linear Layout

```xml
<?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"
    android:paddingLeft="20dp"
    android:paddingRight="20dp">

    <EditText
        android:id="@+id/txtTo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="To" />

    <EditText
        android:id="@+id/txtSub"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Subject" />

    <EditText
        android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="Message" />

    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="Send" />
</LinearLayout>
```



2. Relative Layout

This layout is for specifying the position of the elements in relation to the other elements that are present there.

In the relative layout, alignment of the position of the elements to the parent container is possible. To define it in such a way, we write the following:

- android:layout_alignParentTop= “true”
- android:layout_alignParentLeft= “true”

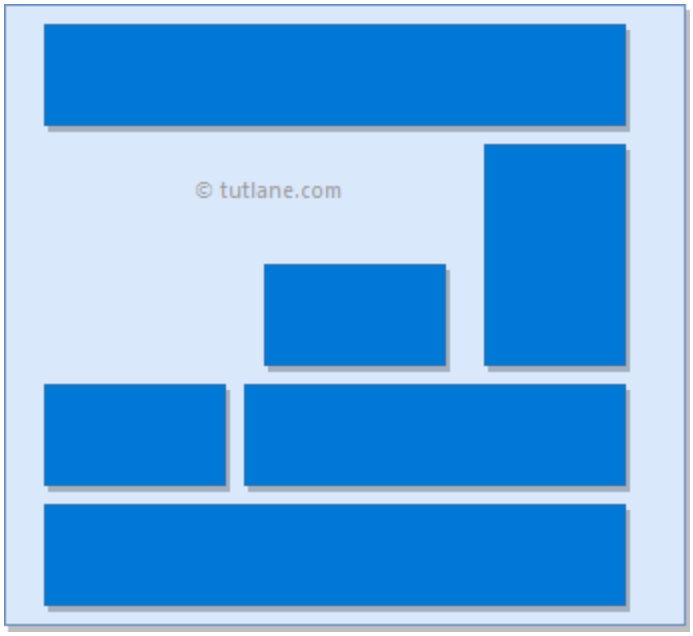
If we write the above code, the element will get aligned on the top left of the parent container.

If we want to align it with some other element in the same container, it can be defined is as follows:

- android:layout_alignLeft= “@+id/element_name”
- android:layout_below= “@+id/element_name”

This will align the element below the other element to its left.

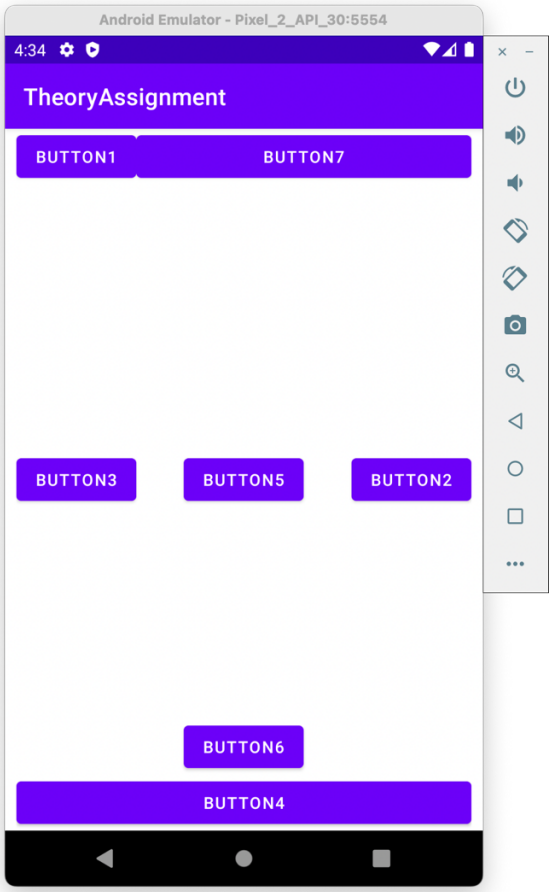
Following is the pictorial representation of relative layout in android applications.



Implementation:

Relative Layout

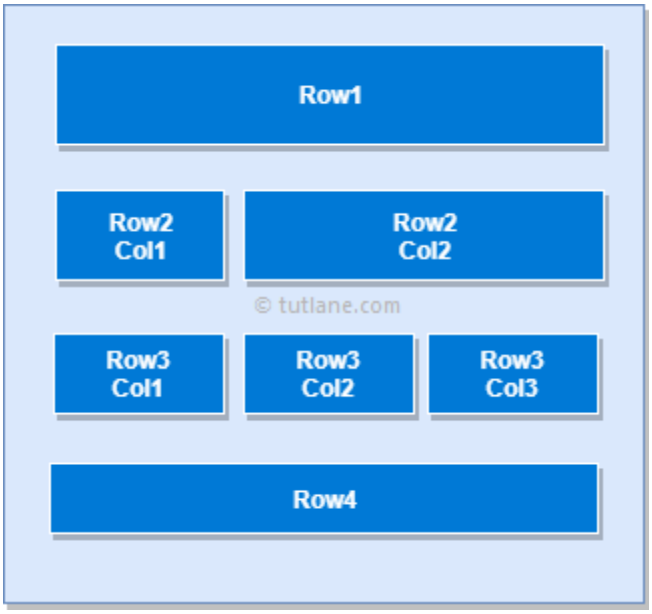
```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="10dp"
    android:paddingRight="10dp">

    <Button
        android:id="@+id/btn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:text="Button1" />

    <Button
        android:id="@+id/btn2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
        android:text="Button2" />

    <Button
        android:id="@+id/btn3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_centerVertical="true"
        android:text="Button3" />

    <Button
        android:id="@+id/btn4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:text="Button4" />

    <Button
        android:id="@+id/btn5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/btn2"
        android:layout_centerHorizontal="true"
        android:text="Button5" />

    <Button
        android:id="@+id/btn6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/btn4"
        android:layout_centerHorizontal="true"
        android:text="Button6" />

    <Button
        android:id="@+id/btn7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_toEndOf="@+id/btn1"
        android:layout_toRightOf="@+id/btn1"
        android:text="Button7" />
</RelativeLayout>
```



3. Table Layout

n android, **TableLayout** is a **ViewGroup** subclass that is used to display the child View elements in rows and columns.

Following is the pictorial representation of relative layout in android applications.



In android, TableLayout will position its children elements into rows and columns and it won't display any border lines for rows, columns or cells.
The TableLayout in android will work same as the HTML table and the table will have as many columns as the row with the most cells. The TableLayout can be explained as **<table>** and TableRow is like **<tr>** element.

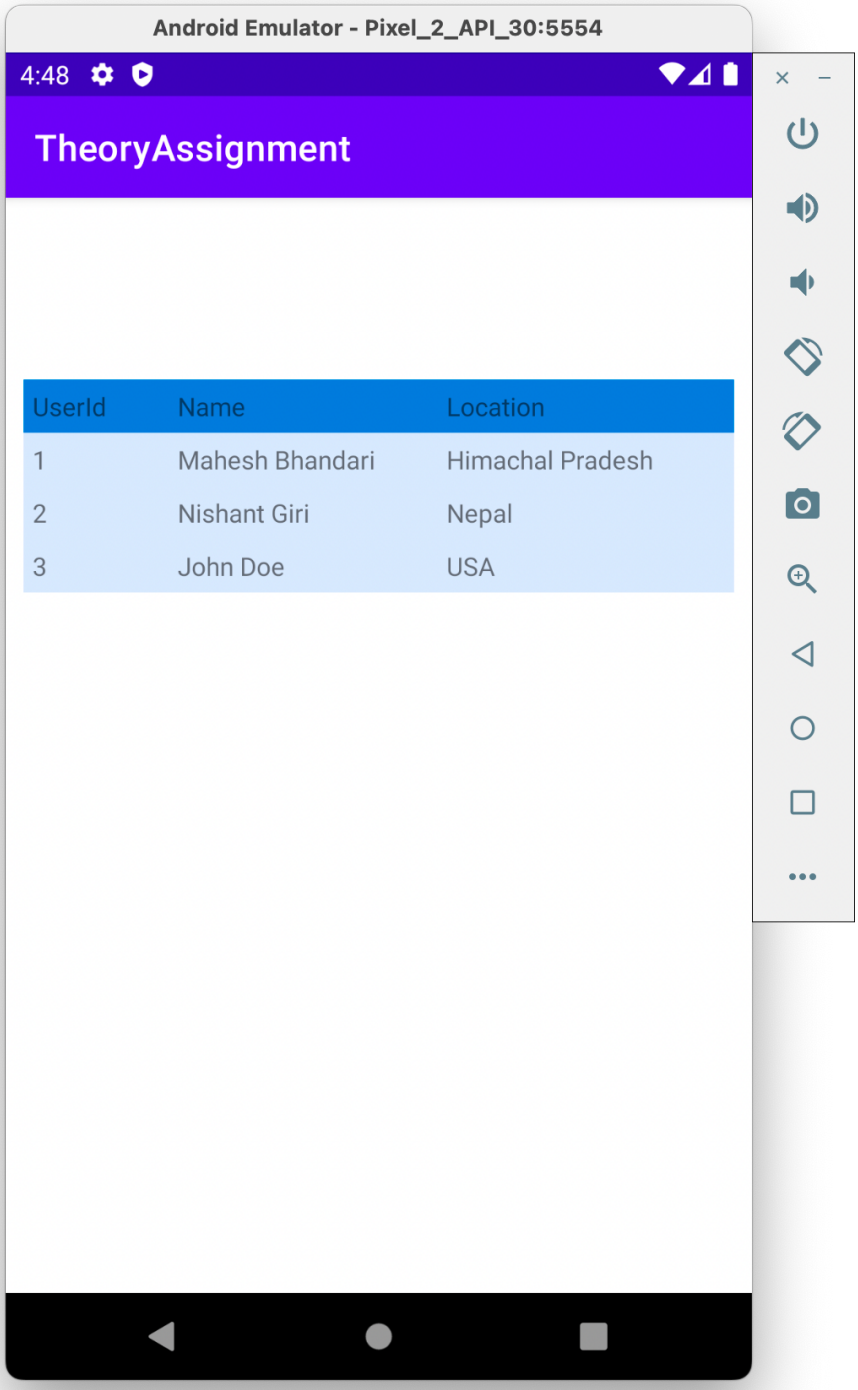
Implementation:

Table Layout

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="100dp"
    android:paddingLeft="10dp"
    android:paddingRight="10dp">

    <TableRow
        android:background="#0079D6"
        android:padding="5dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="UserId" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Name" />

        <TextView
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Location" />
    </TableRow>

    <TableRow
        android:background="#DAE8FC"
        android:padding="5dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="1" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Mahesh Bhandari" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Himachal Pradesh" />
    </TableRow>

    <TableRow
        android:background="#DAE8FC"
        android:padding="5dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="2" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Nishant Giri" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Nepal" />
    </TableRow>

    <TableRow
        android:background="#DAE8FC"
        android:padding="5dp">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="3" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="John Doe" />
```

```
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="USA" />
    </TableRow>
</TableLayout>
```



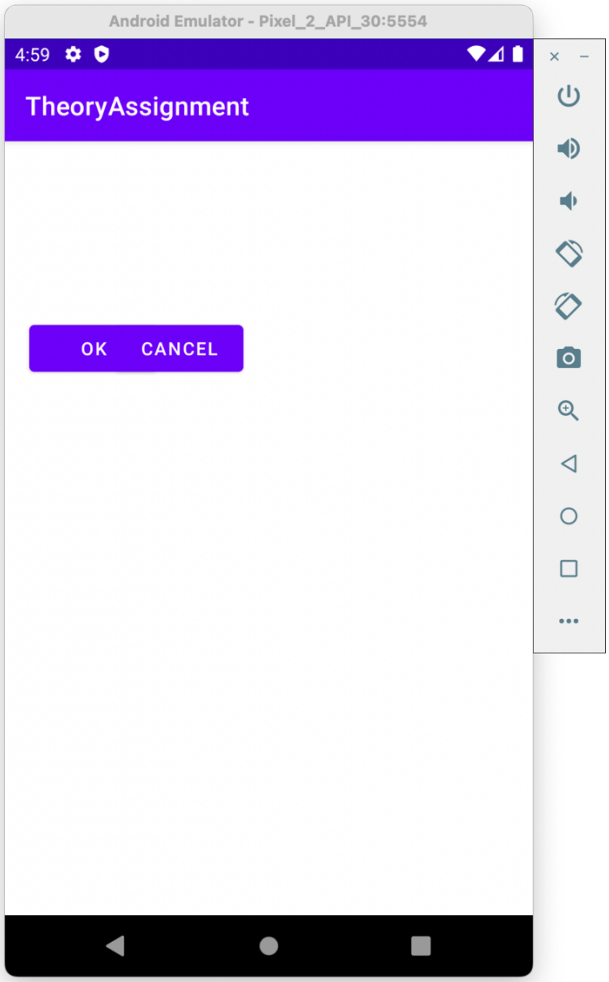
4. Absolute Layout

An **Absolute Layout** allows you to specify the exact location .i.e., X and Y coordinates of its children with respect to the origin at the top left corner of the layout. The absolute layout is less flexible and harder to maintain for varying sizes of screens that's why it is not recommended.

Implementation:

Absolute Layout

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:text="OK"
        android:layout_x="50px"
        android:layout_y="361px" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:text="Cancel"
        android:layout_x="225px"
        android:layout_y="361px" />
</AbsoluteLayout>
```



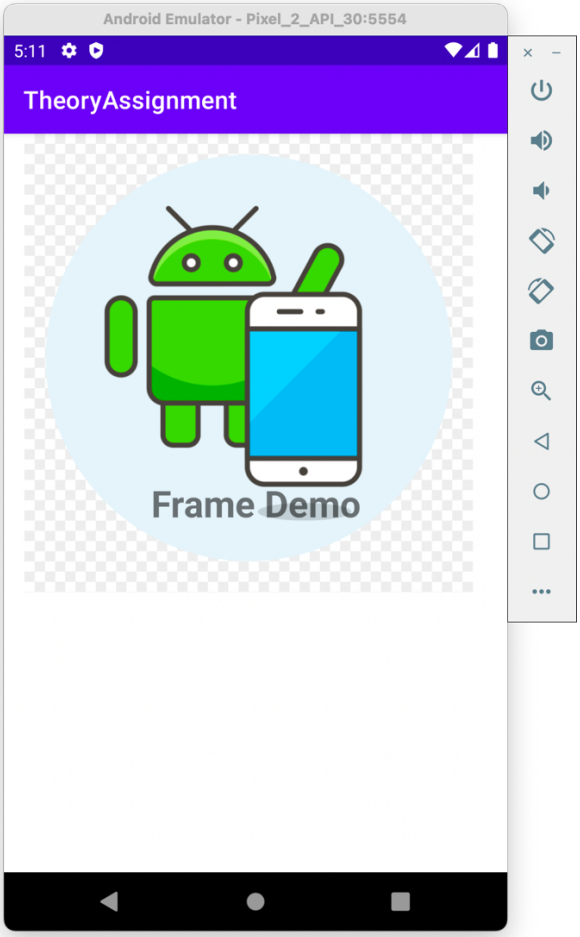
5. Frame Layout

Frame Layout is designed to block out an area on the screen to display a single item. Generally, Frame Layout should be used to hold a single child view, because it can be difficult to organize child views in a way that's scalable to different screen sizes without the children overlapping each other.

Implementation:

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="400dp"
        android:layout_height="375dp"
        android:scaleType="fitCenter"
        android:src="@drawable/ic_launcher" />

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:gravity="center"
        android:text="Frame Demo"
        android:textSize="30dp"
        android:textStyle="bold" />
</FrameLayout>
```



6. List View Layout

Android **ListView** is a view which groups several items and display them in vertical scrollable list. The list items are automatically inserted to the list using an **Adapter** that pulls content from a source such as an array or database.

Implementation:

List View Layout

MainActivity.java

```
package com.example.theoryassignment;

import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    ListView listView;
    TextView textView;
    String[] listItem;

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

        listView = (ListView) findViewById(R.id.listView);
        textView = (TextView) findViewById(R.id.textView);
        listItem = getResources().getStringArray(R.array.array_technology);
        final ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
                android.R.layout.simple_list_item_1, android.R.id.text1, listItem);
        listView.setAdapter(adapter);

        listView.setOnItemClickListener((adapterView, view, position, l) -> {
            // TODO Auto-generated method stub
            String value = adapter.getItem(position);
            Toast.makeText(getApplicationContext(), value, Toast.LENGTH_SHORT).show();

        });
    }
}
```

activity_main.xml

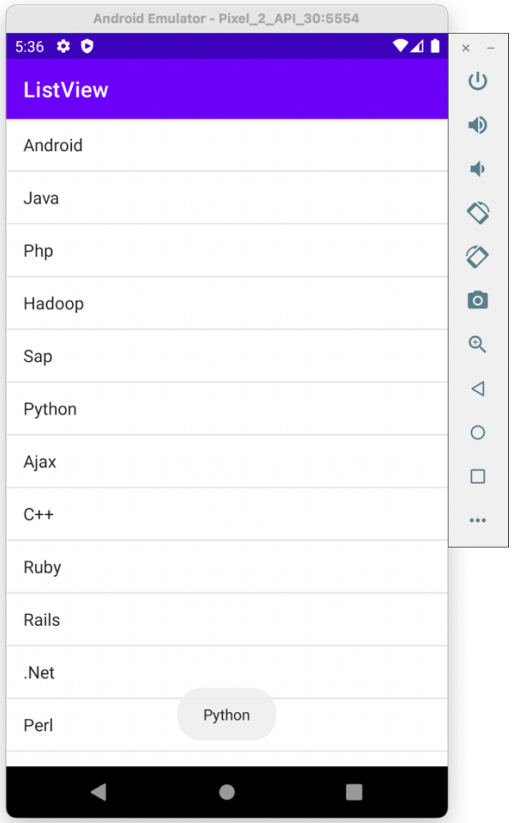
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.theoryassignment.MainActivity">

    <ListView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="fill_parent" />
</LinearLayout>
```

strings.xml

```
<resources>
    <string name="app_name">ListView</string>
    <string-array name="array_technology">
        <item>Android</item>
        <item>Java</item>
        <item>Php</item>
        <item>Hadoop</item>
        <item>Sap</item>
        <item>Python</item>
        <item>Ajax</item>
        <item>C++</item>
        <item>Ruby</item>
        <item>Rails</item>
        <item>.Net</item>
        <item>Perl</item>
    </string-array>
</resources>
```

mylist.xml

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

<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="5dp"
    android:padding="2dp"
    android:text="Medium Text"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:textColor="#4d4d4d"
    android:textStyle="bold" />
```



7. Grid View Layout

Android **Grid View** shows items in two-dimensional scrolling grid (rows & columns) and the grid items are not necessarily predetermined but they automatically inserted to the layout using a **List Adapter.**

Implementation:

Grid View Layout

ImageAdapter.java

```
package com.example.theoryassignment;

import android.content.Context;

import android.view.View;
import android.view.ViewGroup;

import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;

public class ImageAdapter extends BaseAdapter {
    private Context mContext;

    // Constructor
    public ImageAdapter(Context c) {
        mContext = c;
    }

    public int getCount() {
        return mThumbIds.length;
    }

    public Object getItem(int position) {
        return null;
    }

    public long getItemId(int position) {
        return 0;
    }

    // create a new ImageView for each item referenced by the Adapter
    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageView;
```

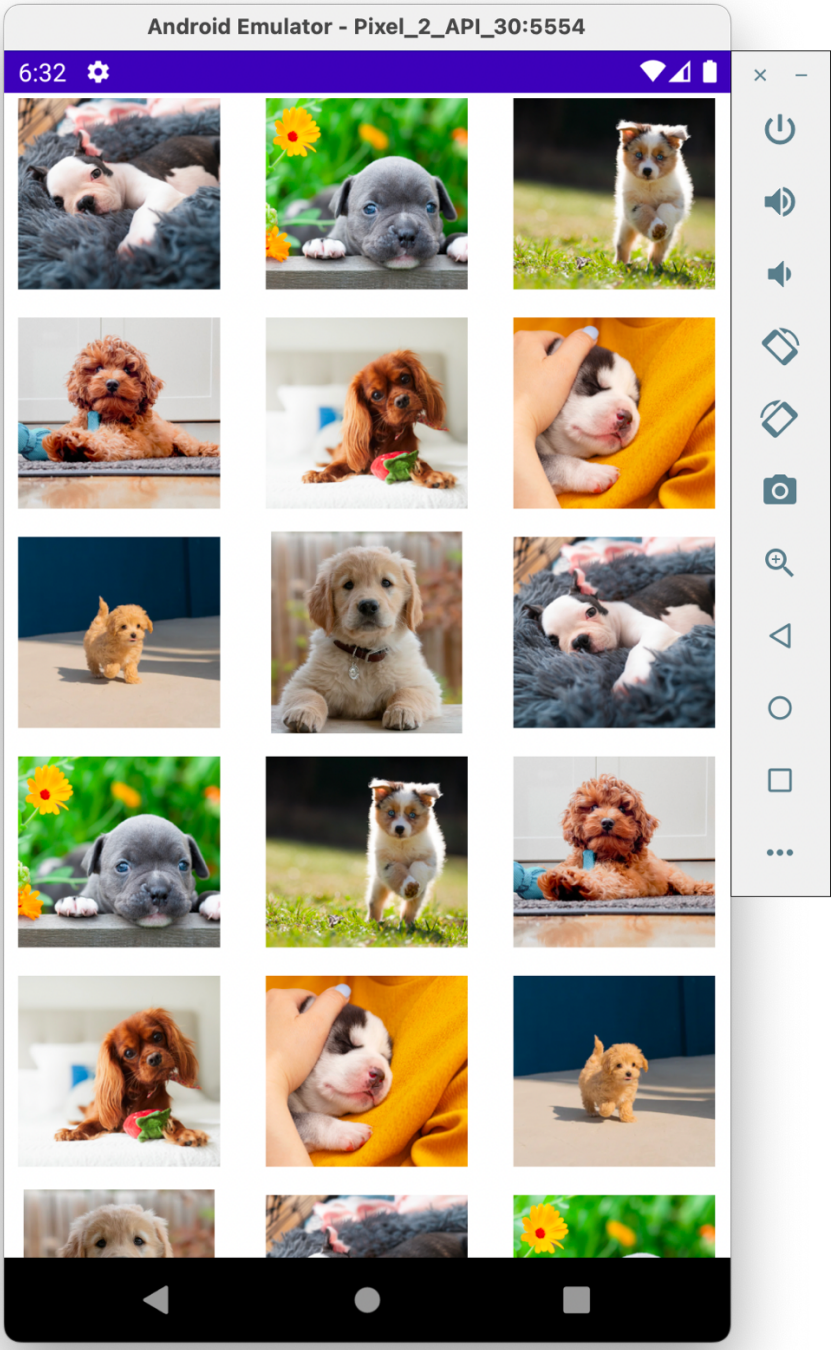
```
        if (convertView == null) {
            imageView = new ImageView(mContext);
            imageView.setLayoutParams(new GridView.LayoutParams(400, 400));
            imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
            imageView.setPadding(8, 5, 10, 10);
        }
        else
        {
            imageView = (ImageView) convertView;
        }
        imageView.setImageResource(mThumbIds[position]);
        return imageView;
    }

    // Keep all Images in array
    public Integer[] mThumbIds = {
            R.drawable.sample_2, R.drawable.sample_3,
            R.drawable.sample_4, R.drawable.sample_5,
            R.drawable.sample_6, R.drawable.sample_7,
            R.drawable.sample_0, R.drawable.sample_1,
            R.drawable.sample_2, R.drawable.sample_3,
            R.drawable.sample_4, R.drawable.sample_5,
            R.drawable.sample_6, R.drawable.sample_7,
            R.drawable.sample_0, R.drawable.sample_1,
            R.drawable.sample_2, R.drawable.sample_3,
            R.drawable.sample_4, R.drawable.sample_5,
            R.drawable.sample_6, R.drawable.sample_7
    };
}
```

MainActivity.java

```
package com.example.theoryassignment;

import android.app.Activity;
import android.os.Bundle;
import android.widget.GridView;

public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        GridView gridview = (GridView) findViewById(R.id.gridview);
        gridview.setAdapter(new ImageAdapter(this));
    }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gridview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:columnWidth="100dp"
    android:numColumns="auto_fit"
    android:verticalSpacing="10dp"
    android:horizontalSpacing="10dp"
    android:stretchMode="columnWidth"
    android:gravity="center"
    />
```


Android Emulator - Pixel_2_API_30:5554
6:32