MANIPAL INSTITUTE OF TECHNOLOGY
Manipal – 576 104

DEPARTMENT OF INFORMATION & COMMUNICATION TECHNOLOGY

CERTIFICATE

This is to certify that Ms./Mr. ……...…… Reg.No. …..…… Section: …… Roll No:..... has satisfactorily completed the lab exercises prescribed for **Mobile Application Development Lab** [**ICT 3268**] of Third Year B. Tech. CCE Degree at MIT, Manipal, in the academic year 20_ _- 20_ _.

Date: …….....

Signature of the faculty

CONTENTS

Course Objectives

- To design and develop mobile applications using android.
- To familiarize with mobile UI design.
- To learn Database Connectivity to mobile applications.
- To be able to develop an application using advanced android concepts

Course Outcomes

At the end of this course, students will be able to

- Design and develop mobile applications for real world scenerio.
- Design and Implement mobile UI design.
- Demonstrate database connectivity for mobile applications.
- Implement wireless functionalities such as Bluetooth, etc in android applications.

Evaluation plan

Split up of 60 marks for Regular Lab Evaluation
Record :8 marks 2 regular evaluations will be carried . Each evaluation is for 12 marks.:2X12=24 marks Midterm:20 Marks Internal Project Demo: 8 marks Total Internal Marks: 8+24+20+8 =60 Marks
End Semester Lab evaluation: 20 marks (Duration 2 hrs)
Program write up: 6 Marks Program execution: 14 Marks Total: 6+4 =20 Marks
Mini project evaluation: 20 marks

INSTRUCTIONS TO THE STUDENTS

Pre- Lab Session Instructions

1. Students should carry the Lab Manual Book and the required stationery to every lab session
2. Be in time and follow the institution dress code
3. Must sign in the log register provided
4. Make sure to occupy the allotted seat and answer the attendance
5. Adhere to the rules and maintain the decorum

In- Lab Session Instructions

- Follow the instructions on the allotted exercises
- Show the program and results to the instructors on completion of experiments
- On receiving approval from the instructor, copy the program and results in the lab record
- Prescribed textbooks and class notes can be kept ready for reference if required

General Instructions for the exercises in Lab

- Implement the given exercise individually and not in a group.
- The programs should meet the following criteria:
    - Programs should be interactive with appropriate prompt messages, error messages if any, and descriptive messages for outputs.
    - Comments should be used to give the statement of the problem.
    - Statements within the program should be properly indented.
- Plagiarism (copying from others) is strictly prohibited and would invite severe penalty in evaluation.
- In case a student misses a lab, he/ she must ensure that the experiment is completed before the next evaluation with the permission of the faculty concerned.
- Students missing out lab on genuine reasons like conference, sports or activities assigned by the Department or Institute will have to take **prior permission** from the HOD to attend **additional lab**(with other batch) and complete it **before** the student goes on leave. The student could be awarded marks for the write up for that day provided he submits it during the **immediate** next lab.

- Students who fall sick should get permission from the HOD for evaluating the lab records. However attendance will not be given for that lab.
- Students will be evaluated only by the faculty with whom they are registered even though they carry out additional experiments in other batch.
- Presence of the student during the lab end semester exams is mandatory even if the student assumes he has scored enough to pass the examination
- Minimum attendance of 75% is mandatory to write the final exam.
- If the student loses his book, he/she will have to rewrite all the lab details in the lab record.
- Questions for lab tests and examination are not necessarily limited to the questions in the manual, but may involve some variations and / or combinations of the questions.

THE STUDENTS SHOULD NOT

- Bring mobile phones or any other electronic gadgets to the lab.
- Go out of the lab without permission.

LAB NO: 1 **Date:**

BASICS OF ANDROID MOBILE APPLICATION DEVELOPMENT TOOL

Objectives

- To familiarize with mobile application development tool.
- To gain knowledge about how to develop simple mobile application using android features.

What is Android?

Android is an open source and Linux-based **Operating System** for mobile devices such as smartphones and tablet computers. Android was developed by the *Open Handset Alliance*, led by Google, and other companies.

Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android.

The first beta version of the Android Software Development Kit SDK was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008.

On June 27, 2012, at the Google I/O conference, Google announced the next Android version, 4.1 **Jelly Bean**. Jelly Bean is an incremental update, with the primary aim of improving the user interface, both in terms of functionality and performance.

The source code for Android is available under free and open source software licenses. Google publishes most of the code under the Apache License version 2.0 and the rest, Linux kernel changes, under the GNU General Public License version 2.

Android Applications

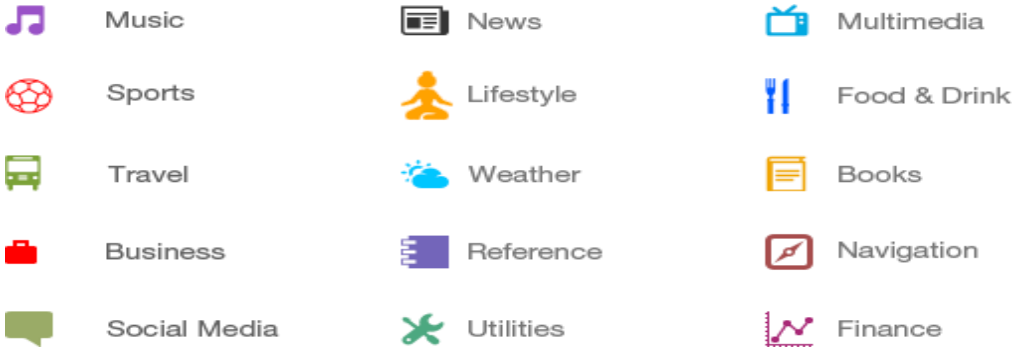
Android applications are usually developed in the Java language using the Android Software Development Kit.

Once developed, Android applications can be packaged easily and sold out either through a store such as **Google Play**, **SlideME**, **Opera Mobile Store**, **Mobango**, **F-droid** and the **Amazon Appstore**.

Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast. Every day more than 1 million new Android devices are activated worldwide.

Categories of Android applications

There are many android applications in the market. The top categories are −



What is API level?

API Level is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform.

Android Architecture

Android architecture contains a different number of components to support any Android device's needs. Android software contains an open-source Linux Kernel having a collection of a number of C/C++ libraries which are exposed through application framework services. Among all the components Linux Kernel provides the main functionality of operating system functions to smartphones and Dalvik Virtual Machine (DVM) provide a platform for running an Android application. linux kernel.

Components of Android Architecture

The main components of Android architecture are the following:-

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

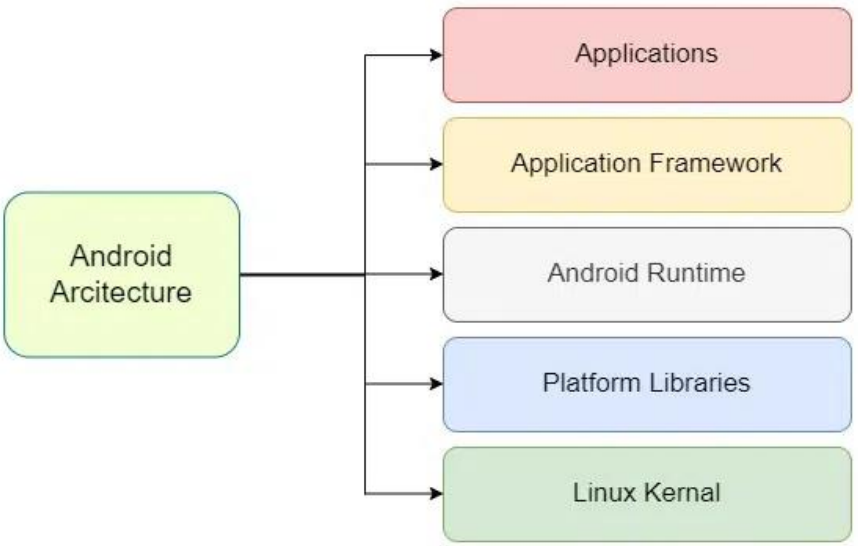
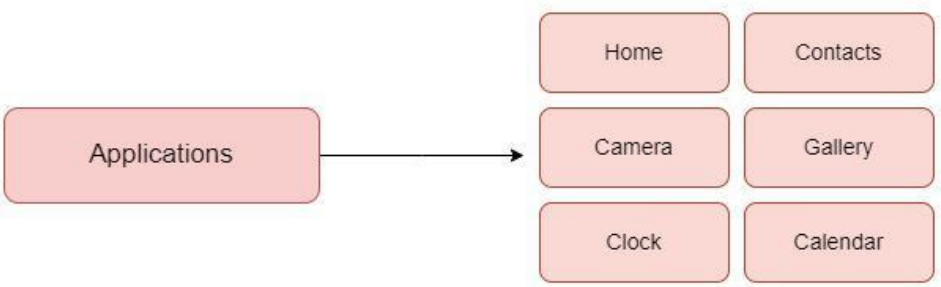


Figure 1: Pictorial representation of Android architecture with several main components and their sub-components(src: Android developers).

Understanding Android's architecture is essential for building efficient applications.

1. Applications

Applications is the top layer of android architecture. The pre-installed applications like home, contacts, camera, gallery etc and third party applications downloaded from the play store like chat applications, games etc. will be installed on this layer only. It runs within the Android run time with the help of the classes and services provided by the application framework.



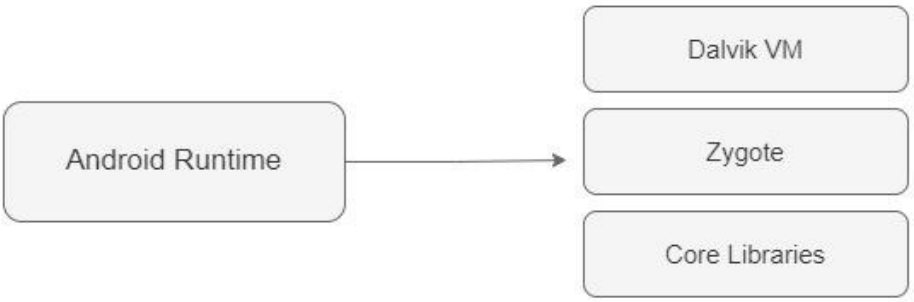
2. Application framework

Application Framework provides several important classes which are used to create an Android application. It provides a generic abstraction for hardware access and also helps in managing the user interface with application resources. Generally, it provides the services with the help of which we can create a particular class and make that class helpful for the Applications creation. It includes different types of services activity manager, notification manager, view system, package manager etc. which are helpful for the development of our application according to the prerequisite.

3. Application runtime

Android Runtime environment is one of the most important part of Android. It contains components like core libraries and the Dalvik virtual machine(DVM). Mainly, it

provides the base for the application framework and powers our application with the help of the core libraries. Like Java Virtual Machine (JVM), **Dalvik Virtual Machine (DVM)** is a register-based virtual machine and specially designed and optimized for android to ensure that a device can run multiple instances efficiently. It depends on the layer Linux kernel for threading and low-level memory management. The core libraries enable us to implement android applications using the standard JAVA or Kotlin programming languages.



Note: Now, starting from Android 5.0 and above, we use **ART (Android Runtime)** to compile bytecode into native code to leverage ahead-of-time compilation.

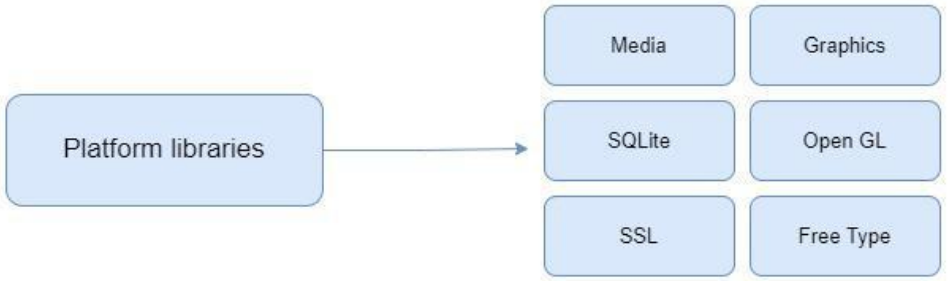
4. Platform libraries

The Platform Libraries includes various C/C++ core libraries and Java based libraries such as Media, Graphics, Surface Manager, OpenGL etc. to provide a support for android development.

- **Media** library provides support to play and record an audio and video formats.
- **Surface manager** responsible for managing access to the display subsystem.
- **SGL** and **OpenGL** both cross-language, cross-platform application program interface (API) are used for 2D and 3D computer graphics.
- **SQLite** provides database support and **FreeType** provides font support.
- **Web-Kit** This open source web browser engine provides all the functionality to

display web content and to simplify page loading.

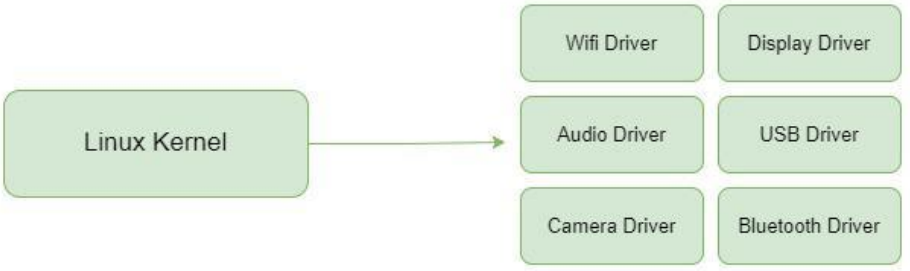
- **SSL (Secure Sockets Layer)** is security technology to establish an encrypted link between a web server and a web browser.



5. Linux Kernel

Linux Kernel is heart of the android architecture. It manages all the available drivers such as display drivers, camera drivers, Bluetooth drivers, audio drivers, memory drivers, etc. which are required during the runtime. The Linux Kernel will provide an abstraction layer between the device hardware and the other components of android architecture. It is responsible for management of memory, power, devices etc. The features of Linux kernel are:

- **Security:** The Linux kernel handles the security between the application and the system.
- **Memory Management:** It efficiently handles the memory management thereby providing the freedom to develop our apps.
- **Process Management:** It manages the process well, allocates resources to processes whenever they need them.
- **Network Stack:** It effectively handles the network communication.
- **Driver Model:** It ensures that the application works properly on the device and hardware manufacturers responsible for building their drivers into the Linux build.



Project structure:

Each project in Android Studio contains one or more modules with source code files and resource files. The types of modules include:

- Android app modules
- Library modules
- Google App Engine modules

By default, Android Studio displays your project files in the Android project view, as shown in figure 1. This view is organized by modules to provide quick access to your project's key source files. All the build files are visible at the top level, under **Gradle Scripts**.

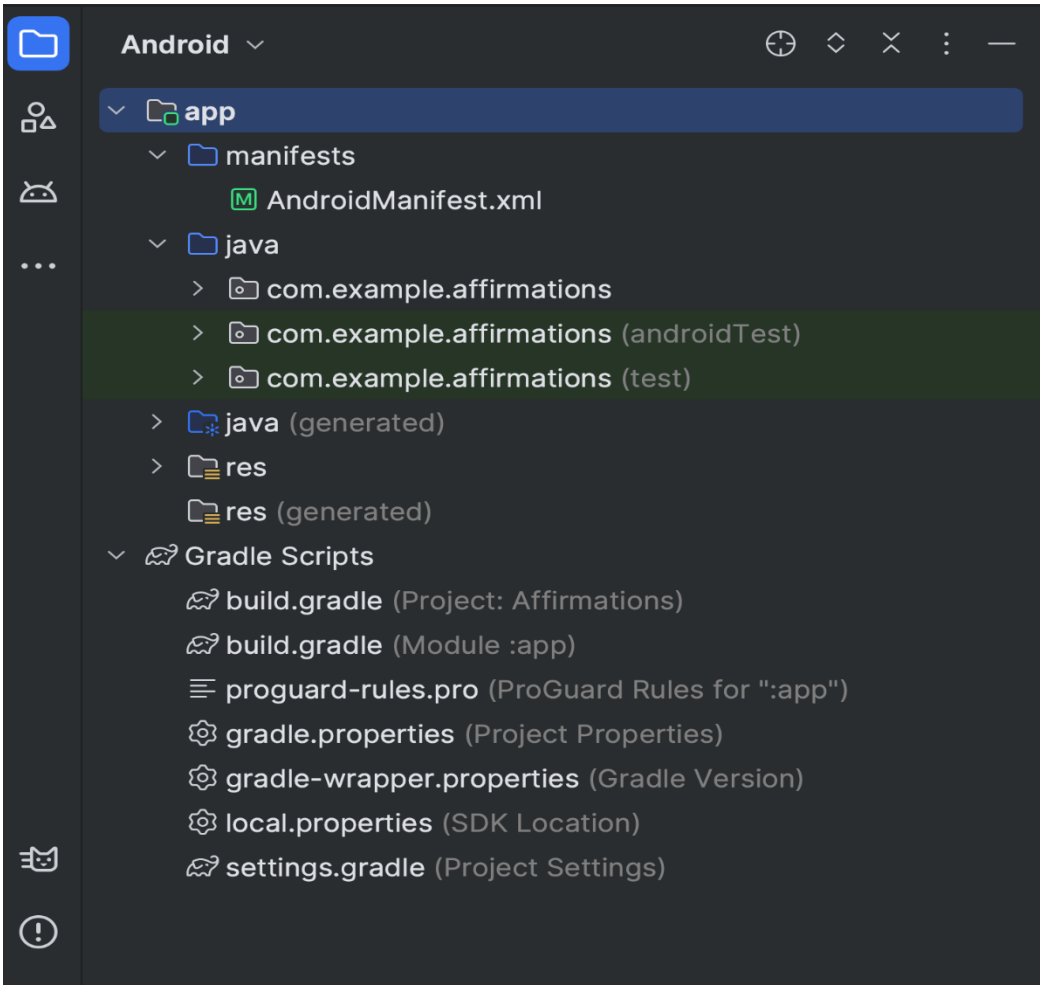


Figure 2. Project files in Android project view.

Each app module contains the following folders:

- **manifests**: Contains the AndroidManifest.xml file.
- **java**: Contains the Kotlin and Java source code files, including JUnit test code.
- **res**: Contains all non-code resources such as UI strings and bitmap images.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select **Project** instead of **Android** from the **Project** menu.

Gradle build system

Android Studio uses Gradle as the foundation of the build system, with more Android-specific capabilities provided by the Android Gradle plugin. This build system runs as an integrated tool from the Android Studio menu and independently from the command line. You can use the features of the build system to do the following:

- Customize, configure, and extend the build process.
- Create multiple APKs for your app with different features, using the same project and modules.
- Reuse code and resources across source sets.

By employing the flexibility of Gradle, you can achieve all of this without modifying your app's core source files.

Android Studio build files are named build.gradle.kts if you use Kotlin (recommended) or build.gradle if you use Groovy. They are plain text files that use the Kotlin or Groovy syntax to configure the build with elements provided by the Android Gradle plugin. Each project has one top-level build file for the entire project and separate module-level build files for each module. When you import an existing project, Android Studio automatically generates the necessary build files.

Manage dependencies

Dependencies for your project are specified by name in the module-level build script. Gradle finds dependencies and makes them available in your build. You can declare module dependencies, remote binary dependencies, and local binary dependencies in your build.gradle.kts file.

Android Studio configures projects to use the Maven Central Repository by default. This configuration is included in the top-level build file for the project.

Debug and profile tools

Android Studio helps you debug and improve the performance of your code,

including inline debugging and performance analysis tools.

Inline debugging

Use inline debugging to enhance your code walkthroughs in the debugger view with inline verification of references, expressions, and variable values.

Inline debug information includes:

- Inline variable values
- Objects that reference a selected object
- Method return values
- Lambda and operator expressions
- Tooltip values

To enable inline debugging, in the **Debug** window, click **Settings** and select **Show Variable Values in Editor**.

Performance profilers

Android Studio provides performance profilers so you can easily track your app's memory and CPU usage, find deallocated objects, locate memory leaks, optimize graphics performance, and analyze network requests.

To use performance profilers, with your app running on a device or emulator, open the Android Profiler by selecting **View > Tool Windows > Profiler**.

Data file access

The Android SDK tools, such as Systrace and Logcat, generate performance and debugging data for detailed app analysis.

To view the available generated data files:

1. Open the Captures tool window.
2. In the list of the generated files, double-click a file to view the data.

3. Right-click any HPROF files to convert them to the standard.

4. Investigate your RAM usage file format.

Lab exercises

Develop an Android application for the following program

1. Create an Android application to show the demo of displaying text with justifying elements, changing text colors ,fonts etc.

2. Find the “hello word” text in the XML document and modify the text.

LAB NO: 2 **Date:**

INTRODUCTION TO ACTIVITY AND LAYOUTS IN ANDROID

Objectives

- To apply the concepts of layouts to enrich the user interface:
- Understanding layout types and how to use them effectively improves the UI design, enhancing the user experience.
- To understand different activity and fragments to use in android application.

Activity

An Activity in Android represents a single screen with a user interface. It acts as the entry point for interacting with the user and is a core component of an Android application. Activities are declared in the AndroidManifest.xml file and can be linked together to create a seamless user experience.

Key Components of an Activity:

Lifecycle: Activities have a lifecycle managed by the Android system, which includes methods such as:

- **onCreate():** Initializes the activity and sets up the UI.
- **onStart(), onResume(), onPause(), and onStop():** Handle the transition states of the activity.
- **onDestroy():** Final cleanup before the activity is destroyed.

Intent Handling: Activities can start other activities or services using intents.

Layouts

Layouts define the structure and appearance of the user interface. They are declared in XML files and referenced in Java code. Layouts determine how widgets (e.g., buttons, text views) are arranged on the screen.

Common Layout Types:

LinearLayout: Aligns children in a single row or column.

RelativeLayout: Positions elements relative to each other or the parent.

ConstraintLayout: Provides more flexibility with constraints between elements.

FrameLayout: Used for a single child view, often for overlays.

TableLayout: Arranges children into rows and columns.

Activities and fragments allow modular and dynamic UI creation. Fragments can be embedded within activities for reusability and better device compatibility (e.g., tablets and phones).

Example Code: Setting up an Activity with a LinearLayout in Java

XML Layout (res/layout/activity_main.xml):

```
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical">
   <TextView
      android:id="@+id/textView"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello, World!" />

   <Button
      android:id="@+id/button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Click Me" />
</LinearLayout>
```

Java Code (MainActivity.java):

```
package com.example.myapp;
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

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

      TextView textView = findViewById(R.id.textView);
```

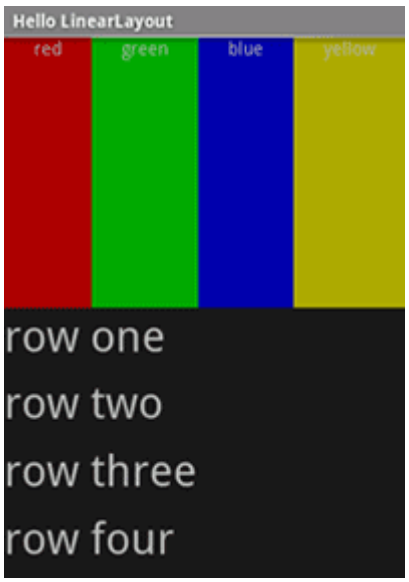
```
        Button button = findViewById(R.id.button);

        button.setOnClickListener(view -> textView.setText("Button Clicked!"));
    }
}
```

Lab exercises

1. Create an app that illustrates that the activity lifecycle method being triggered by various action. Understand when onCreate(),onStart(),onResume event occur.

2. Create a Calculator app that does the function of multiplication, addition, division, subtraction but displays the result in the format of:-Num1 operator num2 = result. Back button on the next activity should get back to the calculator activity again.

3. Create the following given scenario using linear and relative layout concept.



4. Create an app such that when the user click on the given URL typed by the user, it visits the corresponding page.