UNIT III

CHAPTER 6

Data, Security, App Publishing

Syllabus

Data - saving, retrieving, and loading: Overview to storing data, Shared preferences, SQLite primer, store data using SQLite database, ContentProviders, loaders to load and display data, Permissions, performance and security, Firebase and AdMob, Publish your app.

Syllabus Topic : Data : Saving, Retrieving, and Loading: Overview to Storing Data

6.1 Data : Saving, Retrieving and Loading: Overview to Storing Data

- Most non-trivial applications will have to store data in one way or another. This data can be of different forms, such as user settings, application settings, user data, images, or a cache of data fetched from the internet.
- Some apps might generate data that ultimately belongs to the user, and so, would prefer to store the data (perhaps documents or media) in a public place that the user can access at anytime, using other apps.
- Other apps might want to store data, but do not want this data to be read by other apps (or even the user). The Android platform provides developers with multiple ways to store data, with each method having its advantages and disadvantages.
- There are basically four different ways to store data in an Android app.

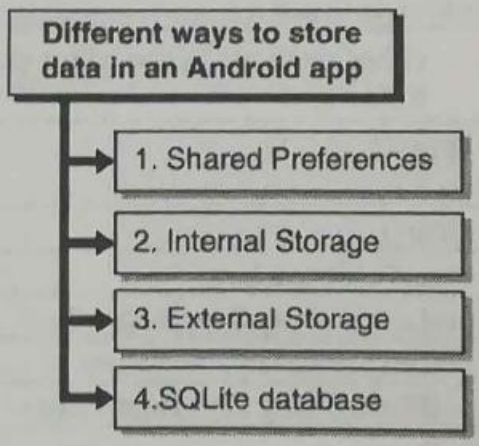


Fig. C6.1 : Ways to store data

- Let's look at each one of them in detail.

Syllabus Topic : Shared Preferences

➜ 6.1.1 Shared Preferences

- Android provides many ways of storing data of an application. One of this way is called Shared Preferences.
- Shared Preferences allow you to save and retrieve data in the form of key,value pair.
- In order to use shared preferences, you have to call a method getSharedPreferences() that returns a SharedPreference instance pointing to the file that contains the values of preferences.

```
SharedPreferencessharedpreferences = getSharedPreferences(MyPREFERENCES,
Context.MODE_PRIVATE);
```

- The first parameter is the key and the second parameter is the MODE. Other modes available are:

MODE_APPEND	Appends new preferences with already existing preferences
MODE_ENABLE_WRITE_AHEAD_LOGGING	This is a Database open flag. When it is set , it enables write ahead logging by default
MODE_WORLD_READABLE	When set, this mode allows other applications to read the preferences
MODE_WORLD_WRITEABLE	When set, this mode allows other applications to write the preferences
MODE_MULTI_PROCESS	This method checks for modification of preferences even if the SharedPreference instance is already loaded
MODE_PRIVATE	When this mode is set, the file can only be accessed using the calling application

- Saving can be done in the sharedpreferences by using SharedPreferences.Editor class. This can be done by invoking theedit() method of SharedPreference instance and receiving it in an editor object. The syntax is

```
Editor editor = sharedpreferences.edit();
editor.putString("key", "value");
editor.commit();
```

- Other methods in the editor class, that allows manipulation of data inside shared preferences are as follows.

apply()	(abstract method)Commitsthe changes from editor to the sharedPreference object that we are calling.
clear()	Removes all values from the editor
remove(String key)	Removes the value whose key has been passed as a parameter
putLong(String key, long value)	Saves a long value in a preference editor
putInt(String key, int value)	Saves an integer value in a preference editor
putFloat(String key, float value)	Saves a float value in a preference editor

Example

- This example demonstrates the use of the Shared Preferences. It display a screen with some text fields, whose value are saved when the application is closed and brought back when it is opened again.

To experiment with this example, you need to run this on an actual device on after developing the application according to the steps below –

Following is the content of the modified **MainActivity.java.**

```
package com.example.myapplication;
import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity{
EditText ed1,ed2,ed3;
Button b1;
public static final String MyPREFERENCES="MyPrefs";
public static final String Name= "nameKey";
public static final String Phone="phoneKey";
public static final String Email="emailKey";
SharedPreferences sharedpreferences;
@Override
protected void onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.Iayout.activity_main);
ed=(EditText)findViewByld(R.id.editText);
ed2=(EditText)findViewByld(R.id.editText2);
ed3=(EditText)findViewByld(R.id.editText3);
b1=(Button)findViewByld(R.id.button);
sharedpreferences = getSharedPreferences(MyPREFERENCES.Context.MODE_PRIVATE);
bl.setOnClicklistener(new View.OnClicklistener() {
@Override
public void onClick(View v){
String n  = edl.getText().toString();
String ph =ed2.getText().toString();
String e =  ed3.getText().toString();
SharedPreferences.Editor editor=sharedpreferences.edit();
editor.putString(Name,n);
editor.putString(phone,ph);
editor.putString(Email,e);
editor.commit();
Toast.makeText(MainActitivity.this,"Thanks",Toast.LENGTH_LONG).show());
```

```
}
});
} }
```

- Following is the content of the modified main activity file **res/layout/activiy_main.xml.**

```
<?xmlversion="1.0"encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"xmlns:tools="http://schemas.android.co
m/tools"android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="SharedPreference"
android:id="@+id/textView"
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true"
android:textSize="35dp"/>
<EditText
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/editText"
android:layout_below="@+id/textView2"
android:layout_marginTop="67dp"
android:hint="Name"
android:layout_alignParentRight="true"
android:layout_alignParentEnd="true" android:layout_alignParentleft="true"
android:layout_alignParentStart="true"/>
<EditText
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/editText2"
android:layout_below="@+id/editText"
android:layout_alignParentleft="true"
android:layout_alignParentStart="true"
android:layout_alignParentRight="true"
android:layout_alignParentEnd="true"
android:hint="Pass"/>
```

```
<EditText
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/editText3"
android:layout_below="@+id/editText2"
android:layout_alignParentleft="true"
android:layout_alignParentStart="true"
android:layout_alignParentRight="true"
android:layout_alignParentEnd="true"
android:hint="Email"/>
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Save"
android:id="@+id/button"
android:layout_below="@+id/editText3"
android:layout_centerHorizontal="true"
android:layout_marginTop="50dp"/>
</RelativeLayout>
```

- Do not make any changes to the strings.xml and AndroidManifest.xml file.

- To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar. If your phone is attached to your computer, Android studio installs the app on your phone and starts it and if everything is fine with your setup and application, it will display following on your phone.

- Now just put in some text in the field.

- Now when you press save button, the text will be saved in the shared preferences. Now press back button and exit the application. Now open it again and you will see all the text you have written back in your application.

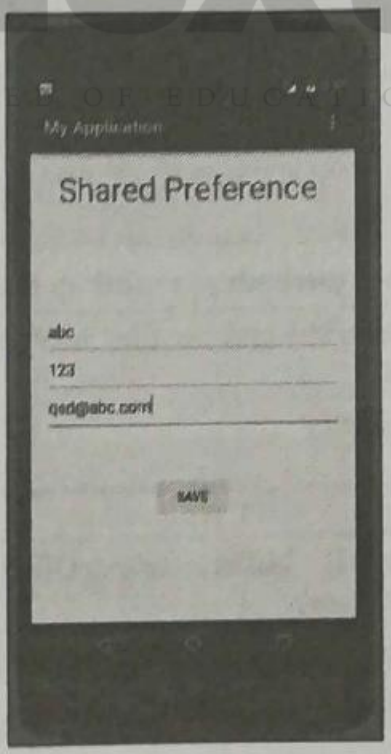


Fig. 6.1.1

6.1.2 Internal Storage

- Internal storage is the storage of the private data of an application on the device memory.
- By default these files are private and are accessed onlybythe application and get deleted, when user deletesthe application.

```
<EditText
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/editText3"
android:layout_below="@+id/editText2"
android:layout_alignParentleft="true"
android:layout_alignParentStart="true"
android:layout_alignParentRight="true"
android:layout_alignParentEnd="true"
android:hint="Email"/>
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Save"
android:id="@+id/button"
android:layout_below="@+id/editText3"
android:layout_centerHorizontal="true"
android:layout_marginTop="50dp"/>
</RelativeLayout>
```

- Do not make any changes to the strings.xml and AndroidManifest.xml file.
- To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar. If your phone is attached to your computer, Android studio installs the app on your phone and starts it and if everything is fine with your setup and application, it will display following on your phone.
- Now just put in some text in the field.
- Now when you press save button, the text will be saved in the shared preferences. Now press back button and exit the application. Now open it again and you will see all the text you have written back in your application.

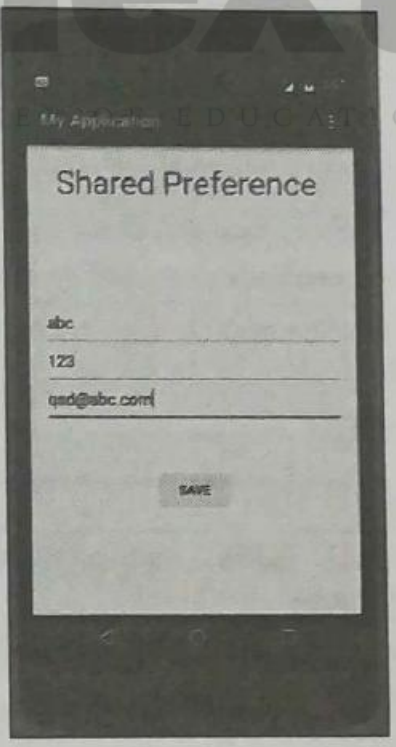


Fig. 6.1.1

6.1.2 Internal Storage

- Internal storage is the storage of the private data of an application on the device memory.
- By default these files are private and are accessed onlybythe application and get deleted, when user deletesthe application.

Writing file

- In order to use internal storage we need to write data in the file.
- For this purpose we call the openFileOutput() method with the name of the file and the mode.
- The mode could be private , public etc. Its syntax is given below –

```
FileOutputStreamfOut = openFileOutput("file name here",MODE_WORLD_READABLE);
```

- The method openFileOutput() returns an instance of FileOutputStream. It has to be received it in the object of FileInputStream. After that you can call write method to write data on the file. Its syntax is given below –

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

Reading file

- In order to read from the file you just created , call the openFileInput() method with the name of the file. It returns an instance of FileInputStream. Its syntax is given below –

```
FileInputStream fin = openFileInput(file);
```

- After that, you can call read method to read one character at a time from the file and then you can print it. Its syntax is given below –

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


Other methods provided by the FileOutputStream class

FileOutputStream(File file, boolean append)	This method constructs a new FileOutputStream that writes to file.
getChannel()	This method returns a write-only FileChannel that shares its position with this stream
getFD()	This method returns the underlying file descriptor
write(byte[] buffer, intbyteOffset, intbyteCount)	This method Writes count bytes from the byte array buffer starting at position offset to this stream

- Other methods provided by the **FileInputStream** class:

available()	This method returns an estimated number of bytes that can be read or skipped without blocking for more input
getChannel()	This method returns a read-only FileChannel that shares its position with this stream
getFD()	This method returns the underlying file descriptor
read(byte[] buffer, intbyteOffset, intbyteCount)	This method reads at most length bytes from this stream and stores them in the byte array b starting at offset

6.1.2(A) Example of Internal Storage

- Following is the content of the modified main activity file **src/MainActivity.java**.

```
package com.example.myapplication;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import java.io.FileInputStream;
import java.io.FileOutputStream;
public class MainActivity extends Activity{
Button b1,b2;
TextView tv;
EditText ed;
String data;
private String file="mydata";
@Override
protected void onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
b=(Button)findViewByld(R.id.button);
b2=(Button)findViewByld(R.id.button2);
ed =(EditText)findViewByld(R.id.editText);
tv=(TextView)findViewByld(R.id.textView2);
b  setOnClickListener(new View.OnClickListener(){
@Override
public void onClick(View v){
data=ed.getText().toString();
try{
FileOutputStream fOut=openFileOutput(file,MODE_WORLD_READABLE);
fOut.write(data .getBytes());
fOut.close();
Toast.makeText(getBaseContext(),"fiIesaved",Toast.LENGTH_SHORT).show();
}
catch(Exception e){
//TODO Auto-generated catch block
printStackTrace();
}
};
```

```
b2.setOnClickListener(new View.OnClickListener(){
@Override
public void onClick(View v){
try{
FileInputStream fin=openFileInput(file);
int c;
Stringt emp="";
while((c=fin.read())!=-1){
temp=temp+Character.toString((char)c);
}
tv.setText(temp);
Toast.makeText(getBaseContext(),"fileread",Toast.LENGTH_SHORT).show();
} } });
} }
catch(Exception e){ }
```

- Following is the modified content of the xml **res/layout/activity_main.xml**.
- In the below example abc indicates the image of Android logo which should be pasted in the **drawable** folder of the **res** folder.

```
<?xmlversion="1.O"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:paddingeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".MainActivity">
<TextView android:text="Internal Storage"
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"/>
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Save"
android:id="@+id/button"
android:layout_alignParentBottom="true"
android:layout_aligneft="@+id/textView"
```

```
android:layout_alignStart="@+id/textView"/>
<EditText
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android :id="@+id/editText"
android:hint="EnterText"
android:focusable="true"
android:textColorHighlight="#ff7eff15"
android:textColorHint="#ffff25e6"
android:layout_below="@+id/imageView"
android:layout_alignRight="@+id/textView"
android:layout_alignEnd="@+id/textView"
android:layout_marginTop="42dp"
android:layout_aligneft="@+id/imageView"
android:layout_alignStart="@+id/imageView"/>
<ImageView
android:layout_width="wrap_content"
android:layout_height="wrap_content"android :id="@+id/imageView"
android:src="@drawable/abe"
android:layout_below="@+id/textView"
android:layout_centerHorizontal="true"/>
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Load"
android:id="@+id/button2"
android:layout_alignTop="@+id/button"
android:layout_alignRight="@+id/editText"
android:layout_alignEnd="@+id/editText"/>
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Read"
android:id="@+id/textView2"
android:layout_below="@+id/editText"
android :layout_toLeft0f="@+id/button2"
android:layout_toStartOf="@+id/button2"
android:textColor="#ffSbf1f"
android:textSize="25dp"/>
</RelativeLayout>
```

- Do not make any changes to the strings.xml and AndroidManifest.xml file.

Following is the content of the modified main activity file **src/MainActivity.java**.

```
package com.example.externalstorage;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileinputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import android.os.Bundle;
import android.app.Activity;
import android.content.Context;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickistener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class MainActivity extends Activity{
EditText editTextFileName,editTextData;
Button saveButton,readButton;
@Override
protected void onCreate(Bundle savedInstanceState){
super.onCreate( savedInstanceState);
setContentView(R.layout.activity_main);
editTextFileName= (EditText)findViewByid(R.id.editTextl);
editTextData=(EditText)findViewBy d(R.id.editText2);
saveButton=(Button)findViewByid(R.id.button1);
readButton=(Button)findViewBy d(R.id.button2);
//Performing action on savebutton
saveButton.setOnClickListener(new  OnClickistener(){
@Override
public void onClick(View arg0){
String filename=editTextFileName.getText().toString();
String data=editTextData.getText().toString();
FileOutputStream  fos;
try{
File myFile=new File("/sdcard/"+filename);
myFile.createNewFile();
FileOutputStream fOut=new FileOutputStream(myFile);
```

```
OutputStreamWriter myOutWriter=new OutputStreamWriter(fOut);myOutWriter.append(data);
myOutWriter.close();
fOut.close();
Toast.makeText(getApplicationContext(),filename+"saved",Toast.LENGTH_LONG).show();
}
catch(FileNotFoundException e){e.printStackTrace();}
catch(IOExceptione){e.printStackTrace();}
 }
});
//Performing action on ReadButton
readButton.setOnClickistener(new OnClickistener(){
@Override
public void onClick(View arg0){
String filename=editTextFileName.getText().toString();
StringBuffer  stringBuffer=new StringBuffer();
String aDataRow="";
String aBuffer="";
try{
File myFile=new File("/sdcard/"+filename);
FileInputStream fln=new FileinputStream(myFile);
BufferedReader myReader=new BufferedReader(newInputStreamReader(fin));
while((aDataRow=myReader.readline())!=null) {
aBuffer+=aDataRow+"\n";
}
myReader.close();
}catch(IOException e){
e.printStackTrace();
 }
});
} }
Toast.makeText(getApplicationContext(),aBuffer,Toast.LENGTH_LONG).show();
@Override
public boolean onCreateOptionsMenu(Menu     menu){
//Inflate the menu; this adds items to the actionbar  if it is present.
getMenuInflater().inflate(R.menu.activity_main,menu);
return true;
} }
```

– Following is the modified content of the xml **res/layout/activity_main.xml**.

```
<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"
```

```
xt.MainActivity" >
<EditText
android:id="@+id/editText1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android :layout_alignParentRight="true"
android:layout_alignParentTop="true"
android:layout_marginRight="20dp"
android:layout_marginTop="24dp"
android:ems="10">
<requestFocus/>
</EditText>
<EditText
android:id="@+id/editText2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignRight="@+id/editText1"
android:layout_below="@+id/editText1"
android:layout_marginTop="24dp"
android:ems="10"/>
<TextView
android:id="@+id/textView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignBaseline="@+id/editText1"
android:layout_alignBottom="@+id/editText
android:layout_alignParentLeft="true"
android:text="File    Name:"/>
<TextView
android:id="@+id/textView2"
android:layout_width="wrap_content"android:layout_height="wrap_content"
android :layout_alignBaseline="@+id/editText2"
android:layout_alignBottom="@+id/editText2"
android:layout_alignParentLeft="true"
android:text="Data:"/>
<Button
android:id="@+id/button1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignLeft="@+id/editText2"android:layout_below="@+id/editText2"
android:layout_marginLeft="70dp"android:layout_marginTop="16dp"android:text="save"/>
```

```
<Button
android:id="@+id/button2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignBaseline="@+id/button1"
android:layout_alignBottom="@+id/button1"
android:layout_toRightOf="@+id/button1"
android:text="read"/>
</RelativeLayout>
```

- Following is the modified content of the xml ***AndroidManifest.xml***

```
<?xmlversion="1.0"encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.externalstorage"
android:versionCode="1"
android:versionName="1.0">
<uses-sdk
android:minSdkVersion="8"
android:targetSdkVersion="16" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<application
android:allowBackup="true"
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme">
<activity
android:name="com.example.externalstorage.MainActivity"
android:label="@string/app_name" >
<intent-filter>
<action android:name="android.intent.action.MAIN"/>
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
</application>
</manifest>
```

To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar. If your phone is attached to your computer, Android studio installs the app on your phone and starts it.

Syllabus Topic : SQLite Primer

6.1.4 SQL Databases

- A database is an organized collection of data.
- A database-management system (DBMS) is a computer-software application that interacts with end-users, other applications, and the database itself to capture and analyze data. Well-known DBMSs include MySQL, PostgreSQL, EnterpriseDB, MongoDB, Microsoft SQL Server, Oracle, Sybase, SQLite and IBM DB2.
  - Database designers typically organize the data, to model aspects of reality,in a way that supports processesrequiring information, for example modelling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.
  - In a database data is stored in tables of rows and columns.
  - The intersection of a row and column is called a field.
  - Fields contain data, references to other fields, or references to other tables.
  - Rows are identified by unique IDs.
  - Columns are identified by names that are unique per table.
  - It is like a spreadsheet with rows, columns, and cells, where cells can contain data, references to other cells, and links to other sheets.

SQLite

- SQLite is a software library that implements SQL database engine that is:
  - self-contained (requires no other components)
  - server less (requires no server backend)
  - zero-configuration (does not need to be configured for your application)
  - transactional (changes within a single transaction in SQLite either occur completely or not at all)
- SQLite is the most widely deployed database engine in the world. The source code for SQLite is in the public domain.

Example table

- SQLite stores data in tables.
- Assume the following:
  - A database DATABASE_NAME
  - A table WORD_LIST_TABLE
  - Columns for _id, word, and description
- After inserting the words "alpha" and "beta", where alpha has two definitions, the table might look like this:

☞ **DATABASE_NAME**

id	word	Definition
1	"alpha"	"first letter"
2	"beta"	"second letter"
3	"alpha"	"particle"

- You can find what's in a specific row using the _id, or you can retrieve rows by formulating qu that select rows from thetable be specifying constraints.
- You use the SQL query language discussed below to create queries.

6.2 Transactions

- A sequence of operations performed as a single logical unit of work is a transaction.
- In order to qualify as a transaction, such a logical unit of work exhibits fourproperties name atomicity, consistency, isolation, and durability (ACID) properties.
- All changes within a single transaction must either occur completely or not at all, even if writing to the disk is interrupted by:
  - a program crash,
  - an operating system crash, or
  - a power failure.

☞ **Examples of transactions**

- Transferring money from a savings account to a checking account.
- Entering a term and definition into dictionary.
- Committing a changelist to the master branch.

6.2.1 ACID

☞ **Atomicity**

- Atomicity is based on the concept that each transaction be "all or nothing": if any one part of the transaction in a sequence fails, then the entire transaction fails, and there will be no change in database state.
- An atomic system must guarantee atomicity in every situation, including power failures, errors and crashes. For the outside world, a committed transaction appears completely by its effects on the database. That means it should be indivisible ("atomic").
- This property states that, either all operations contained by a transaction are done successfully, or none of them complete at all. This property is very useful to maintain the consistency of data.

☞ **Consistency**

- The consistencyproperty ensures that the transaction executed on the database system will bring the database from one valid state to another.
- The data which is written by the transaction must be valid according to the standard rules and regulations regarding constraints, cascades, triggers etc. Consistency does not guarantee accuracy of the transaction as per the expectations of programmer.

Isolation

- When transactions are performed in a sequence, the state of a system is always valid without any problem. But sometimes we may have to perform multiple transactions concurrently.
- In case of concurrent transactions, the isolation property ensures that the system state should be same that would be obtained if transactions were executed sequentially, i.e. one after the other. The effect of any incomplete transaction should be invisible to other transaction by means of isolation.

Durability

- The durability property assures that after a transaction has committed successfully, the updates made should remain permanent in the database, even in the event of power loss, crashes or errors.
- For example in a database management system, when a series of transactions is executed, the modification done should be stored permanently, even if the database crashes just after the transaction. To avoid the problem because of power loss, the states of a database after every transaction, must be recorded in a non-volatile memory.

6.2.2 Query Language

- Query language is primarily created for creating, accessing and modifying data in and out from a database management system (DBMS).
- Typically, a query language requires users to input a structured command that is similar and close to the English language querying construct.
- For example, the SQL query: SELECT * FROM
- The customer will retrieve all data from the customer records/table.
- The simple programming context makes it one of the easiest programming languages to learn. There are several different variants of query languages and it has wide implementation in various database-centered services(such as extracting data from deductive and OLAP databases, providing API based access to remote applications and services and more.)
- Queries can be very complex, but the basic operations are:
  - inserting rows
  - deleting rows
  - updating values in rows
  - retrieving rows that meet given criteria
- On Android, the database object provides convenient methods for inserting, deleting, and updating the database. You only need to understand SQL for retrieving data.

6.2.3 Query Structure

- An SQL query is highly structured and contains the following basic parts:

```
SELECT word, description FROM WORD_LIST_TABLE WHERE word="alpha"
```

- Generic version of sample query:

```
SELECT columns FROM table WHERE column="value"
```

- Parts
  - SELECT columns : Select the columns to return. Use * to return all columns.
  - FROM table : Specify the table from which to get results.
  - WHERE : Keyword for conditions that have to be met.

- column="value" : The condition that has to be met.
- common operators : =, LIKE, <, >
- AND, OR : connect multiple conditions with logic operators.
- ORDER BY : omit for default order, or specify ASC for ascending, DESC for descending.
- LIMIT is a very useful keyword if you want to only get a limited number of results.

☞ **Sample queries**

SELECT * FROM WORD_LIST_TABLE	Get the whole table.
SELECT word, definition FROM WORD_LIST_TABLE WHERE _id > 2	Returns **[["alpha", "particle"]]**
SELECT _id FROM WORD_LIST_TABLE WHERE word="alpha" AND definition LIKE"%art%"	Return the id of the word alpha with the substring "art" in thedefinition. **[["3"]]**
SELECT * FROM WORD_LIST_TABLE ORDER BY word DESC LIMIT 1	Sort in reverse and get the first item. This gives you the lastitem per sort order. Sorting is by the first column, in this case, the _id. **[["3","alpha","particle"]]**
SELECT * FROM WORD_LIST_TABLE LIMIT 2,1	Returns 1 item starting at position 2. Position counting startsat 1 (not zero!). Returns **[["2", "beta", "second letter"]]**

6.2.4 Queries for Android SQLite

- You can send queries to the SQLite database of the Android system as raw queries or as parameters.
  - **rawQuery(String sql, String[] selectionArgs)** runs the provided SQL and returns a Cursor of the result set.
- The following table shows how the first two queries from above would look as raw queries.

1.	String query = "SELECT * FROM WORD_LIST_TABLE"; rawQuery(query, null);
2.	query = "SELECT word, definition FROM WORD_LIST_TABLE WHERE _id> ? "; String[] selectionArgs = new String[]{"2"} rawQuery(query, selectionArgs) ;

- query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, StringorderBy, String limit) queries the given table, returning a Cursor over the result set.
- Here's a query showing how to fill in the arguments:

```
SELECT * FROM WORD_LIST_TABLE
WHERE word="alpha"
ORDER BY word ASC
LIMIT 2,1;
```

Returns

```
[["alpha", "particle"]]
String table = "WORD_LIST_TABLE"
```

```
String[] columns = new String[]{"*"};
String selection = "word = ?"
String[] selectionArgs = new String[]{"alpha"};
String groupBy = null;
String having = null;
String orderBy = "word ASC"
String limit = "2,1"
query(table, columns, selection, selectionArgs, groupBy, having, orderBy, limit);
```

- Note that in real code, you wouldn't create variables for null values.

6.2.5 Cursors

- Queries always return a Cursor object.
- A Cursor is an object interface that provides random read-write access to the resultset returned by a database query. It points to the first element in the result of the query.
- A cursor is a pointer into a row of structured data. You can think of it as a pointer to table rows.
- The Cursor class provides methods for moving the cursor through that structure, and methods to get the data from the columns of each row.
- When a method returns a Cursor object, you iterate over the result, extract the data, do something with the data, and finally close the cursor to release the memory.

Syllabus Topic : Store Data using SQLite database

6.3 Store Data using SQLite Database

- SQLite is an open-source relational database i.e. used to perform database operations on Android devices such as storing, manipulating or retrieving persistent data from the database.
- It is embedded in Android by default. So, there is no need to perform any database setup or administration task like JDBC, ODBC etc.
- SQLiteOpenHelper class provides functionality to use SQLite database.
- SQLite stores data to a text file on the device.
- SQLite supports all the relational database features.

6.3.1 Database Package

- The main package is android.database.sqlite that contains the classes to manage databases

Database - Creation

- In order to create a database invoke the method openOrCreateDatabase() with the database name and mode as a parameter.
- It returns an instance of SQLite database which we have to receive in our own object. Its syntax is given below

```
SQLiteDatabase mydatabase = openOrCreateDatabase("your database
name",MODE_PRIVATE,null);
```

- Other functions available in the database package, which does this job are listed below.

openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags, DatabaseErrorHandlererrorHandler)	Opens existing database with the appropriate flag mode. Modes specify OPEN_READWRITE OPEN_READONLY
openDatabas(String path, SQLiteDatabase.CursorFactory factory, int flags)	Also opens the existing database but does not define any handler to handle the errors of databases
openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory)	This method not only opens the database, but also creates the database if it not exist. This method is equivalent to openDatabase method.
openOrCreateDatabase(File file, SQLiteDatabase.CursorFactory factory)	This method is similar to above method but it takes the File object as a path not a string. It is equivalent to file.getPath()

☞ **Database – Insertion**

- To create table or insert data into a table we use,execSQL() method. It is defined in SQLiteDatabase class. The syntax is given below:

```
mydatabase.execSQL("CREATE TABLE IF NOT EXISTS TutorialsPoint(Username
VARCHAR,Password VARCHAR);");
mydatabase.execSQL("INSERT INTO TutorialsPoint VALUES('admin','admin');");
```

- This method will insert values into the table in the database. Another method that does the same job but takes some additional parametersis :

execSQL(String sql, Object[] bindArgs)	This method not only inserts data , but is also used to update or modify the already existing data in database (using bind arguments)

☞ **Database – Fetching**

- Data can be retrieved from the database using an object of the Cursor class.
- For thisinvoke a method of this class called rawQuery(). It returns a resultset with the cursor pointing to the table.
- Data can be retrieved by moving the cursor forward.

```
Cursor resultSet = mydatbase.rawQuery("Select * from TutorialsPoint",null);
resultSet.moveToFirst();
String username = resultSet.getString(0);
String password = resultSet.getString(1);
```

- There are other functions available in the Cursor class that allows us to effectively retrieve the data. That includes
- Functions available in the Cursor class that allow effective retrieval of data are:

getColumnCount()	Returns the total number of columns in the table.
getColumnIndex(String columnName)	Returns the index number of a column by specifying the name of the column.
getColumnName(intcolumnIndex)	Returns the name of the column by specifying the index of the column
getColumnNames()	Returns the array of all the column names of the table.
getCount()	Returns the total number of rows in the cursor.
getPosition()	Returns the current position of the cursor in the table.
isClosed()	Returns true if the cursor is closed else false.

Database - Helper class

In order to manage all the operations related to the database,a helper class called **SQLiteOpenHelper**has been given. It manages the creation and updation of the database. Its syntax is:

```
public class DBHelper extends SQLiteOpenHelper {
public DBHelper(){
super(context,DATABASE_NAME,null,1);
}
public void onCreate(SQLiteDatabase db) {}
public void onUpgrade(SQLiteDatabase database, int oldVersion, int newVersion) {}
}
```

6.3.1(A) Example of Database Package

Here is an example demonstrating the use of SQLite Database. It creates a basic contacts applications that allows insertion, deletion and modification of contacts.

Following is the content of the modified **MainActivity.java**.

```
package com.example.myapplication;
import android.content.Context;
import android.content.Intent;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
import java.util.Arraylist;
import java.util.List;
public class MainActivity extends ActionBarActivity{
public final static String EXTRA_MESSAGE="MESSAGE"; private ListView obj;
DBHelper mydb;
@Override
protected void onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R .layout.activity_ main);
mydb=new DBHelper(this);
ArrayList array_list=mydb.getAllCotacts();
ArrayAdapter arrayAdapter=new ArrayAdapter(this,android.R.layout.simple_list_item_ ,array_list);
```

```
obj=(ListView)findViewByld(R.id.listViewl);
obj.setAdapter(arrayAdapter);
obj.setOnItemClicklistener(new OnltemClicklistener(){
@Override
public void onltemClick(AdapterView<?>argO,View arg1,int arg2,long arg3){
//TODO Auto-generated method stub
int id_To_Search=arg2+1;
Bundle dataBundle=new Bundle();
dataBundle.putInt("id",id_To_Search);
Intent intent=new Intent(getApplicationContext(),DisplayContact.class);
}
});
}
intent.putExtras(dataBundle);
startActivity(intent);
@Override
public boolean onCreateOptionsMenu(Menu menu) {
//Inflate the menu; this adds items to the actionbar if it is present.
getMenuInflater().inflate(R.menu.menu_ main,menu);
return true;
}
@Override
public boolean onOptionsItemSelected{MenuItem item){
super.onOptionsItemSelected{item);
switch{item.getItemld()){
case R.id.iteml:Bundle dataBundle=new Bundle();
dataBundle.putInt("id",O);
Intent intent=new Intent(getApplicationContext(),DisplayContact.class);
intent.putExtras(dataBundle);
startActivity(intent);
return true;
default:
return super.onOptionsItemSelected(item);
} }
Public Boolean
onKeyDown(int keycode,KeyEvent event){if(keycode==KeyEvent.KEYCODE_BACK){
moveTaskToBack(true);
}
return super.onKeyDown(keycode,event);
} }
```

Create new src/DBHelper.java that will manage the database work. Following is the content of Database class **DBHelper.java**

```
package com.example.myapplication;
import java.util.Arraylist;
import java.util.HashMap;
import java.util.Hashtable;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.DatabaseUtils;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteDatabase;

public class DBHelper extends SQLiteOpenHelper{
public static final String DATABASE_NAME="MyDBName.db";
public static final String CONTACTS_TABLE_NAME="contacts";
public static final String CONTACTS_COLUMN_ID="id";
public static final String CONTACTS_COLUMN_NAME="name";
public static final String CONTACTS_COLUMN_EMAIL="email";
public static final String CONTACTS_COLUMN_STREET="street";
public static final String CONTACTS_COLUMN_CITY="place";
public static final String CONTACTS_COLUMN_PHONE="phone";
private HashMap hp;

public DBHelper(Context context){
super(context, DATABASE_NAME,nulll);
}

@Override
public void onCreate(SQLiteDatabase db){
//TODO Auto-generated method stub
db.execSQL("createtablecontacts"+"(id integer primary key, name text,phone text, email text, street
text, place text)"
);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion){
//TODO Auto-generated method stub
db.execSQL("DROP TABLE IF EXISTS contacts");
onCreate(db);
```

```
}

public boolean insertContact(String name,String phone,String email,String street,String place){
SQLiteDatabase db=this.getWritableDatabase();
ContentValues contentValues=new ContentValues();
contentValues.put("name",name);
contentValues.put("phone",phone);
contentValues.put("emaii",emaii);
contentValues.put("street",street);
contentValues.put("place",place);
db.insert("contacts",null,contentValues);
return true;
}
public Cursor getData(int id){
SQLiteDatabase db=this.getReadableDatabase();
Cursor res= db.rawQuery("select*from contacts where id="+id+"",null);
return res;
}

public int numberOfRows(){
SQLiteDatabase db=this.getReadableDatabase();
int numRows=(int)DatabaseUtils.queryNumEntries(db,CONTACTS_TABLE_NAME);
return numRows;
}
public boolean updateContact(Integer id,String name,String phone,String email,String street,String
place)
{
SQLiteDatabase db=this.getWritableDatabase();
ContentValues contentValues=new ContentValues();
contentValues.put("name",name);
contentValues.put("phone",phone);
contentValues.put("emaii",emaii);
contentValues.put("street",street);
contentValues.put("place",place);
db.update("contacts",contentValues,"id=?",new String[]{Integer.toString(id)});
return true;
}

public Integer deleteContact(Integer id){
SQLiteDatabase db=this.getWritableDatabase();
return db.delete("contacts", "id-- ?", new String[]{ Integer.toString(id)});
```

```
}
public Arraylist<String>getAllCotacts(){Arraylist<String> array_list=new Arraylist<String>();

//hp=new HashMap();
SQLiteDatabase db=this.getReadableDatabase();
Cursor res=db.rawQuery("select * from contacts ",null);
res.moveToFirst();
while(res.isAfterlast()==false){
array_list.add(res.getString(res.getColumnIndex(CONTACTS_COLUMN_NAME)));res.moveToNe
xt();
}
return array_list;
}}
```

- Create a new Activity as DisplayContact.java that will display the contact on the screen. Following is the modified content of display contact activity **DisplayContact.java**

```
package com.example.myapplication;
import android.os.Bundle;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class DisplayContact extends Activity{
int from_Where_I_Am_Coming=0;
private DBHelper mydb;

TextView name;
TextView phone;
TextView email;
TextView street;
TextView place;
int id_To_Update=0;
@Override
protected void onCreate(Bundle savedInstanceState){
```

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_display_contact);
name=(TextView)findViewById(R.id.editTextName);
phone=(TextView)findViewById(R.id.editTextPhone);
email=(TextView)findViewById(R.id.editTextStreet);
street=(TextView)findViewById(R.id.editTextEmail);
place=(TextView)findViewById(R.id.editTextCity);

mydb=new DBHelper(this);

Bundle extras=getIntent().getExtras();

if(extras !=null){
int Value=extras.getInt("id");
if(Value>0){
//means this is the view part not the add contact part.
Cursor rs=mydb.getData(Value);
id_To_Update=Value;
rs.moveToFirst();

String nam=rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_NAME));
String phon=rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_PHONE));
String emai=rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_EMAIL));
String stree=rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_STREET));
String plac=rs.getString(rs.getColumnIndex(DBHelper.CONTACTS_COLUMN_CITY));

if(!rs.isClosed()){rs.close();}
Button b=(Button)findViewById(R.id.button1);
b.setVisibility(View.INVISIBLE);
name.setText((CharSequence)nam);
name.setFocusable(false);
name.setClickable(false);

phone.setText((CharSequence)phon);
phone.setFocusable(false);
phone.setClickable(false);

email.setText((CharSequence)emai);
email.setFocusable(false);
email.setClickable(false);
```

```
street.setText((CharSequence)stree);
street.setFocusable(false);
street.setClickable(false);

place.setText((CharSequence)plac);
place.setFocusable(false);
place.setClickable(false);
}
}
}

@Override
public Boolean onCreateOptionsMenu(Menu menu){
//Inflate the menu; this adds items to the actionbar if it is present.
Bundle extras=getIntent().getExtras();

if(extras!=null){
int Value=extras.getInt("id");if(Value>0){
getMenuInflater().inflate(R.menu.display_contact,menu);
}else{
getMenuInflater().inflate(R.menu.menu_mainmenu);
}
}
return true;
}

public boolean onOptionsItemSelected(MenuItem item){
super.onOptionsItemSelected(item);
switch(item.getItemId()){
case R.id.Edit Contact:
Button b=(Button)findViewByld(R.id.button1);
b.setVisibility(View.VISIBLE);name.setEnabled(true);
name.setFocusableInTouchMode(true);
name.setClickable(true);
phone.setEnabled(true);
phone.setFocusableInTouchMode(true);
phone.setClickable(true);
email.setEnabled(true);
email.setFocusableInTouchMode(true);
email.setClickable(true);
street.setEnabled(true);
```

```
street.setFocusableInTouchMode(true);
street.setClickable(true);

place.setEnabled(true);
place.setFocusableInTouchMode(true);
place.setClickable(true);

return true;
case R.id.Delete_Contact:
AlertDialog.Builder builder=new AlertDialog.Builder(this);
builder.setMessage(R.string.deleteContact)
.setPositiveButton(R.string.yes,new DialogInterface.OnClicklistener(){
public void onClick(DialogInterface dialog,int id){
mydb.deleteContact(id_To_Update);
Toast.makeText(getApplicationContext(),"DeletedSuccessfully",Toast.LENGTH_SHORT).show();
Intent intent=new Intent(getApplicationContext(),MainActivity.class);startActivity(intent);
}
})
.setNegativeButton(R.string.no,new DialogInterface.OnClicklistener(){
public void onClick(DialogInterface dialog,int id){
//User cancelled the dialog
}
});

AlertDialog d=builder.create();
d.setTitle("Areyousure");
d.show();

return true;
default:
return super.onOptionsItemSelected(item);
} }
public void run(View view){
Bundle extras=getIntent().getExtras();
if(extras!=null){
int Value= extras.getInt("id");
if(Value>0){
if(mydb.updateContact(id_To_Update,name.getText().toString(),
phone.getText().toString(),email.getText().toString(),
street.getText().toString(),place.getText().toString()))
{
```

```
Toast.makeText(getApplicationContext(),"Updated",Toast.LENGTH_SHORT).show();
Intent intent=new Intent(getApplicationContext(),MainActivity.class);
startActivity(intent);

}
else{
Toast.makeText(getApplicationContext(),"notUpdated",Toast.LENGTH_SHORT).show();

}
}else{
if(mydb.insertContact(name .getText().toString(),phone.getText().toString(),
email.getText().toString(),street.getText().toString(),place.getText().toString())){
Toast .makeText(getApplicationContext(),"done",
}else{
Toast.LENGTH_SHORT).show();
Toast.makeText(getApplicationContext(),"not done",
Toast.LENGTH_SHORT).show();
}
Intent intent=new Intent(getApplicationContext(),MainActivity.class);
startActivity(intent);
}
) }
}
```

- Following is the content of the **res/layout/activity_main.xml**

```
<?xmlversion="1.0"encoding="utf-8"?>
<Relativelayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".MainActivity">

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/textView"
android:layout_alignParentTop="true"
android :layout_centerHorizontal="true"
android:textSize="30dp"
```

```
android:text="DataBase"/>

<ImageView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/imageView"
android:layout_below="@+id/textView2"
android :layout_centerHorizontal="true"
android:src="@drawable/abe"/>

<ScrollView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/scrollView"
android:layout_below="@+id/imageView"
android:layout_alignParentleft="true"
android:layout_alignParentStart="true"
android:layout_alignParentBottom="true"
android:layout_alignParentRight="true"
android:layout_alignParentEnd="true">

<ListView
android:id="@+id/listViewl"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_centerVertical="true" >
</ListView>
</ScrollView>
```

- Following is the content of the res/layout/activity_display_contact.xml

```
<?xmlversion="1.0"encoding="utf-8"?>
<ScrollViewxmlns:android="http:/schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/scrollView1"
android:layout_width="match_parent"
android:layout_height="wrap_content"
tools:context=".DisplayContact">

<RelativeLayout
android:layout_width="match_parent"
android:layout_height="370dp"
```

```
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin">

<EditText
android:id="@+id/editTextName"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentleft="true"
android:layout_marginTop="Sdp"
android:layout_margineft="82dp"
android:ems="10"android:inputType="text">
</EditText>

<EditText
android:id="@+id/editTextEmail"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignLeft="@+id/editTextStreet"
android:layout_below="@+id/editTextStreet"
android:layout_marginTop="22dp"
android:ems="10"
android:inputType="textEmailAddress"/>

<TextView
android:id="@+id/textView1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignBottom="@+id/editTextName"
android:layout_alignParentleft="true"
android:text="@string/name"
android:textAppearanee="?android:attr/textAppearaneeMedium"/>

<Button
android:id="@+id/button1"
android :layout_width="wrap_content"
android :layout_height="wrap_content"
android:layout_a lignLeft="@+id/editTextCity"
android:layout_alignParentBottom="true"
android:layout_marginBottom="28dp"
```

```
android:onClick="run"
android:text="@string/save"/>

<TextView
android:id="@+id/textView2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignBottom="@+id/editTextEmail"
android:layout_alignleft="@+id/textViewl"
android:text="@string/emaii"
android:textAppearanee="?android:attr/textAppearaneeMedium"/>


<TextView
android:id="@+id/textView5"
android:layout_width="wrap_eontent"
android:layout_height="wrap_eontent"
android:layout_alignBottom="@+id/editTextPhone"
android:layout_alignleft="@+id/textView1"
android:text="@string/phone"
android:textAppearanee="?android:attr/textAppearaneeMedium"/>

<TextView
android:id="@+id/textView4"
android:layout_width="wrap_eontent"
android:layout_height="wrap_eontent"
android:layout_above="@+id/editTextEmail"
android:layout_alignleft="@+id/textView4"
android:text="@string/street"
android:textAppearanee="?android:attr/textAppearaneeMedium"/>

<EditText
android:id="@+id/editTextCity"
android:layout_width="wrap_eontent"
android:layout_height="wrap_eontent"
android:layout_alignRight="@+id/editTextName"
android:layout_below="@+id/editTextEmaiI"
android:layout_marginTop="30dp"
android:ems="10"
android:inputType="text"/>
```

```
<TextView
android:id="@+id/textView3"
android:layout_width="wrap_eontent"
android:layout_height="wrap_eontent"
android:layout_ a lignBaseline="@+id/editTextCity"
android:layout_alignBottom="@+id/editTextCity"
android:layout_alignParentleft="true"
android:layout_toeftOf="@+id/editTextEmaiI"
android:text="@string/eountry"
android:textAppearanee="?android:attr/textAppearaneeMedium"/>

<EditText
android:id="@+id/editTextStreet"
android:layout_width="wrap_eontent"
android:layout_height="wrap_eontent"
android:layout_alignleft="@+id/editTextName"
android:layout_below="@+id/editTextPhone"
android:ems="10"
android:inputType="text">

<requestFoeus/>
</EditText>

<EditText
android :id="@+id/editTextPhone"
android:layout_width="wrap_eontent"
android:layout_height="wrap_eontent"
android:layout_alignLeft="@+id/editTextStreet"
android:layout_below="@+id/editTextName"
android:ems="10"
android:inputType="phoneItext"/>
</Relativelayout>
</SerollView>
```

- Following is the content of the **res/value/string.xml**

```
<?xml version="1.0"encoding="utf-8"?>
<resources>
<string name="app_name">Address Book</string>
<string name="action_settings">Settings</string>
<string name="hello_world">Hello world!</string>
```

```
<string name="Add_New">Add New</string>
<string name="edit">Edit Contact</string>
<string name="delete">Delete Contact</string>
<string name="title_activity_display_contact">Display Contact</string>
<string name="name">Name</string>
<string name="phone">Phone</string>
<string name="email">Email</string>
<string name="street">Street</string>
<string name="country">City/State/Zip</string>
<string name="save">Save Contact</string>
<string name="deleteContact">Are you sure ,you want to delete it.</string>
<string name="yes">Yes</string>
<string name="no">No</string>
</resources>
```

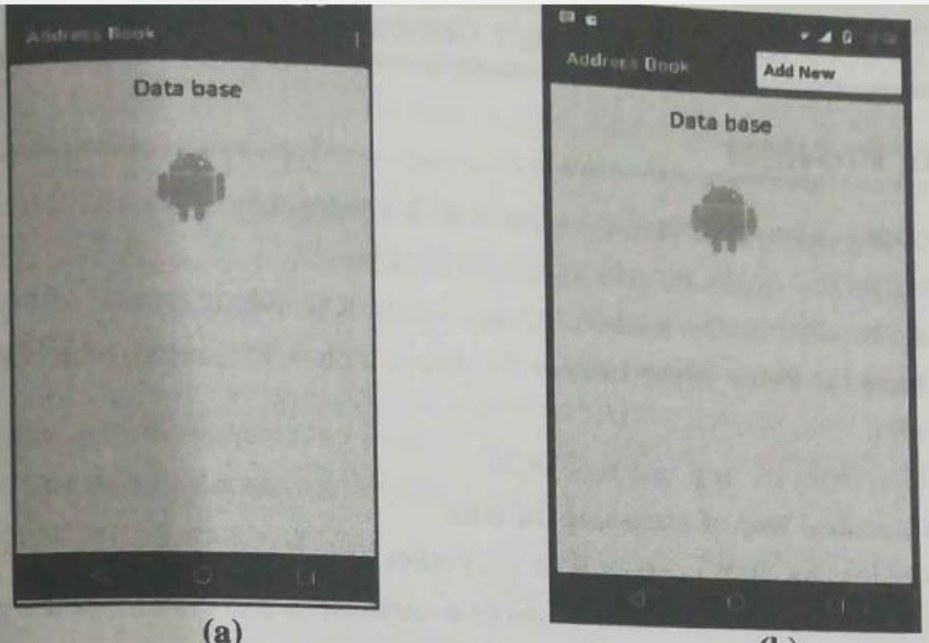
- Following is the content of the **res/menu/main_menu.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item android:id="@+id/item "
android:icon="@drawable/add"
android:title="@string/Add_New">
</item>
</menu>
```


- Following is the content of the **res/menu/display_contact.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item
android:id="@+id/Edit_Contact"
android:orderInCategory="100"
android:title="@string/edit"/>

<item android:id="@+id/Delete_Contact"
android:orderInCategory="100"
android:title="@string/delete"/>
</menu>
```

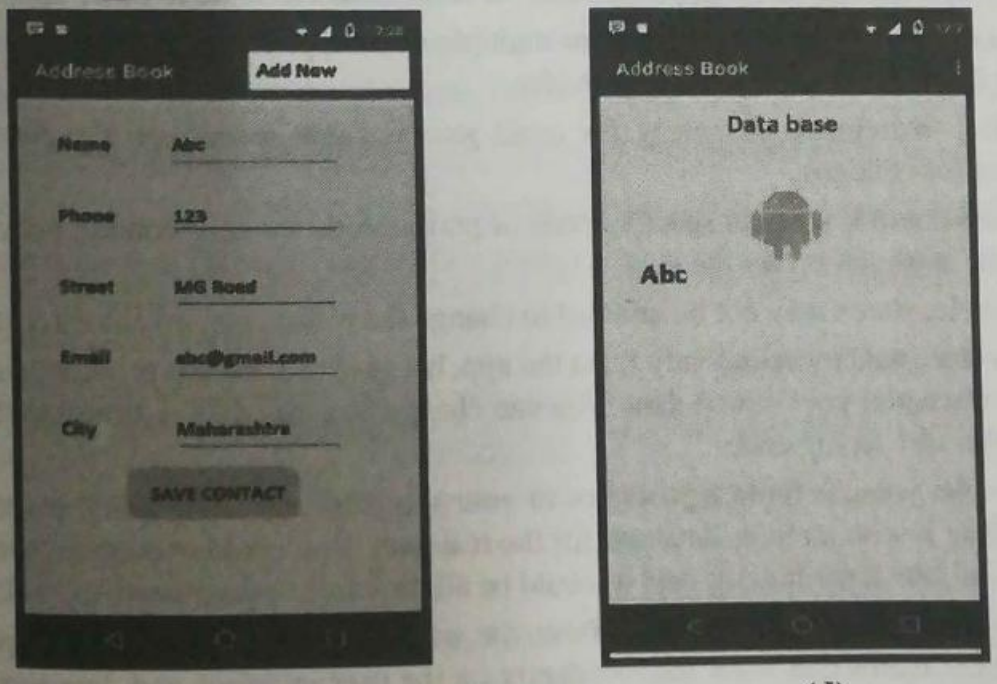
- Do not make any changes to AndroidManifest.xml file.
- To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar.
- If your phone is attached to your computer, Android studio installs the app on your phone and starts it and if everything is fine with your setup and application, it will display following on your phone (Fig. 6.3.1(a)).



(a) (b)

Fig. 6.3.1

- Open the Optional menu, it will show as Fig. 6.3.1(b) image.
- Click on the add button of the menu screen to add a new contact. It will display the following screen((Fig. 6.3.1(c))).
- Please enter the required information and click on save contact. It will bring you back to main screen (Fig. 6.3.1(d)).
- Now our contact **Abc** has been added.In order to see where the database is created, open Android Studio, connect your mobile. Go **tools/android/android device monitor**.
- Now browse the file explorer tab.
- Now browse this folder /data/data/<your.package.name>/databases<database-name>.



(c) (d)

Fig. 6.3.1

6.4 Content Provider

- A ContentProvider is a component that interacts with a repository.
- It supplies data from one application to others on request.
- Such requests are handled by the methods of the ContentResolver class.
- The app doesn't need to know where or how the data is stored, formatted, or accessed.
- A content provider:
  - Separates data from the app interface code
  - Provides a standard way of accessing the data
  - Makes it possible for apps to share data with other apps
  - Is agnostic to the repository, which could be a database, a file system, or the cloud.

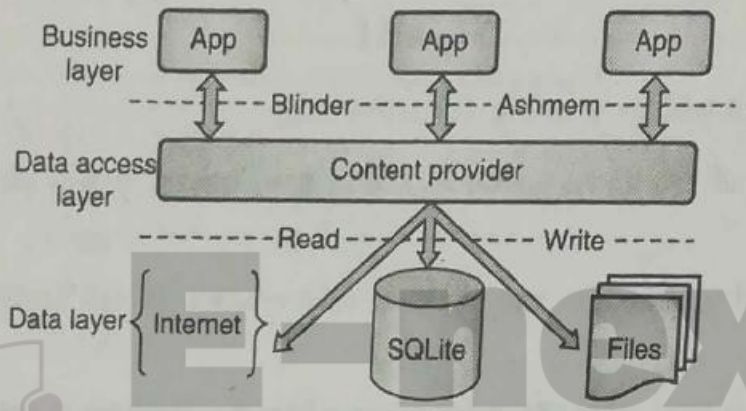


Fig. 6.4.1

ContentProviders are good.

- Content providers are useful for apps that want to make data available to other apps.
  - With a content provider, you can allow multiple other apps to securely access, use, and mo a single data source that your app provides.
  - Examples: Warehouse inventory for retail stores, game scores, or a collection of phy problems for colleges.
  - For access control, you can specify levels of permissions for your content provider, specify how other apps can access the data.
  - For example, stores may not be allowed to change the warehouse inventory data.
  - You can store data independently from the app, because the content provider sits between yo user interface and your stored data. You can change how the data is stored without needing change the user-facing code.
  - For example, you can build a prototype of your shopping app using mock inventory data, the later replace it with an SQL database for the real data. You could even store some of your dat in the cloud and some locally, and it would be all the same to your users.
  - Another benefit of separating data from the user interface with a content provider is that development teams can work independently on the user interface and data repository of your app. For larger, complex apps it is very common that the user interface and the data backend are developed by different teams, and they can even be separate apps; that is, it is not required

that the app with the content provider have a user interface. For example, your inventory app could consist only of the data and the content provider.

- There are other classes that expect to interact with a content provider. For example, you must have a content provider to use a loader, such as CursorLoader, to load data in the background.

> **Note :** If your app is the only one using the data, and you are developing all of it by yourself, you probably don't need acontent provider.

- Content providers let you centralize content in one place and have many different applications access it as needed.
- A content provider behaves very much like a database where you can query it, edit its content, as well as add or delete content using insert(), update(), delete(), and query() methods.
- In most cases this data is stored in an **SQlite** database.
- A content provider is implemented as a subclass of **ContentProvider** class and must implement a standard set of APIs that enable other applications to perform transactions.

```
public class My Application extends ContentProvider {
}
```

6.4.1 Content URIs

- To query a content provider, you specify the query string in the form of a URI which has following format :

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

- Here is the detail of various parts of the URI.

Prefix	This is always set to content://
authority	Specifies the name of the content provider, eg*contacts*, *browser* etc. For third-party content providers, specify the fully qualified name, such as *com.androidprograms.statusprovider*
data_type	Specifies the type of data that this particular provider provides. For example, for are getting all the contacts from the *Contacts*content provider, then the data path would be *members* and URI would look like this *content://contacts/members*
id	Specifies the specific record requested. For example, to search for contact number 3 in the Contacts content provider, then URI would be *content://contacts/people/3*.

6.4.2 Create Content Provider

Q. How to create content provider?

- This involves number of simple steps to create your own content provider.
  - First of all you need to create a Content Provider class that extends the ContentProviderbaseclass.
  - Second, you need to define your content provider URI address which will be used to access the content.

- Next you will need to create your own database to keep the content. Usually, Android uses SQLite database and framework needs to override onCreate() method which will use SQLite Open Helper method to create or open the provider's database. When your application is launched, the onCreate() handler of each of its Content Providers is called on the main application thread.
- Next you will have to implement Content Provider queries to perform different database specific operations.
- Finally register your Content Provider in your activity file using <provider> tag.

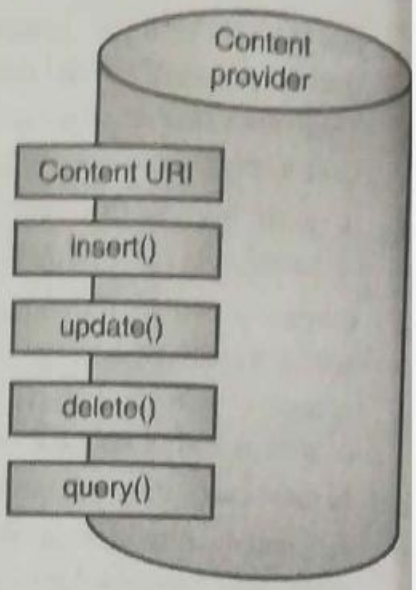


Fig. 6.4.2 : ContentProvider

Methods of ContentProvider

- Here is the list of methods which you need to override in Content Provider class to have y Content Provider working.
  - **onCreate() :** This method is called when the provider is started.
  - **query() :** This method receives a request from a client. The result is returned as a Curs object.
  - **insert() :** This method inserts a new record into the content provider.
  - **delete() :** This method deletes an existing record from the content provider.
  - **update() :** This method updates an existing record from the content provider.
  - **getType() :** This method returns the MIME type of the data at the given URI.

6.4.2(A) Example of ContentProvider

- Following is the content of the modified main activity file src/com.exampl MyApplication/MainActivity.java.
- There are two new methods onClickAddName() and onClickRetrieveStudents() to handle use interaction with the application.

```
package com.example.MyApplication;
import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.content.ContentValues;
import android.content.Cursoroader;
import android.database.Cursor;
import android.view.Menu;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
```

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

}

public void onClickAddName(View view){
//Add a new student record
ContentValues values=new ContentValues();
values.put(StudentsProvider.NAME,((EditText)findViewById(R.id.editText2)).getText().toString());

values.put(StudentsProvider.GRADE, ((EditText)findViewById(R.id.editText3)).getText().toSt
ring());

Uri uri=getContentResolver().insert(StudentsProvider.CONTENT_URI,values);
Toast.makeText(getBaseContext(),
uri.toString(),Toast.LENGTH_LONG).show();
}
public void onClickRetrieveStudents(View view){
//Retrieve student records
String URL="content://com.example.MyApplication.StudentsProvider";
Uri students=Uri.parse(URL);
Cursor c  = managedQuery(students, null, null, null, "name");

if(c.moveToFirst()){do{
Toast.makeText(this, c.getString(c.getColumnIndex(StudentsProvider._ID)) +
","+ c.getString(c.getColumnIndex(StudentsProvider.NAME))+
","+ c.getString(c.getColumnIndex(StudentsProvider.GRADE)),Toast.LENGTH_SHORT).show();
}while(c.moveToNext());
}
}
```

Create new file StudentsProvider.java under *com.example.MyApplication*package and following is the content of **src/com.example.MyApplication/StudentsProvider.java** –

```
package com.example.MyApplication;
import java.util.HashMap;
import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
```

```
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteQueryBuIder;
import android.net.Uri;
import android.text.TextUtils;

public class StudentsProvider extends ContentProvider{
static final String PROVIDER_NAME="com.example.MyApplication.StudentsProvider";
static final String URL="content://"+PROVIDER_NAME+"/students";
static final Uri CONTENT_URI=Uri.parse(URL);

static final String_ID= "_id";
static final String NAME="name";
static final String GRADE="grade";
private static HashMap<String,String> STUDENTS_PROJECTION_MAP;
static final int STUDENTS =1;
static final int STUDENT_ID=2;

static final UriMatcher uriMatcher;
static{
uriMatcher=new UriMatcher(UriMatcher.NO_MATCH);
uriMatcher.addURI(PROVIDER_NAME,"students",STUDENTS);
uriMatcher.addURI(PROVIDER_NAME,"students/#",STUDENT_ID);
}
/**
*  Database specific constant declarations
*/
private SQLiteDatabase db;
static final  String DATABASE_NAME="College";
static final String STUDENTS_TABLE_NAME="students";
static final int DATABASE_VERSION =1;
static final String CREATE_DB_TABLE = ""CREATE TABLE "+ STUDENTS_TABLE_NAME+
"(_id INTEGE RPRIMARY KEY AUTOINCREMENT,"+"name TEXT NOT NULL,"+
" grade TEXT NOTNULL);";

/**
*  Helper class that actually creates and manages
*  the provider's underlying data repository.
```

```
private static class DatabaseHelper extends SQLiteOpenHelper{DatabaseHelper(Context context){
super(context, DATABASE_NAME, null, DATABASE_VERSION);


@Override
public void onCreate(SQLiteDatabase db){ db.execSQL(CREATE_DB_TABLE);


@Override
public void onUpgrade(SQLiteDatabase db,int oldVersion,int newVersion){
db.execSQL("DROP TABLE IF EXISTS" +STUDENTS_TABLE_NAME);
onCreate(db);


@Override
public boolean onCreate(){
Context context=getContext();
DatabaseHelper dbHelper=new DatabaseHelper(context);
/**
* Create a write able data base which will trigger its
* creation if it doesn't already exist.
*/
db=dbHelper.getWritableDatabase();
return(db==null)?false:true;

@Override
public Uri insert(Uri uri, ContentValues values){
/**
*Add a new student record
*/
long rowID=db.insert(STUDENTS_TABLE_NAME,"",values);
/**
*If record is added successfully
*/
if(rowID>0){
Uri _uri=ContentUris.withAppendedId(CONTENT_URI,rowID);
getContext().getContentResolver().notifyChange(_uri,null);
return _uri;
}
throw new SQLException("Failed to add a record into"+uri);
```

```
}
@Override
public Cursor query(Uri uri, String[] projection,
String selection, String[] selectionArgs, String sortOrder) {
SQLiteQueryBuilder qb=new SQLiteQueryBuilder();
qb.setTables(STUDENTS_TABLE_ NAME);

switch (uriMatcher.match(uri))
{
case STUDENTS:
qb.setProjectionMap(STUDENTS_PROJECTION_MAP);
break;
case STUDENTID:
qb.appendWhere(_ID +"=" +uri.getPathSegments().get(l));break;
default:
}
if(sortOrder==null ||sortOrder==" ")
/**
*By default sort on student names
*/
sortOrder=NAME;
}
Cursor c=qb.query(db, projection, selectionArgs, null, null, sortOrder);
/**
 selection,
 *register to watch a content URI for changes
*/
c.setNotificationUri(getContext().getContentR  esoIver(),uri);
return c;
}
@Override
public void onCreate(SQLiteDatabase db){db.execSQL(CREATE_DB_TABLE);}
@Override
public void onUpgrade(SQLiteDatabase db,int oldVersion,int newVersion)
{db.execSQL("DROP TABLE IF EXISTS" +STUDENTS_TABLE_NAME);
onCreate(db);
} }
@Override
public boolean onCreate(){Context context=getContext();
DatabaseHelper dbHelper=new DatabaseHelper(context);
/**
```

```
* Create a write able data base which will trigger its
* creation if it doesn't already exist.

*/
db=dbHelper.getWritableDatabase();
return(db==null)?false:true;

}
@Override
public Uri insert(Uri uri, ContentValues values){
/**
*Add a new student record
*/
long rowID=db.insert(STUDENTS_TABLE_NAME,"",values);
/**
*If record is added successfully
*/
if(rowID>0){
Uri _uri=ContentUris.withAppendedId(CONTENT_URI,rowID);
getContext() .getContentResolver().notifyChange(_uri,null);
return _uri;
}
throw new SQLException(""Failed to add a record into" +uri);
}
@Override
public Cursor query(Uri uri, String[]projection,
String selection, String[] selectionArgs, String sortOrder) {
SQLiteQueryBuilder qb=new SQLiteQueryBuilder();
qb.setTables(STUDENTS_TABLE_ NAME);

switch (uriMatcher.match(uri)){
case STUDENTS:
qb.setProjectionMap(STUDENTS_PROJECTION_MAP);
break;
case STUDENT_ID:
qb.appendWhere(_ID +"=" +uri.getPathSegments().get(1));break;
default:
}
if(sortOrder==null ||sortOrder==" "){
/**
*By default sort on student names
*/
sortOrder=NAME;
```

```
}

Cursor c=qb.query(db,projection,selectionArgs,null,null,sortOrder);
/**
register to watch a content URI for changes
*/
c.setNotificationUri(getContext().getContentR  esolver(),uri);
return c;
}
@Override
public int delete(Uri uri, String selection, String[] selectionArgs){
int count=O;
switch(uriMatcher.match(uri)){
case STUDENTS:
count=db.delete(STUDENTS_TABLE_NAME,selection,selectionArgs);break;
case STUDENTID:
Stringid=uri.getPathSegments().get(l);
count=db.delete(STUDENTS_TABLE_NAME,_ID+"="+id+ (!TextUtils.isEmpty(selection)?"
AND("+selection+')':""),selectionArgs);break;
default:
throw new IllegalArgumentException("UnknownURI"+uri);
}
getContext().getContentResolver().notifyChange(uri,null);
return count;
}

@Override
public int update(Uriuri,ContentValues values,String selection,String[] selectionArgs){
int count=0;
switch (uriMatcher.match(uri)){
case STUDENTS:
count=db.update(STUDENTS_TABLE_NAME,values,selection,selectionArgs);
break;

case STUDENT_ID:
count=db.update(STUDENTS_TABLE_NAME,values,
_ID+"="+uri.getPathSegments().get(l)+ (!TextUtils.isEmpty(selection)?"
AND("+selection+')':""),selectionArgs);
break;
default:
throw new IllegalArgumentException("UnknownURI"+uri);
```

```
}
getContext().getContentResolver().notifyChange(uri,null);
return count;

}
@Override
public String getType(Uri uri){switch(uriMatcher.match(uri)){
/**
*Get all student records
*/
case STUDENTS:
return "vnd.android.cursor.dir/vnd.example.students";
/**
*Get a particular student
*/
case STUDENT_ID:
return "vnd.android.cursor.item/vnd.example.students";
default:
throw new IllegalArgumentException("Unsupported URI:"+uri);
}
} }
```

- Following will the modified content of *AndroidManifest.xml* file. Here we have added <provider.../> tag to include our content provider:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http:/schemas.android.com/apk/res/android"
package="com.example.MyApplication">
<application
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:supportsRtl="true"
android:theme="@style/AppTheme">
<activity android:name=".MainActivity">
<intent-filter>
<action android:name="android.intent.action.MAIN"/>
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<provider android:name="StudentsProvider"
android:authorities="com.example.MyApplication.StudentsProvider"/>
</application>
</manifest>
```

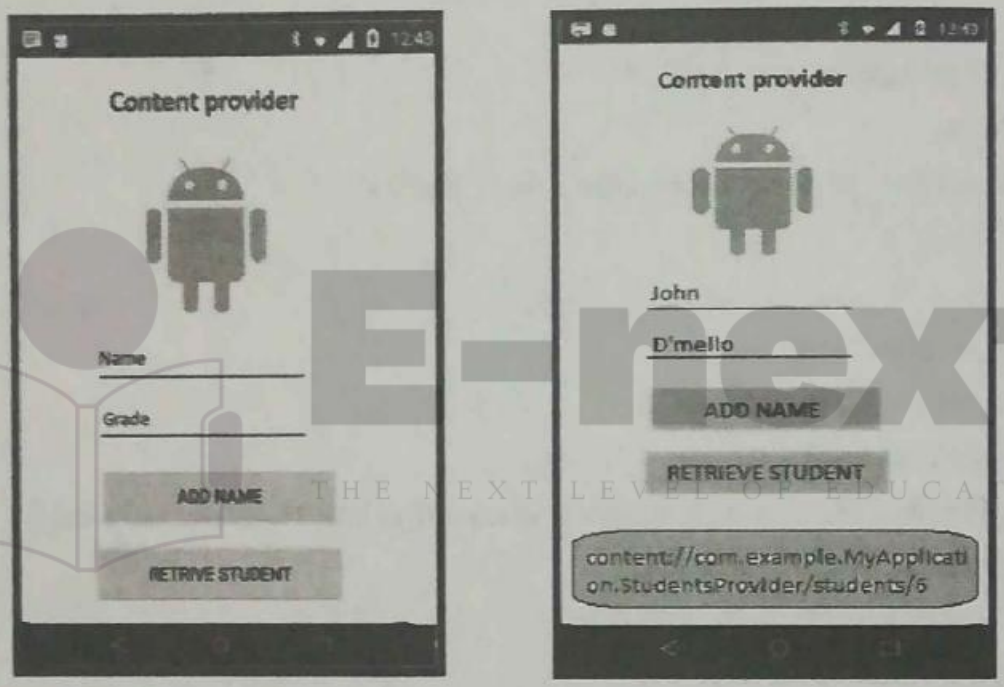
- Following will be the content of **res/layout/activity_main.xml** file.

```
<?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="@dimen/activity_vertical_margin"
android:paddingeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context=" com.example.MyApplication.MainActivity" >

<TextView
android:id="@+id/textViewl"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Contentprovider"
android:layout_alignParentTop="true"
android :layout_centerHorizontal="true"android:textSize="30dp"/>

<lmageButton
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android :id="@+id/imageButton"android:src="@drawable/abe"
android:layout_below="@+id/textView2"
android:layout_centerHorizontal="true"/>

<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/button2" android:text="AddName"
android:layout_below="@+id/editText3"
android:layout_ alignRight="@+id/textView2"
android:layout_alignEnd="@+id/textView2"
android:layout_alignLeft="@+id/textView2"
android:layout_alignStart="@+id/textView2"
android:onClick="onClickAddName"/>

<EditText
Android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

```
droid:id="@+id/editText"
droidlayout_below="@+id/imageButton"
droid:layout_alignRight="@+id/imageButton"
droidlayout_alignEnd="@+id/imageButton"/>

EditText
droid:layout_width="wrap_content"
droid:layout_height="wrap_content"
droid:id="@+id/editText2"
droid:layout_alignTop="@+id/editText"
droid:layout_alignLeft="@+id/textView1"
droid:layout_alignStart="@+id/textView1"
droid:layout_alignRight="@+id/textView1"
droid:layout_alignEnd="@+id/textView1"
droid:hint="Name"
droidtextColorHint="@android:color/holo_blue_light"/>

EditText
ndroid:layout_width="wrap_content"
ndroid:layout_height="wrap_content"
ndroidid="@+id/editText3"
ndroidlayout_below="@+id/editText" android:layout_alignLeft="@+id/editText2"
ndroidlayout_alignStart="@+id/editText2"
ndroidlayout_alignRight="@+id/editText2"
ndroid:layout_alignEnd="@+id/editText2"
ndroid:hint="Grade"
ndroid:textColorHint="@android:color/holo_blue_bright"/>

<Button
ndroid:layout_width="wrap_content"
ndroid:layout_height="wrap_content"
ndroid:text="Retrivestudent"
ndroid:id="@+id/button"
ndroid:layout_below="@+id/button2"
ndroid:layout_alignRight="@+id/editText3"
ndroid:layout_alignEnd="@+id/editText3"
ndroid:layout_alignLeft="@+id/button2"
ndroidlayout_alignStart="@+id/button2"
```

```
android:onClick="onClickRetrieveStudents"/>
</RelativeLayout>
```

- To run the app from Android Studio IDE, open one of your project's activity files and click Run icon from the tool bar. Android Studio installs the app on your AVD and starts it and if everything is fine with your set-up and application, it will display following Emulator window (may take sometime based on your computer speed)(Fig. 6.4.3(a)) .
- Now let's enter student **Name** and **Grade** and finally click on **Add Name**button. This will add student record in the database and will flash a message at the bottom showing ContentProvider URI along with record number added in the database. This operation makes use of our **insert**() method.Lets add some more students in the database of our Content Provider
- Now click on **Retrieve Students** button which will fetch and display all the records one by one which is as per our the implementation of our **query**() method (Fig. 6.4.3(b)).



(a) (b)

Fig. 6.4.3

Syllabus Topic : Loaders to Load and Display Data

6.5 Loaders to Load and Display Data

- One of the major reasons why users abandon apps, is its startup time. Research shows that if an app or page takes more than 3 seconds to load, 40% of users will abandon it.
- The loading time of apps is directly related to whatever happens on the UI thread. The less work the UI thread has to do, the faster the users will see the page.
- There are many factors that affect app startup time.
- One of the big, obvious actions that affect performance is how long it takes for your app to load its data.

The solution is to load most or all of your data in the background, while you show your users relevant information that you have stored locally. For example, you could show them the latest cached weather information, until you have retrieved new data that shows the current weather for the current location.

- Loaders therefore are special purpose classes that manage loading and reloading updated data asynchronously in the background using AsyncTask.
- Loaders were introduced in Android 3.0. They have these characteristics:
  - They are available to every Activity and Fragment.
  - They provide asynchronous loading of data in the background.
  - They monitor the source of their data and automatically deliver new results when the content changes. For example, if you are displaying data in RecyclerView, when the underlying data changes, then a CursorLoader automatically loads an updated set of data, and when finished with loading, can notify the RecyclerView.Adapter to update what it displays to the user.
  - Loaders automatically reconnect to the last loader's cursor when being recreated after a configuration change. Thus, they don't need to re-query their data to display it.

Types of Loaders

1. **AsyncTaskLoader :** _Keeps data available through configuration changes.
2. **CursorLoader :** Works with content providers and databases.
3. **User-defines loader :** Rarely necessary.

6.5.1 CursorLoader with Content Provider

Implementing a CursorLoader

- An application that uses loaders includes the following:
  - An Activity or Fragment.
  - An instance of the LoaderManager.
  - A CursorLoader to load data backed by a ContentProvider. Alternatively, you can implement your own subclass of
  - Loader or AsyncTaskLoader to load data from some other source.
  - An implementation for LoaderManager.LoaderCallbacks. This is where the new loaders are created andwhere our references to existing loaders are managed.
  - A way of displaying the loader's data, such as a SimpleCursorAdapter or RecyclerViewAdapter.
  - A data source, such as a ContentProvider (with a CursorLoader).

LoaderManager

- The LoaderManager is a convenience class that manages all loaders. Only one loader manager is needed per activity. It needs to be got into onCreate() of the activity, where we need to register the loaders that we are going to use.
- The loader manager takes care of registering an observer with the content provider, which receives callbacks when data inthe content provider changes.
- The only calls to the loadermanagerthat we need to make are for registering a loader, and restarting it when we need todiscard all the loaded data.

- The first parameter is the ID of the loader, the second is optional arguments, and the third is thecontext where the callbacks are defined.

```
getLoaderManager().initLoader(0, null, this);
getLoaderManager().restartLoader(0, null, this);
```

☞ **LoaderManager.LoaderCallbacks**

- In order to interact with the loader, the activity has to implement a set of callbacks specified in the LoaderCallbacksinterface of the LoaderManager.
- When the state of the loader changes, these methods are called accordingly.

☞ **Methods of LoaderManager**

- **onCreateLoader() :** Called when a new loader is created. Associates loader with the data source it should load and observe. (You don't have to do anything additional for the loader to observer your data source.)
- **onLoadFinished() :** Called every time the loader finishes loading. Trigger an update of user-visible data in this method.
- **onLoaderReset() :** When the loader is reset, you usually want to invalidate the currently held data until new data has been loaded.

- To implement these callbacks, we need to implement LoaderManager callbacks for the type of loader we have. For acursor loader, change the signature of our activity as follows, then implement the callbacks.

```
public class MainActivity extends AppCompatActivity implements
LoaderManager.LoaderCallbacks<Cursor>
```

☞ **onCreateLoader()**

- This callback instantiates and returns a new loader instance of the desired type. Since the loader manager can bemanaging multiple loaders, an ID argument identifies the loader to instantiate. Once created, the loader will start loadingdata, and it will observe the date for changes, and reload as necessary.
- To create CursorLoader, you need
  - **uri :**The URI for the content to retrieve from the content provider. This identifies the content provider and the data toobserve to the loader.
  - **projection :** A list of columns to return. Passing null will return all columns.
  - **selection :** A filter declaring which rows to return, formatted as an SQL WHERE clause (excluding the WHERE itself).Passing null will return all rows for the given URI.
  - **selectionArgs :** You may include ?s in the selection, which will be replaced by the values from selectionArgs, in theorder that they appear in the selection. The values will be bound as Strings.
  - **sortOrder :** How to order the rows, formatted as an SQL ORDER BY clause (excluding the ORDER BY itself).Passing null will use the default sort order, which may be unordered.

```
@Override
public Loader<Cursor>onCreateLoader(int id, Bundle args) {
String queryUri = CONTENT_URI.toString();
String[] projection = new String[] {CONTENT_PATH};
```

```
return new CursorLoader(this, Uri.parse(queryUri),
projection, null, null, null);
}
```

- Notice how this is very similar to initiating a content resolver:

```
Cursor cursor = mContext.getContentResolver().query(Uri.parse(uri),
projection, selectionClause, selectionArgs, sortOrder);
```

onLoadFinished()

- Here we specify what happens with the data once the loader has acquired it. In this function we should:
  - Release the old data.
  - Save the new data and, for example, make it available to your adapter.
- The cursor loader monitors the data for us, so we should never do it ourselves.
- The loader also cleans up after itself, so there is no need for us to close the cursor.If we are using a RecyclerView to display the data, all we need to do is hand the data over to the adapter whenever the loading or reloading has finished.

```
@Override
public void onLoadFinished(Loader<Cursor> loader, Cursor cursor) {
mAdapter.setData(cursor);
}
```

onLoaderReset()

- This method is called when a previously created loader is being reset, and thus making its data unavailable. We should clean allreferences to the data at this point. Again, if we are passing the data to an adapter for display in a RecyclerView, theadapter does the actual work, we just have to instruct it to do so.

```
@Override
public void onLoaderReset(Loader<Cursor> loader) {
mAdapter.setData(null);
}
```

Using the data returned by the loader

- Oncethe loader receives the data, it hands the data over to the adapter through a setData() call. The setData()method updates an instance variable in the adapter that holds the most current data set, and notifies the adapter that thereis fresh data.

```
public void setData(Cursor cursor) {
mCursor = cursor;
notifyDataSetChanged();
}
```

Benefits of cursors

- You may have noticed that the database uses cursors, the content provider uses cursors, and the loader uses cursors.

- Using the same data type throughout your backend, and only unpacking it in the adapter, where the contents of the cursorare prepared for display, makes for a uniform backend with clean interfaces. This makes it easier to write, test and debug the code. It also makes the code simpler and shorter.

☞ **Complete app with methods**

- The following diagram shows the methods and data types that connect the different parts of an application that uses:
  - An SQLite database to store data, and an SQLiteOpenHelper subclass to manage the database.
  - A content provider to make data available to this (and other) apps.
  - A loader to load data to display to the user.
  - A RecyclerView.Adapter that displays and updates data shown to the user in a RecyclerView.
- Note the call stack and journey of the cursor through the layers of the application for a query().
- Inserting, deleting, and updating are still handled by the content resolver. However, the loader will notice anychanges made by insert, delete, or update operations, and will reload the data as necessary.

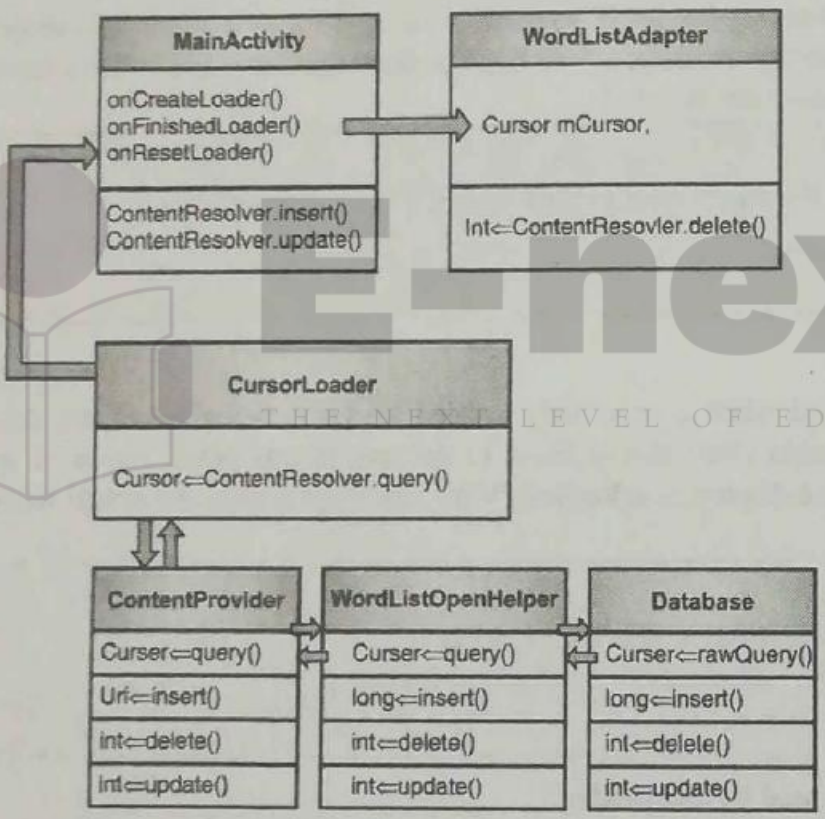


Fig. 6.5.1

Syllabus Topic : Permissions

6.6 Permissions

- As we work through the programs, there were times when our app required permission to do something,including when it needed to:
  - Connect to the Internet.
  - Use a content provider in another app.

This section gives an overview of permissions, how and when to ask for permission when your app needs it before it can perform an action.

Ask permission if it isn't yours

An app is free to use any resources or data that it creates, but must get permission to use anything—data, resources/hardware, software—that does not belong to it.

For example, your app must get permission to read the user's Contactsdata, or to use the device's camera. This is because the camera hardware does not belong to the app, and the app must always get permission to use anything that is not part of the app itself.

Requesting permission

To request permission, add the `<uses-permission>` attribute to the Android manifest file, along with the name of therequested permission.

For example, to get permission to use the camera:

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

Examples of permissions

The Android framework provides more than 100 predefined permissions. These include some obvious things includingpermission to access or write the user's personal data such as:

- reading and writing a user's Contacts list, calendar, or voicemail
- accessing the device's location
- accessing data from body sensors

Some of the other pre-defined permissions are less obvious, such as permission to collect battery statistics, permission toconnect to the internet, and permission to use hardware such as the camera or fingerprint hardware.

Android includes pre-defined permissions for initiating a phone call without requiring the user to confirm it, reading the calllog, capturing video output, rebooting the device, changing the date and time zone, and many more.

You can see all the system-defined permissions at:

https://developer.android.com/reference/android/Manifest.permission.html.

Normal and Dangerous Permissions

Android classifies permissions as normal or dangerous.

A **normal permission** is for actions that do not affect user privacy or user data, such as connecting to the internet.

A **dangerous permission** is for an action that does affect user privacy or user data, such as permission to write to the user's voicemail.

Android automatically grants normal permissions but asks the user to explicitly grant dangerous permissions.

> Apps must list all permissions they use in the Android manifest, even normal permissions.

How users grant and revoke permissions?

- The way users grant and revoke permissions depends on:
  - The version of Android that the device is running.
  - The version of Android that the app was created for.

Fig. 6.6.1

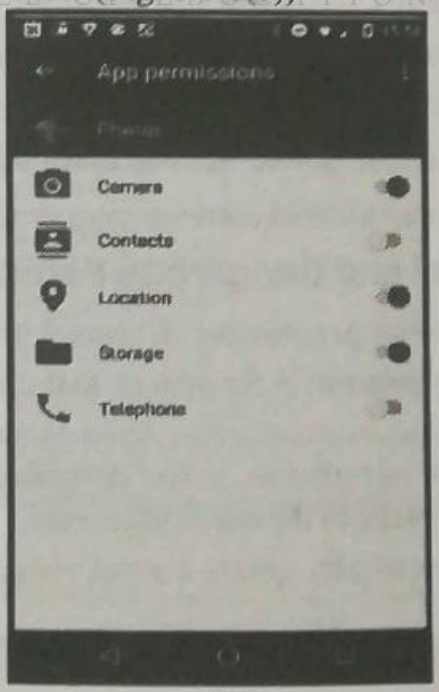
Before Marshmallow (Android 6.0)

- If an app was *created* for a version of Android before 6.0 (Marshmallow) ***or*** it is *running* on a device that uses a version ofAndroid before Marshmallow, Google Play asks the user to grant required dangerous permissions before installing the app (Fig. 6.6.1).
- If the user changes their mind and wants to deny permissions to the app after it is installed, the only thing they can do is to uninstall the app.

Marshmallow onwards

- If an app was created for a version of Android from Android 6.0 (Marshmallow) onwards ***and*** it is running on a device that uses a version of Android from Marshmallow onwards, then Google Play does not ask the user to grant dangerous permissions to the app before installing it.
- Instead, when the user starts to do something in the app that needs that level of permission, Android shows a dialog box asking the user to grant permission. (Fig. 6.6.2(a))
- The user can grant or revoke individual permissions at any time. They do this by going to the Settings App, choosing Apps, and selecting the relevant app. In the Permissions section, they can enable or disable any of the permissions that the app uses. (Fig. 6.6.2(b))

(a)



(b)

Fig. 6.6.2

Q. How differences in the permissions models affect developers?

- In the "old" permissions model, Google Play and the Android Framework worked together to get permission from the user.All that the developer needed to do was to make sure that the app listed the permissions it needed in the Android manifestfile.
- The developer could assume that if the app was running, then the user had granted permission. The developer did notneed to write code to check if permission had been granted or not.
- In the "new" permissions model, you can no longer assume that if the app is running, then the user has granted the neededpermissions.
- The user could grant permission the first time they run the app, then, at any time, change their mind andrevoke any or all of the permissions that the app needs.So, the app must check whether it still has permission every time it does something that requires permission.
- The Android SDK includes APIs for checking if permission has been granted. Here is a code snippet that checks if the app haspermission to write to the user's calendar:

```
// assume this Activity is the current activity
int permissionCheck = ContextCompat.checkSelfPermission(thisActivity,
Manifest.permission.WRITE_CALENDAR);
```

- The Android framework for Android 6.0 (API level 23) includes methods for checking for and requesting permissions.
- The Support Library also includes methods for checking for and requesting permission.
- We recommend that you use the support library methods for handling permissions, because the permission methods in the support library take care of checking which version of Android your app is running on, and taking the appropriate action.
- For example, if the user's device is running an older version, then the checkSelfPermission() method in the support library checks if the user already granted permission at runtime, but if the device is running Marshmallow or later, then it checks if permission is still granted, and if not, shows the dialog to the user to ask for permission.

Best practices for permissions

- When an app asks for too many permissions, users get suspicious. Make sure your app only requests permission for features and tasks it really needs, and make sure the user understands why they are needed.
- Wherever possible, use an Intent instead of asking for permission to do it yourself. For example, if your app needs to use the camera, send an Intent to the camera app, and that way the camera app will do all your work for you and your app does not need to get permission to use the camera (and it will be much easier for you to write the code than if you accessed the camera APIs directly).

Syllabus Topic : Performance

6.7 Performance

- In order to make your app stand out from the crowd,you should also make it as small, fast, and efficient as possible. We must consider the impact the app might have on the device'sbattery, memory, and disc space. Most of all, we need to be considerate of users' data-plans.

There are the following recommendations, where performance is concerned, on where to start:

- **Important:** Maximizing performance is all about balance, finding and making the best trade-offs between app complexity,functionality, and visuals, to give users the best possible experience with your app.

☞ **Keep long-running tasks off the main thread**

- We already know that we need to take work off the main thread into the background to help keep the UI smooth and responsive for the user.
- The hardware that renders the display to the screen typically updates the screen every 16milliseconds, so if the main thread is doing work that takes longer than 16 milliseconds, the app might skip frames, stutterer hang, all of which are likely to annoy the users.
- We can check how well our app does at rendering screens within the 16 millisecond limit by using the **Profile GPURendering** tool on our Android device.
  1. Go to Settings >Developer options.
  2. Scroll down to the Monitoring section.
  3. Select Profile GPU rendering.
  4. Choose On screen as bars in the dialog box.
- Immediately you will start seeing colored bars on your screen.

Note: You can run the tool on an emulator, but the data is not indicative of how your app would perform on areal device.

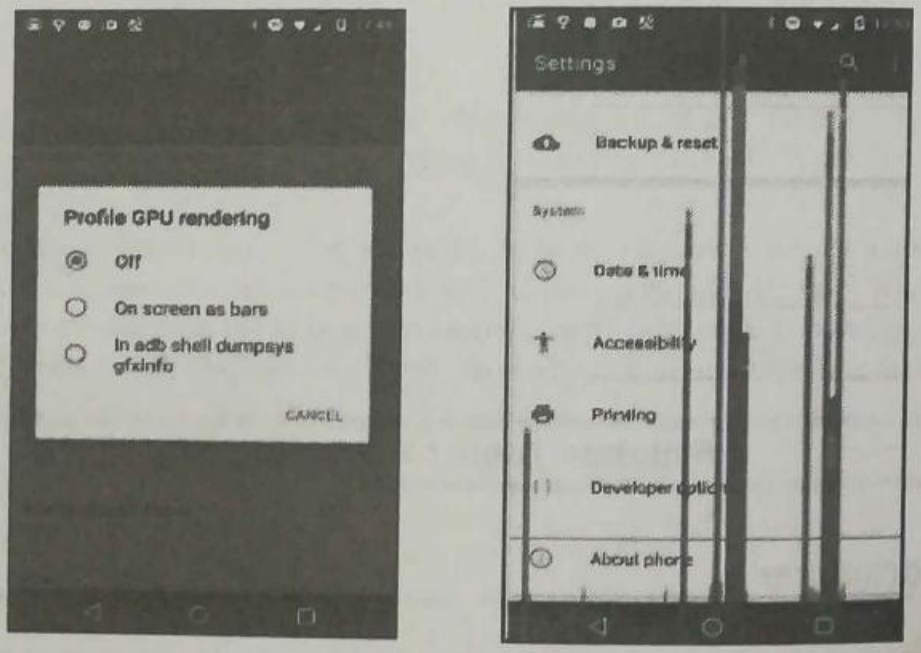


Fig. 6.7.1

- Open the app, and watch the colored bars (Fig. 6.7.2).

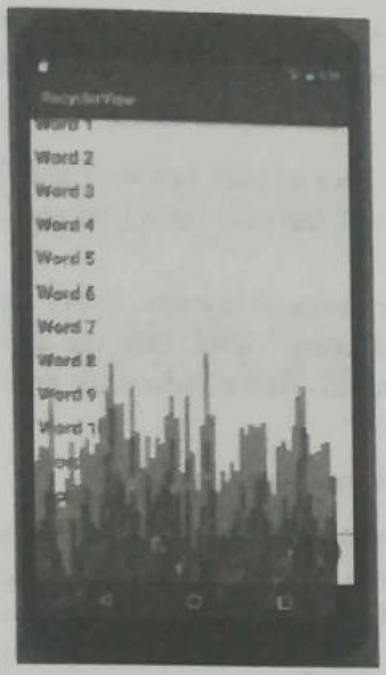


Fig.6.7.2

One bar represents one screen rendered. If a bar goes above the green line, it took more than 16 ms to render. The colors in the bar represent the different stages in rendering the screen. (Fig. 6.7.3)

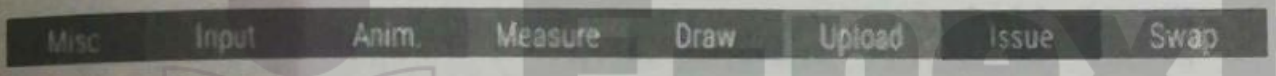


Fig. 6.7.3

Spending time using the Profile GPU rendering tool on your app, it will help you identify which parts of the UI interaction are slower than might be expected, and then you can take action to improve the speed of your app's UI.

For example, if the green Input portion of the bar is large, your app spends a lot of time handling input events, that is, executing code called as a result of input event callbacks. To fix this, consider when and how you request user input, and whether you can handle it more efficiently.

Simplify your UI

The layouts will draw faster and use less power and battery if we spend time designing them in the most efficient way.

Try to avoid

- **Deeply nested layouts :** If our layouts are narrow and deep, the Android system has to perform more passes to layout all the views than if our view hierarchy is wide and shallow. Consider how we can combine, flatten, or eveneliminate views.
- **Overlapping views :** This results in "overdraw" where the app wastes time drawing the same pixel multiple times, andonly the final rendition is visible to the user. Consider how you can size and organize your views so that every pixel isonly drawn once or twice.

Simplify layouts

Make sure your layouts include only the views and functionality your app needs. Simple layouts are more visually appealing to users, and they draw faster. Flatten the layouts as much possible, i.e. reduce the number of nested levels in your app's view hierarchy.

- For example, if your layout contains a LinearLayout inside a LinearLayout inside a LinearLayout, instead you may be able to arrange all the views inside a single ConstraintLayout.

☞ **Minimize overlapping views**

- Imagine that you are painting the door of your house in red. They you paint it again in green. Then you paint it again in blue. In the end, the only color you see is blue, but you wasted a lot of energy painting the door multiple times.
- Each layout in your app hides the previous layouts. Every time the app "paints" (draws) a pixel, it takes time. If the layout has overlapping views, then the app is using time and resources drawing pixels that it then draws all over again. Hence reduce the amount of time the app overdraws pixels, by reducing overlapping views.

☞ **Monitor the performance of your running app**

- Android Studio has tools to measure your app's memory usage, GPU, CPU, and Network performance. App crashes are often related to memory leaks, which is when your app allocates memory and does not release it. If your app leaks memory, or uses more memory than the devices makes available, it will eventually use up all the available memory on the device.
- Use the Memory Monitor tool that comes with Android Studio to observe how your app uses memory.

1. In Android Studio, at the bottom of the window, click the **Android Monitor** tab. By default this opens on logcat.
2. Click the **Monitors** tab next to logcat. Scroll or make the window larger to see all four monitors: Memory, CPU, Networking, and GPU.
3. Run your app and interact with it. The monitors update to reflect the app's use of resources. Note that to get accurate data, you should do this on a physical, not virtual, device.

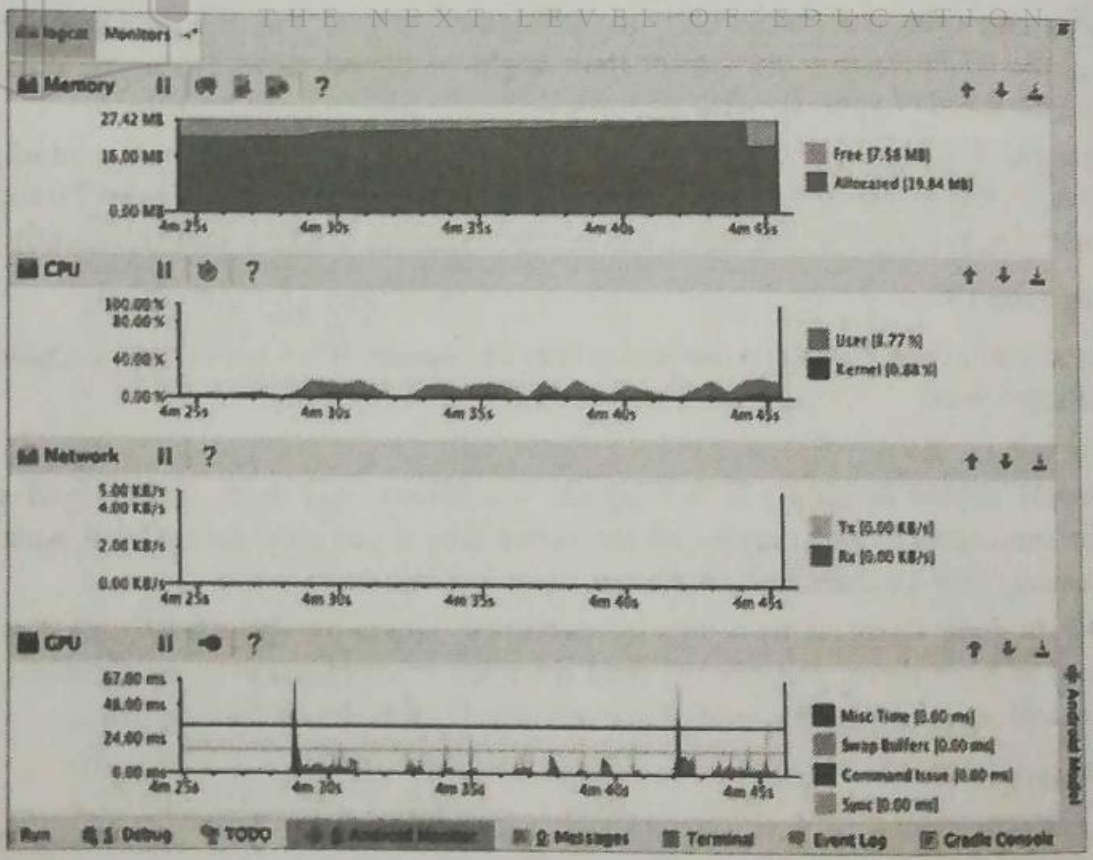


Fig. 6.7.4

Types of monitors

1. **Memory monitor** : Reports how your app allocates memory and helps you to visualize the memory your app uses.
2. **CPU monitor** : Lets you monitor the central processing unit (CPU) usage of your app. It displays CPU usage in real time.
3. **GPU monitor** : Gives a visual representation of how long the graphical processing unit (GPU) takes to render frames to the screen.

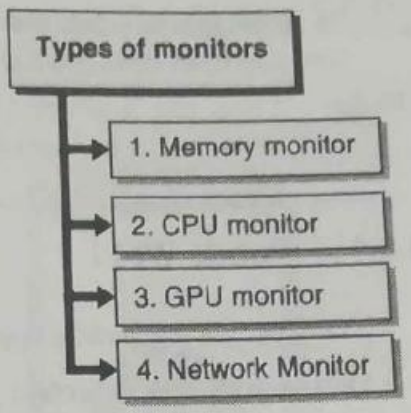


Fig. C. 6.2 : Types of monitors

4. **Network Monitor** : Shows when your application is making network requests. It lets you see how and when your app transfers data, and optimize the underlying code appropriately.

Syllabus Topic : Security

6.8 Security

Most of the burden of building secure apps is handled by the Android Framework.

For example, apps are isolated from each other so they can't access each other or use each other's data without permission.

However, as an app developer, you have the responsibility to make sure your app treats the user's data safely and with integrity. Your app is also responsible for keeping its own data safe.

Handling user data

How to handle data in Android?

We have already seen how Android uses permissions to make sure apps cannot access the user's personal data without their permission. However, even if the user gives your app permission to access their private data, do not do sounless absolutely necessary. If you do, treat the data with integrity and respect.

For example, just because the user gives your app permission to update their calendar does not mean you have permission to delete all their calendar entries.

Android apps operate on a foundation of implied trust. The users trust that the apps will use their data in a way that makes sense within the context of the app.

If your app is a messaging app, it's likely that the user will grant it permission to read their contacts. That does not mean your app is allowed to read all the user's contacts and send everyone a spam message.

Your app must only read and write the user's data when absolutely necessary, and only in a way that the user would expect the app to do so. Once your app has read any private data, you must keep it safe and prevent any leakage. Do not share private data with other apps.

Depending on how your app uses user data, you might also need to provide a written statement regarding privacy practices when you publish your app in the Google Play store.

Types of monitors

1. **Memory monitor** : Reports how your app allocates memory and helps you to visualize the memory your app uses.
2. **CPU monitor** : Lets you monitor the central processing unit (CPU) usage of your app. It displays CPU usage in real time.
3. **GPU monitor** : Gives a visual representation of how long the graphical processing unit (GPU) takes to render frames to the screen.

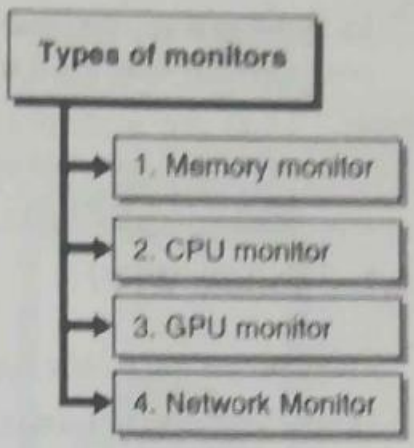


Fig. C. 6.2 : Types of monitors

4. **Network Monitor** : Shows when your application is making network requests. It lets you see how and when your app transfers data, and optimize the underlying code appropriately.

Syllabus Topic : Security

6.8 Security

Most of the burden of building secure apps is handled by the Android Framework.

For example, apps are isolated from each other so they can't access each other or use each other's data without permission.

However, as an app developer, you have the responsibility to make sure your app treats the user's data safely and with integrity. Your app is also responsible for keeping its own data safe.

Handling user data

Q. How to handle data in Android?

We have already seen how Android uses permissions to make sure apps cannot access the user's personal data without their permission. However, even if the user gives your app permission to access their private data, do not do sounless absolutely necessary. If you do, treat the data with integrity and respect.

For example, just because the user gives your app permission to update their calendar does not mean you have permission to delete all their calendar entries.

Android apps operate on a foundation of implied trust. The users trust that the apps will use their data in a way that makes sense within the context of the app.

If your app is a messaging app, it's likely that the user will grant it permission to read their contacts. That does not mean your app is allowed to read all the user's contacts and send everyone a spam message.

Your app must only read and write the user's data when absolutely necessary, and only in a way that the user would expect the app to do so. Once your app has read any private data, you must keep it safe and prevent any leakage. Do not share private data with other apps.

Depending on how your app uses user data, you might also need to provide a written statement regarding privacy practices when you publish your app in the Google Play store.

- Be aware that any data that the user acquires, downloads, or buys in your app belongs to them, and your app must store itin a way that the user still has access to it even if they uninstall your app.

Note: Logs are a shared resource across all apps. Any app that has the READ_LOGS permission can read all the logs. Do **not** write any of the user's private data to the logs.

6.8.1 Public Wi-Fi

- Many people use mobile apps over public Wi-Fi.
- Design your app to protect your user's data when they are connected on public Wi-Fi. Use https rather than http whenever possible to connect to websites. Encrypt any user data that gets transmitted, even data that might seem innocent like their name.
- For transmitting sensitive data, implement authenticated, encrypted socket-level communication using the SSLSocketclass. This class adds a layer of security protections over the underlying network transport protocol. Those protections include protection against modifications of messages by a wire tapper, enhanced authentication with the server, and increased privacy protection.

6.8.2 Validating User Input

- If your app accepts input (and almost every app does!) you need to make sure that the input does not bring anything harmful in with it.
- If your app uses native code, reads data from files, receives data over the network, or receives data from any external source, it has the potential to introduce a security issue. The most common problems are buffer overflows, dangling pointers, and off-by-one errors.
- Android provides a number of technologies that reduce the exploitability of these errors, butthey do not solve the underlying problem. You can prevent these vulnerabilities by carefully handling pointers and managing buffers.
- If your app allows users to enter queries that are submitted to an SQL database or a content provider, you must guard against SQL injection. This is a technique where malicious users can inject SQL commands into a SQL statement by entering data in a field. Injected SQL commands can alter SQL statement and compromise the security of the application and the database.
- Another thing you can do to limit the risk of SQL injection is to use or grant either the READ_ONLY or WRITE_ONLYpermission for content providers.

Syllabus Topic : Firebase

6.9 Firebase

- Firebase is a set of tools for app developers. It is a platform that provides tools to help us:
  1. Develop our app 2. Grow our user base 3. Earn money from our app

- Firebase features are available for
  1. Android 2. iOS 3. Web

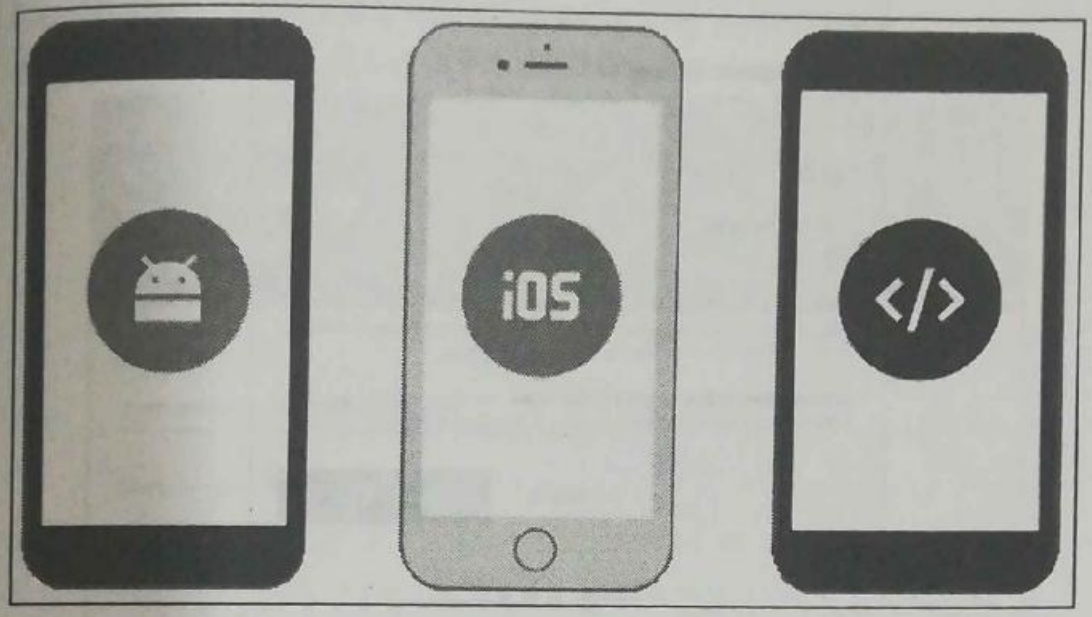


Fig. 6.9.1

- **Get started with Firebase**
- Firebase Console is a web front end for managing your Firebase projects
- To use Firebase, go to the Firebase console at https://console.firebase.google.com/. For this you will need to have a Google account.
- The first time you go to the Firebase you see a welcome screen that includes a button to create a new project.

Fig. 6.9.2

- To use Firebase features with your Android app, first create a Firebase project and then add your Android app to theFirebase project.
- A project is a container for your apps across platforms: Android, iOS, and web. You can name your project anything youwant; it does not have to match the name of any apps.
  1. In the Firebase console, click the **CREATE NEW PROJECT** button.
  2. Enter your Firebase project name in the dialog box that appears then click **Create Project.**

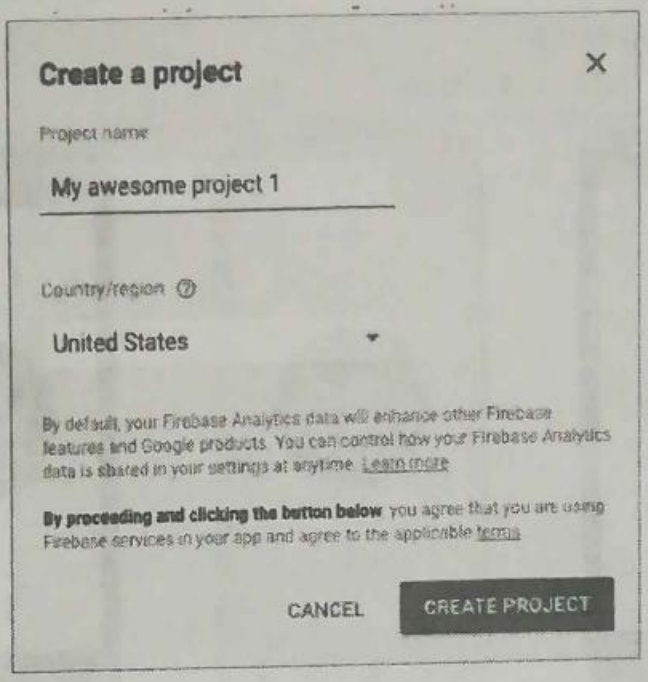


Fig. 6.9.3

The Firebase console opens. Look in the menu bar for the name of your project.

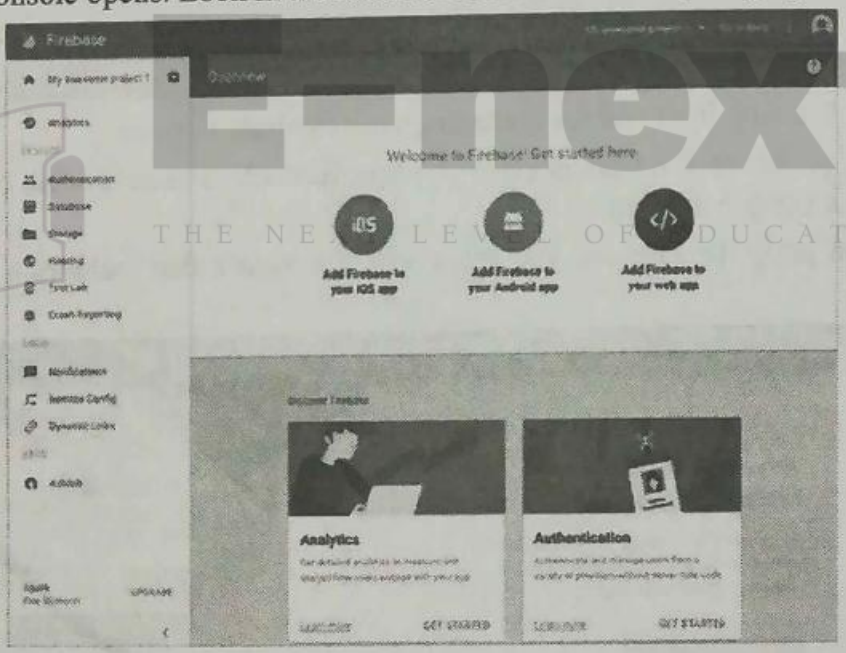


Fig. 6.9.4

6.9.1 Add Your Android App to Your Firebase Project

- The next step is to associate your Android application with your Firebase project. First ge information you will need.
- To connect Firebase to your Android app, you need to know the package name used in your It's best to havey our app open in Android Studio before you start the process.
- To connect your Firebase project and your Android app to each other:
  1. In Android Studio, open your Android app.
  2. Make sure you have the latest Google Play services installed.

3. **Tools > Android SDK Manager > SDK Tools tab > Google Play services.**
4. Note the package of the source code in your Android app.
5. In the Firebase console, Click **Add Firebase to your Android app.**
6. Enter the package name of your app in the dialog box that appears.

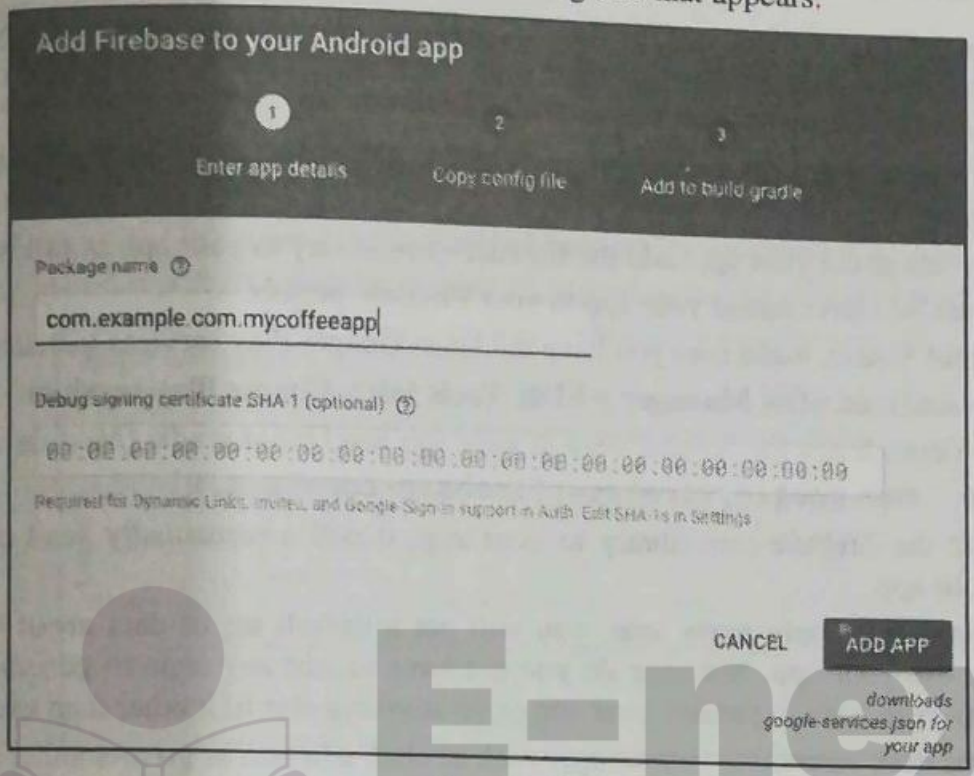


Fig. 6.9.5

- Click **Add App**. Firebase downloads a file called google_services.json to your computer. Save it in a convenient location, such as your Desktop. (If you have any trouble, click Project Settings (the cogwheel next to Overview) and download the file manually.)
- Find the downloaded file on your computer.
- Copy or move that file into the **app** folder of your project in Android Studio. The wizard in Firebase shows where to put the file in Android Studio.
- In the Firebase console, click **Continue**. The screen now shows the instructions for updating your build.gradle files.
- In Android Studio, update the project-level build.gradle (Project: app) file with the latest version of the google-services package (where x.x.x are the numbers for the latest version. See the Android setup guide for the latest version.)

```
buildscript {
dependencies {
// Add this line
classpath 'com.google.gms:google-services:x.x.x'
}
}
```

- In Android Studio, in the app-level build.gradle (Module: app) file, at the bottom, apply the google-services plugin.

```
// Add to the bottom of the file apply plugin: 'com.google.gms.google-services'
```

- In Android Studio, sync the Gradle files.
- In the Firebase console, click Finish.
- Your Android app and your Firebase project are now connected to each other.

6.9.2 Firebase Analytics

- You can enable Firebase Analytics in your app to see data about how and where your app is used. You will be able to see data such as how many people use your app over time and where in the world they use it.
- All the data that your app sends to Firebase is anonymized, so you never see the actual identity of *who* used your app.
- To get usage data about your app, add the firebase-core library to your app as follows:
  1. Make sure you have added your app to your Firebase project.
  2. In Android Studio, make sure you have the latest Google Play services installed.
     Tools > Android SDK Manager > SDK Tools tab > Google Play services
  3. Add the dependency for firebase-core to your app-level build.gradle (Module: app) file:
     `compile 'com.google.firebase:firebase-core:x.x.x'`
- After you add the firebase-core library to your app, it will automatically send usage data when people use your app.
- Without having to add any more code, you will get a default set of data about how, when, and where people use your app. Not only do you not have to add any code to generate default usage data, you don't even have to publish your app or do anything else to it other than use it.
- When someone does something in your app, such as click a button or go to another activity, the app will *log an Analyticsevent*. This means that the app will package up the data about the event that happened, and put it in the queue to send to Firebase.

6.9.3 Check to See if Analytics Event Occurs

- Android sends Analytics events in batches to minimize battery and network usage, as explained in this blog post. Generally speaking, Analytics events are sent approximately every hour or so to the server, but it also takes additional time for the
- Analytics servers to process the data and make it available to reports in the Firebase console.
- While you are developing and testing your app, you don't have to wait for the data to show up in the Analytics dashboard tocheck that your app is logging Analytics events. As you use your app, make sure your logcat is displaying "Debug"messages. Look in the log for statements such as:

```
Logging event (FE): _e, Bundle [{_o=auto, _et=5388, _sc=SecondActivity, ...}]
```

- These kinds of log messages indicate that an Analytics event has been generated. In this example, an Analytics event wasgenerated when the SecondActivity started.

☞ **View reports in the Firebase Analytics dashboard**

- To see the Analytics reports, open the Firebase Console and select **Analytics**. The dashboard opens showing reports for your app for the last 30 days (Fig. 6.9.6).

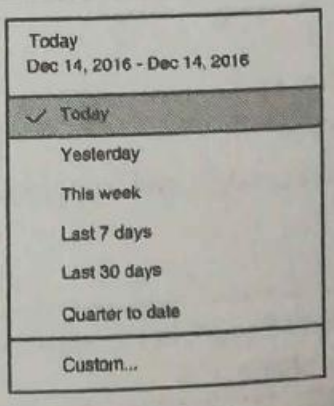


Fig. 6.9.6

Note: The end date is always Yesterday by default. To see the usage statistics including **Today**, change the default date range to **Today**. Look for the calendar icon towards the top right of the screen and change the date range.

Default Analytics

The analytics information you get by default includes:

1. Number of users
2. Devices your users used
3. Location of the users
4. Demographics such as gender
5. App version
6. User engagement
7. And more

For a full list of the available reports see the Firebase help for the Dashboard.

Here's an example of a report showing the number of people who used an app in the past 30 days:

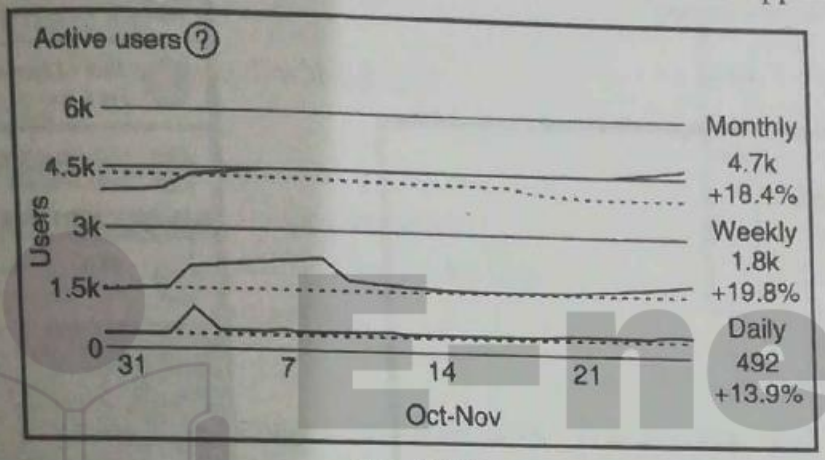


Fig. 6.9.7

And here's an example of a report showing the users' locations.

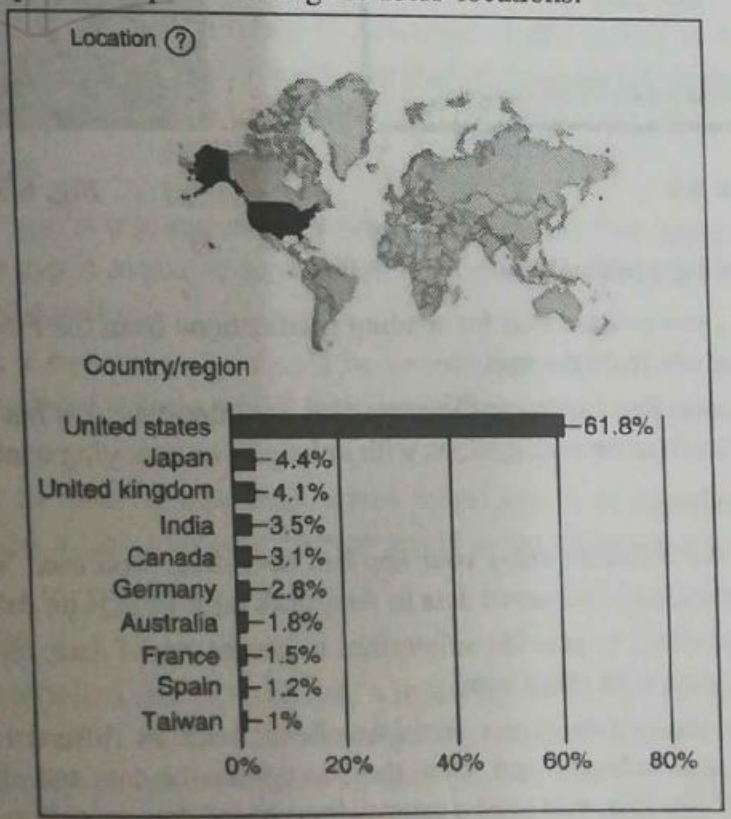


Fig. 6.9.8

6.9.4 Firebase Notifications

- You learned in a previous lesson how to enable *your app* to send notifications to the user. Using the Firebase console, *you* can send notifications to all your users, or to a subset of your users.
- Type the message in the Notifications section of the console, select the audience segment to send the message to, and then send the message.
- To send the message to all users of your app, set the **User Segment** to **App**, and select the package for your app.
- Behind the scenes, Firebase uses Firebase Cloud Messaging to send the notification to the targeted devices where the selected app is installed.

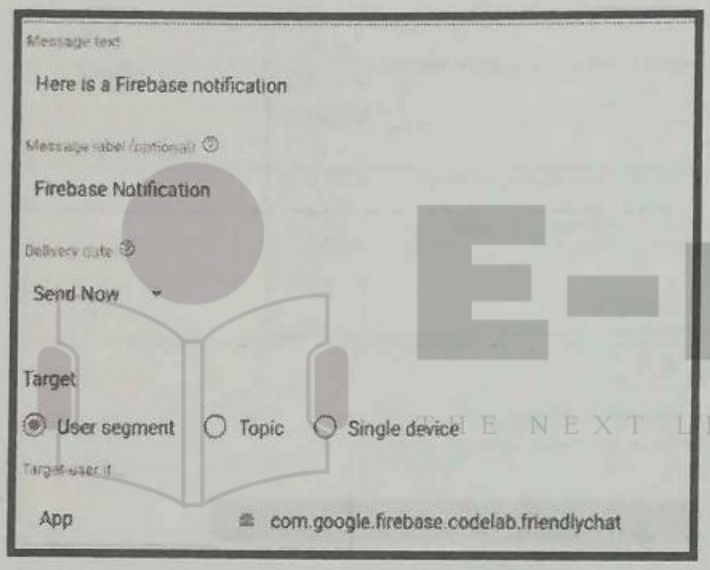


Fig. 6.9.9

Fig. 6.9.10

☞ Best practices for sending notifications

- The same kind of best practices are true for sending notifications from the Firebase console as they are for sending notifications from the app.
- Be considerate of the user. Send only notifications that are important. Do not irritate your usersby sending too many notifications or notifications with unhelpful or annoying content.

☞ Firebase Realtime Database

- So far, you've learned the different ways your app can save data. You used SharedPreferences to save data as key-valuepairs, and you saved data in Android's built-in SQLite database.
- You created a content provider to provide aninterface to access stored data, regardless of how it is stored, and also to share data with other apps.
- However, how do you share data across multiple clients such as different devices and apps, including Android, iOS and webapps, and allow them to update the data and all stay synchronized with the data in realtime? For this, you need a centralcloud-based data repository.

Firebase offers a database that provides cloud-based data storage, allowing clients to all stay in sync as the data changes.If an app goes offline, the data remains available. When the app reconnects to the Internet, the data is synced to the lateststate of the database.

Make money from your app

It's exciting to build an app and see it run. But how do you make money from your app?

First, you need to create an app that works well, is fast enough, doesn't crash, and is useful or entertaining. It must becompelling so users will not only want to install and use it, but will want to keep on using it.

Assuming your app is robust and provides features that are useful, entertaining, or interesting, there are various ways foryou to make money from your app.

Ways to make money

The monetization models are

- Premium model : Users pay to download app.
- Freemium model
  1. downloading the app is free.
  2. users pay for upgrades or in-app purchases.
- **Subscriptions :** Users pay a recurring fee for the app.
- Ads : The app is free but it displays ads.

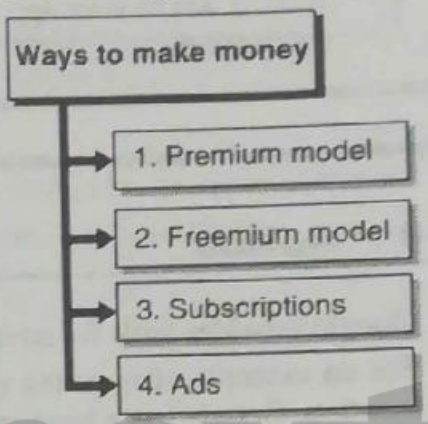


Fig. C6.3 : Ways to make money

1. Premium apps

- Users pay up front to download premium apps. For an app that provides desirable functionality to a small, highly targeted audience, attaching a price to your app can provide a source of revenue.
- Be aware that some users will refuse to download an app if they have to pay for it, or if they cannot try it first free of charge. If users can find other similar apps that are available free or at a lower cost, they might prefer to download and try those other apps.

2. Freemium apps

- A "freemium" app is a compromise between a completely free app and one that charges a fee to install. The app is available for installation free of charge either with limited functionality or for a limited duration.
- Your goal for a freemium app should be to convince users of the value of your app, so that after they have used it for a while, they are willing to pay to keep using it or to upgrade for more features.
- Another way to make money from a free mium app is to provide in-app purchases. For a game, your app might offer new content levels in the game, or new items to make the game more fun.

3. Subscriptions

- With the subscription model, users pay a recurring fee to use the app. This is very similar to the premium model, except that users pay at regular billing cycles rather than once at installation time. You can set up subscriptions so that users pay either monthly or annually.
- If you provide your app on a subscription basis, consider offering regular content updates or some other service that warrants a recurring fee.

- If you decide to offer your app on the subscription model, it's a good idea to let users have a free trial. Many users will ref use to install an app that makes them pay before they have even tried it to see if it suits their needs.

➜ **4. Ads**

- One common monetization strategy is to provide your app free of charge but run ads in it. The user can use your app asmuch as they like, but the app will show them ads occasionally.
- If your app displays ads, always be considerate of the user. If your app shows so many ads that it annoys the user, theymight stop using it or uninstall it.
- Adding ads to your app is straightforward. The best way to incorporate ads into your app is to use AdMob.

Syllabus Topic : AdMob

6.10 AdMob

- Google provides tools for advertisers to create ads and define the targeting criteria for their ads.
- For an example of targeting criteria, an ad might be targeted to be shown to people in a specific location. Google has a huge inventory of adsto display on both websites and mobile apps. You can display ads from this inventory in your app by using AdMob (whichstands for Ads on Mobile).
- To display an ad in your app, add an AdView in the layout of an activity and write a small amount of boilerplate code to loadthe ad. When a user runs your app and goes to that activity, an ad appears in the AdView. You do not need to worry aboutfinding an ad to display because Google handles that for you.

Q. How ads help you make money?

- Google pays you when users click ads in your app.
- The exact amount you get paid depends on the ads that get shown, andhow much the advertisers were willing to pay for their ads.
- The total amount paid by the advertisers is distributed betweenGoogle and the publisher of the website or app where the ad appears.

☞ **Don't click on ads in your own app**

- Google has policies that prevent website publishers and app publishers from clicking ads in their own websites and apps.
- The advertisers pay when people click their ads, so it would not be fair for you to display an ad in your app, then click thead, and cause the advertiser to pay you for clicking the ad in your own app.

6.10.1 Create an AdMob Account

- Before you can experiment with running ads in your app, you need to enable AdMob for that app. To enable AdMob usethese steps:
  1. In the Firebase console, select **AdMobin** the left hand navigation, then select **Sign Up For AdMob**.

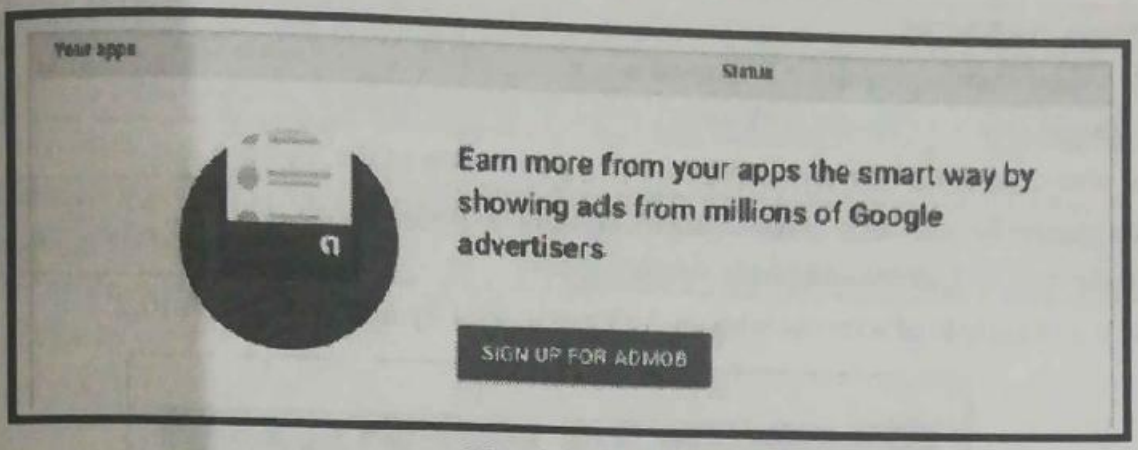


Fig. 6.10.1

2. You will be re-directed to the AdMob console.
3. Follow the sign up wizard to create your AdMob account and add your app to AdMob.

- To display an ad in your app, you need your AdMob app ID and an ad unit ID. You can get both of these in the AdMobconsole.

☞ **Implement AdMob in your app**

- To display an ad in your app:
  1. Make sure you have added your app to your Firebase project.
  2. Add the dependency for firebase-ads to your app-level build.gradle (Module: app) file: (where x is the latest version) compile 'com.google.firebase:firebase-ads:x.x.x'
  3. Add an AdView to the layout for the activity that will display the ad.
  4. Initialize AdMob ads at app launch, by calling MobileAds.initialize() in the onCreate() method of your main activity.
  5. Update the onCreate() of that activity to load the ad into the AdView.

☞ **Get ready to run test ads**

- While you are developing and testing your app, you can display and test ads to make sure your app is setup correctly todisplay ads. When testing ads you need:
- Your device ID (IMEI) for running test ads. To get the device ID either:
  - Go to Settings >About phone >status> IMEI.
  - Dial *#06#.
  - Your AdMob app ID. Get this in the AdMob console.
  - An ad unit ID. Get this in the AdMob console.

☞ **Add the AdView to display the ad**

- In the activity's layout file where you want the ad to appear, add an AdView :

```
<com.google.android.gms.ads.AdView
android:id="@+id/adView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginLeft="16dp"
android:layout_centerHorizontal="true"
```

```
ads:adSize="BANNER"
ads:adUnitId="@string/banner_ad_unit_id">
</com.google.android.gms.ads.AdView>
```

- You also need to add the ads namespace to the root view of the layout:

```
<RootLayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:ads="http://schemas.android.com/apk/res-auto"...>
```

- Here's an example of a layout with an AdView in the Layout Editor Fig. 6.10.2.

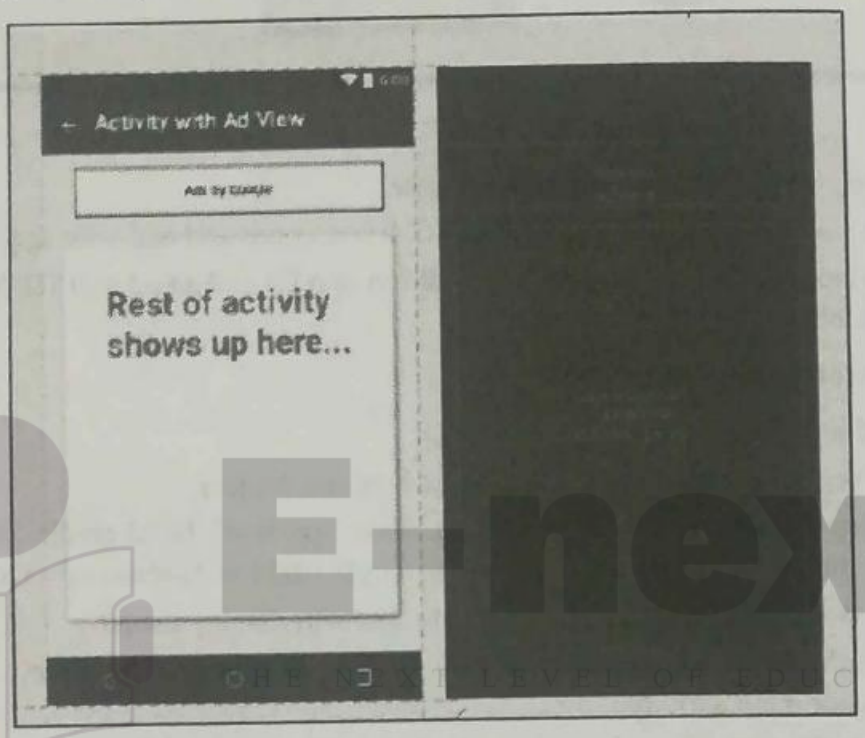


Fig. 6.10.2

☞ **Initialize MobileAds**

- The MobileAds class provides methods for initializing AdMob ads in your app. Call MobileAds.initialize() in theonCreate() method of your main activity, passing the context and your app ID (which you get from the AdMob console).

```
// Initialize AdMob
MobileAds.initialize(this, "ca-app-pub-1234");
```

☞ **Write the code to load the ad**

- In the onCreate() method of the activity that displays the ad, write the code to load the ad.
- While you are developing and testing your app you can display ads in test mode by specifying specific test devices. Toshow ads on your own device, you need to device ID (IMEI), which you can get from **Settings >About phone > Status>IMEI** or by dialling***#06#**.
- To load an ad:
  1. Get the AdView where the ad will appear.
  2. Create an AdRequest to request the ad.
  3. Call loadAd() on the AdView to load the ad into the AdView .

- Here's the code to write in onCreate() in the activity:

```
// Get the AdView
AdViewmAdView = (AdView) findViewById(R.id.adView);
// Create an AdRequest
AdRequestadRequest = new AdRequest.Builder()
// allow emulators to show ads
.addTestDevice(AdRequest.DEVICE_ID_EMULATOR)
// allow your device to show ads
.addTestDevice("1234") // your device id
.build();
// Load the ad into the AdView
mAdView.loadAd(adRequest);
```

Syllabus Topic : Publish Your App

6.11 Publish Your App

- The tasks for publishing the app to the Google Play store are:
  1. Prepare the app for release.
  2. Generate the signed APK.
  3. Upload the APK to the Google Play developer console.
  4. Run alpha and beta tests.
  5. Publish to the world!

6.11.1 APK

- An APK is a zip file that contains everything that the app needs to run on a user's device. It always has the **.apk** extension.
- You need an APK to publish your app in the Google Play store.
- You can use Android Studio to create the APK for your app. Before you generate the APK for your app, you need to doeverything you can to make your app successful, including:
  - Test your app thoroughly.
  - Make sure your app has the correct filters.
  - Add an icon.
  - Choose an Application ID.
  - Specify API levels targets.
  - Clean up your app.
- When your app is completely ready, then you can generate a signed APK to upload to the Google Play store.

6.11.2 Test Your App Thoroughly

- Make sure you run formal tests on your app, including unit and Espresso tests. These tests should cover both the corefeatures of your app, and the main integration points where your app calls out to

another API or retrieves data from theweb. These are points that are critical to your app, and they are the areas of code likely to break.

- Use Firebase Test Lab to run your app on a range of real devices in Google's data centers. This way you can verify boththe functionality and the compatibility of your app across many different kinds and versions of devices before to releasingyour app to a broader audience.

☞ **Make sure your app has the correct filters.**

- Make sure your app specifies the appropriate requirements to ensure that it reaches the right audience. For example, ifyour app requires biometric hardware for reading fingerprints, then add the requirement in the Android manifest.

```
<uses-featureandroid:name="android.hardware.fingerprint"/>
```

- However, specifying that your app needs a fingerprint reader limits the audience for your app to people who have deviceswith a fingerprint reader. You should think carefully before adding restrictions to the manifest that might limit who can seeand download your app.

☞ **Add a launcher icon to your app**

- A launcher icon is a graphic that represents your application. The launcher icon for your app appears in the Google Playstore listing. When users search the Google Play store, the icon for your app appears in the search results.
- When a user has installed the app, the launcher icon appears on the device in various places including:
  - On the home screen
  - In **Manage Applications**
  - In **My Downloads**

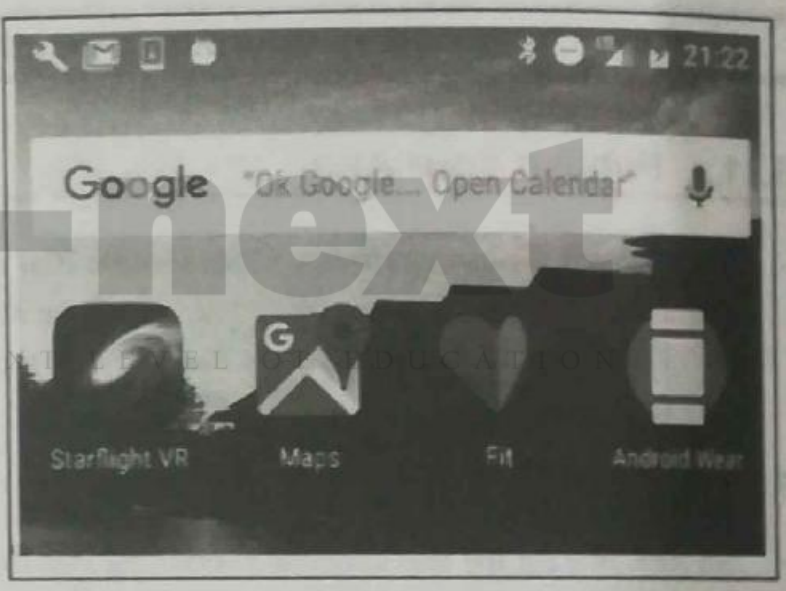


Fig. 6.11.1

☞ **Add an Application ID**

- The Application ID uniquely identifies an application. Make sure your app has an Application ID that will always be uniquefrom all other applications that a user might install on their device.When you create a project for an Android application, Android Studio automatically gives your project an Application ID.
- The value is initially the same as the package for the app. The Application ID is defined in the build.gradle file. For example :

```
defaultConfig {
applicationId "com.example.android.materialme"
minSdkVersion 15
targetSdkVersion 24
versionCode 1
versionName "1.0"
}
```

You can change your app's Application ID. It does not have to be the same as your app's package name.

When you are getting ready to publish your app, review the Application ID. The Application ID defines your application'sidentity. If you change it, then the app becomes a different application and users of the previous app will not be able toupdate to the new app.

Specify API Level targets and version number

When you create a project for an Android app in Android Studio, you select the minimum and target API levels for your app.

- **minSdkVersion** : minimum version of the Android platform on which the app will run.
- **targetSdkVersion** : API level on which the app is designed to run.

You can set these values in the Android manifest file, and also in the app-level build.gradle file.

Note: The value in build.gradle**overrides** the value in the manifest file. To prevent confusion, it is recommended that you put thevalues in build.gradle, and remove them from the manifest file. Setting these attributes in build.gradle also allows you tospecify different values for different versions of your app.

When you get your app ready for release, review API level targets and version number values and make sure they are

correct. People will not be able to find your app in the Google Play store if they are using devices whose SdkVersion is

below the value specified in your app.

Here's an example of setting these attribute values in build.gradle:

```
android {
...
defaultConfig {
...
minSdkVersion 14
targetSdkVersion 24
}
```

Note : The values of minSdkVersion and targetSdkVersion are the level of the API, not the version number of the AndroidOS.

Code name	Version number	Initial release date	API level
Honeycomb 	3.0 – 3.2.6	22 February 2011	11–13

- You can change your app's Application ID. It does not have to be the same as your app's package name.
- When you are getting ready to publish your app, review the Application ID. The Application ID defines your application'sidentity. If you change it, then the app becomes a different application and users of the previous app will not be able toupdate to the new app.

Specify API Level targets and version number

- When you create a project for an Android app in Android Studio, you select the minimum and target API levels for your app.
  - **minSdkVersion** : minimum version of the Android platform on which the app will run.
  - **targetSdkVersion** : API level on which the app is designed to run.
- You can set these values in the Android manifest file, and also in the app-level build.gradle file.

Note: The value in build.gradle**overrides** the value in the manifest file. To prevent confusion, it is recommended that you put thevalues in build.gradle, and remove them from the manifest file. Setting these attributes in build.gradle also allows you tospecify different values for different versions of your app.

- When you get your app ready for release, review API level targets and version number values and make sure they are
- correct. People will not be able to find your app in the Google Play store if they are using devices whose SdkVersion is
- below the value specified in your app.
- Here's an example of setting these attribute values in build.gradle:

```
android {
...
defaultConfig {
...
minSdkVersion 14
targetSdkVersion 24
}
```

Note : The values of minSdkVersion and targetSdkVersion are the level of the API, not the version number of the AndroidOS.

Code name	Version number	Initial release date	API level
Honeycomb 	3.0 – 3.2.6	22 February 2011	11–13

Code name	Version number	Initial release date	API level
Ice Cream Sandwich 	4.0 – 4.0.4	18 October 2011	14–15
Jelly Bean 	4.1 – 4.3.1	9 July 2012	16–18
KitKat 	4.4 – 4.4.4	31 October 2013	19–20
Lollipop 	5.0 – 5.1.1	12 November 2014	21–22
Marshmallow 	6.0 – 6.0.1	5 October 2015	23
Nougat 	7.0	22 August 2016	24

Code name	Version number	Initial release date	API level
Oreo 	8.0	August 21, 2017	26

Version number

- You need to specify a version number for your app. As you improve your app to add new features, you will need to updatethe version number each time you release a new version to the Google Play store. Read more in the Android Versionguide.

Product Flavors

- You can generate different "product flavors" for your app. A product flavor is a customized version of the application build.
- For example, you could have a demo version and a production version. Here's an example of how to define product flavorsin build.gradle:

```
android {
...
productFlavors {
demo {
applicationId "com.example.myapp.demo"
versionName "1.0-demo"
}
full {
applicationId "com.example.myapp.full"
versionName "1.0-full"
}
}
}
```

Reduce your app's size

- The size of your APK affects:
  - How fast your app loads
  - How much memory it uses
  - How much power it consumes
- The bigger the size of your app's APK, the more likely it is that some users will not download it because of size limitationson their device or connectivity limitations.
- Users who have pay-by-the-byte plans will be particularly concerned about howlong an app takes to download. If your app takes up too much space, users will be more likely to uninstall it when they needspace for other apps or files.
- The following steps may be taken to reduce the size of your app:

- Clean up your project to remove unused resources
- Re-use resources
- Minimize resource use from libraries
- Reduce native and Java code
- Reduce space needs for images

6.11.3 Publish Your App!

Q. How to publish app?

- When you've tested your app, cleaned it up, reduced its size, and generated the APK, you are ready to publish it to GooglePlay.
- After you upload your app to Google Play, you can run alpha and beta tests before releasing it to the public. Running alphaand beta tests lets you share your app with real users, and get feedback from them. This feedback does not appear asreviews in Google Play.
- Run alpha tests while you are developing your app. Use alpha tests for early experimental versions of your app that mightcontain incomplete or unstable functionality. Running alpha tests is also a good way to share your app with friends and family.
- Run beta tests with limited number of real users, to do final testing before your app goes public.
- Once your app is public, users can give reviews. So, make sure you test it thoroughly before putting it out on Google Playfor anyone to download.

☞ **Create an account in the Google Play Developer Console**

- Whether you want to run alpha and beta tests, or publish your app to the public on Google Play, you need to upload yourAPK in the Google Play developer console.
- Go to the console at play.google.com/apps/publish/.
- To begin, get a Google Play developer account. You will need to pay for the account. The high-level steps are:
  1. Go to play.google.com/apps/publish/
  2. Accept the agreement.
  3. Pay the registration fee.
  4. Enter your details, such as your name, address, website, phone and email preferences.
- When you have set up your account, you can upload your APK. In the Google Play Developer console interface, choose:
  1. Production 2. Beta Testing 3. Alpha Testing
- Then you can browse for the APK to upload, or drag and drop it to the console.
- You need to satisfy the following requirements before you can publish your app to the public:
  - add a high-res icon
  - add a feature graphic (in case your app is selected as a Featured App in Google Play)
  - add 2 non-Android TV screenshots
  - select a category
  - select a content rating
  - target at least one country
  - enter a privacy policy URL

- make your app free or set a price for it
- declare if your app contains ads
- add a required content rating

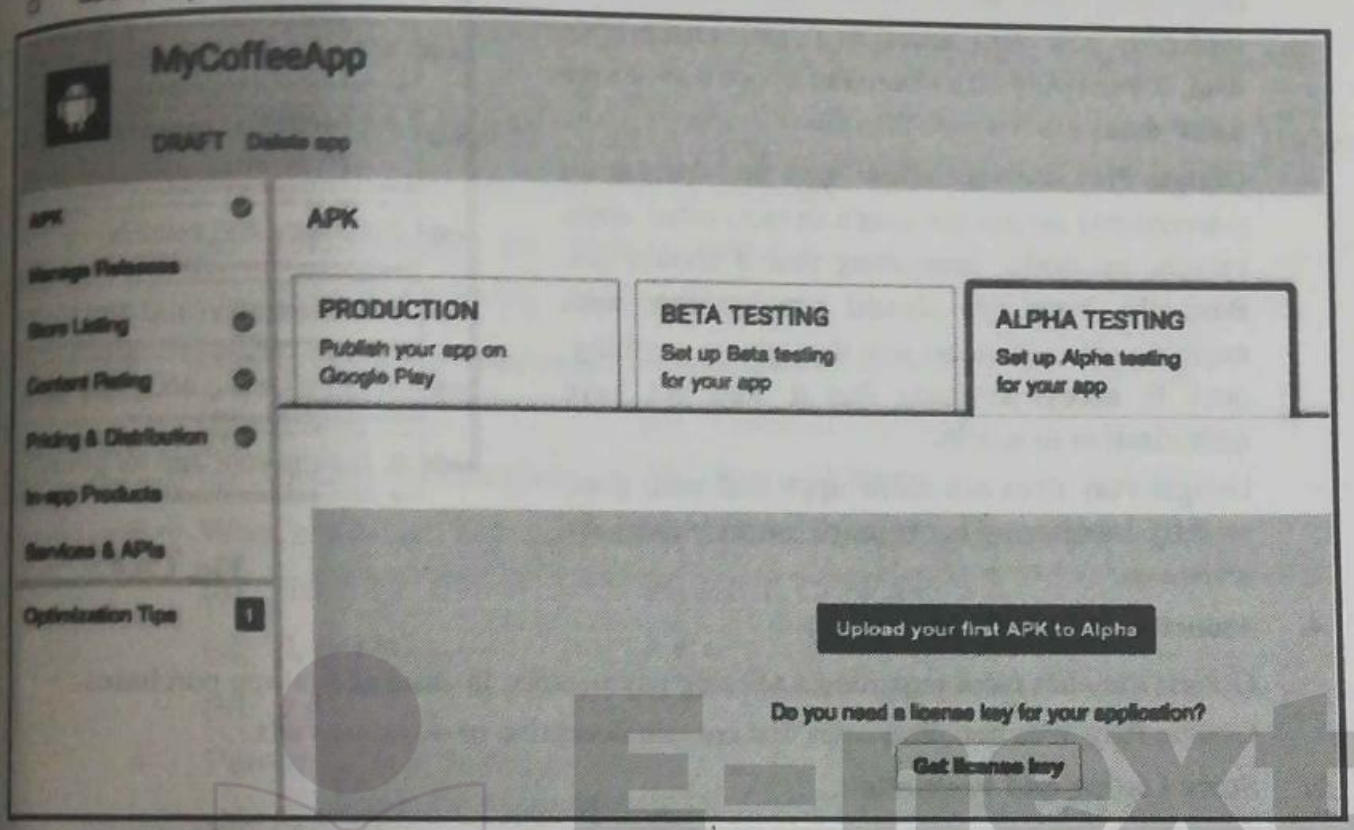


Fig. 6.11.2

Run pre-launch reports

After you upload your APK, you can run pre-launch reports to identify crashes, display issues and security issues. Duringthe pre-launch check, test devices automatically launch and crawl your app for several minutes.

The crawl performs basic actions every few seconds on your app, such as typing, tapping, and swiping. The pre-launchtests use Firebase Cloud Test Lab.

Review criteria for publishing

Your app must comply with Google Play policies, which ensure that all apps on Google Play provide a safe experience for everyone.

6.11.4 Policies Governing

1. Restricted content

Google Play does not allow apps that are sexually-explicit, hateful, racist, encourage bullying or violence, or facilitategambling.

2. Intellectual property, deception and spam

Google Play does not allow apps that are not honest. In other words, they pretend to be other apps or pretend to comefrom other companies or impersonate other brands. Google Play does not allow apps that attempt to deceive users. GooglePlay does not allow apps that spam users such as apps that send users unsolicited messages. Google Play does not allowapps that are duplicative or of low-quality.

3. Privacy and security

- Google Play requires that your app treats users' data safely and keeps private user information secret. If your app accessesor transmits private data, it must publish a statement about how it uses users' data.
- Google Play does not allow apps that damage or subversively access the user's device, other apps, servers, networks, oranything that it should not. Basically, your app should not interfere with anything else, or cause any damage to anything, ortry to access anything that it does not have authorization to access.
- Google Play does not allow apps that steal data, secretly monitor or harm users, or are otherwise malicious.

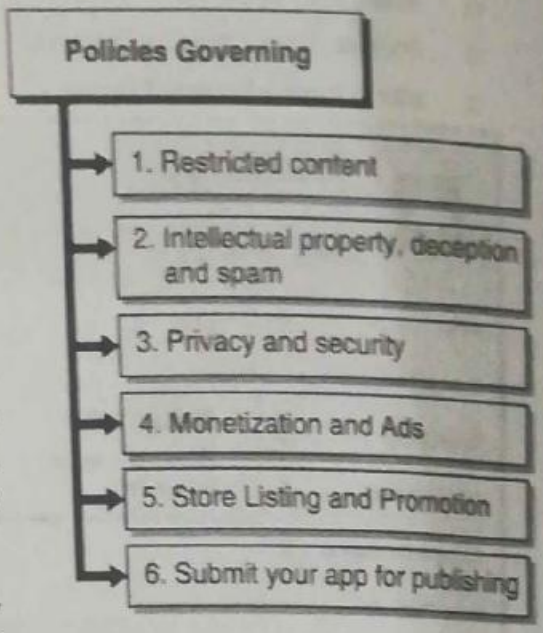


Fig. C6.4

4. Monetization and Ads

- Google Play has rules regarding accepting payment for in-store and in-app purchases.
- Google Play does not allow apps that contain deceptive or disruptive ads.

5. Store Listing and Promotion

- Publishers must not attempt to promote their own apps unfairly. For example, you are not allowed to get 100,000 of yourclosest friends to give your app a 5 star rating so that it appears with very favorable reviews. Your app's icon, title,description and screenshot must all fairly represent your app, and not make any exaggerated or misleading claims.
- In other words, don't cheat to get a better Google Play rating or placement.

6. Submit your app for publishing

- When you upload your app for production, Google checks your app. Google runs both automatic and manual checking onyour app.
- If your app is rejected, fix the problem and try again!

Review Questions

Q. 1 Write short note shared preferences. **(Refer section 6.1.1)**

Q. 2 Explain in internal storage. **(Refer section 6.1.2)**

Q. 3 Describe external storage in detail. **(Refer section 6.1.3)**

Q. 4 Write short note on ACID. **(Refer section 6.2.1)**

Q. 5 How to database handle in Android? **(Refer section 6.3.1)**

Q. 6 Explain content provider in detail.**(Refer section 6.4)**

Q. 7 How to create content provider? **(Refer section 6.4.2)**

❑❑❑