Unit – 7
Advanced Android Concepts [10 Hrs]

Local Database with SQLite

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 very lightweight database that comes with Android OS. It supports all the relational database features. In order to access this database, you don't need to establish any kind of connections for it like JDBC, ODBC etc.

The version of SQLite depends on the version of Android. See the following table:

Android API	SQLite Version
API 27	3.19
API 26	3.18
API 24	3.9
API 21	3.8
API 11	3.7
API 8	3.6
API 3	3.5
API 1	3.4

Features of SQLite:

Following is a list of features which makes SQLite popular among other lightweight databases:

- **SQLite is totally free:** SQLite is open-source. So, no license is required to work with it.
- **SQLite is server less:** SQLite doesn't require a different server process or system to operate.
- **SQLite is very flexible:** It facilitates you to work on multiple databases on the same session on the same time.
- **Configuration Not Required:** SQLite doesn't require configuration. No setup or administration required.
- **SQLite is a cross-platform DBMS:** You don't need a large range of different platforms like Windows, Mac OS, Linux, and Unix. It can also be used on a lot of embedded operating systems like Symbian, and Windows CE.
- **Storing data is easy:** SQLite provides an efficient way to store data.
- **Variable length of columns:** The length of the columns is variable and is not fixed. It facilitates you to allocate only the space a field needs. For example, if you have a varchar (200) column, and you put a 10 characters' length value on it, then SQLite will allocate only 20 characters' space for that value not the whole 200 space.
- **Provide large number of API's:** SQLite provides API for a large range of programming languages. **For example:** .Net languages (Visual Basic, C#), PHP, Java, Objective C, Python and a lot of other programming language.

- **SQLite** is written in **ANSI-C** and provides simple and easy-to-use API.
- **SQLite** is available on UNIX (Linux, Mac OS-X, Android, iOS) and Windows (Win32, WinCE, WinRT).

SQL vs SQLite

S.N.	SQL	SQLite
1	SQL is a Structured Query Language used to query a Relational Database System. It is written in C language.	SQLite is an Embeddable Relational Database Management System which is written in ANSI-C.
2	SQL is a standard which specifies how a relational schema is created, data is inserted or updated in the relations, transactions are started and stopped, etc.	SQLite is file-based. It is different from other SQL databases because unlike most other SQL databases, SQLite does not have a separate server process.
3	Main components of SQL are Data Definition Language(DDL) , Data Manipulation Language(DML), Embedded SQL and Dynamic SQL.	SQLite supports many features of SQL and has high performance and does not support stored procedures.
4	SQL is Structured Query Language which is used with databases like MySQL, Oracle, Microsoft SQL Server, IBM DB2, etc. It is not a database itself.	SQLite is a portable database resource. You have to get an extension of SQLite in whatever language you are programming in to access that database. You can access all of the desktop and mobile applications.
5	A conventional SQL database needs to be running as a service like Oracle DB to connect to and provide a lot of functionalities.	SQLite database system doesn't provide such functionalities.
6	SQL is a query language which is used by different SQL databases. It is not a database itself.	SQLite is a database management system itself which uses SQL.

Advantages and Disadvantages of SQLite

Following are the **advantages** of using SQLite:

1. **Lightweight**
   SQLite is a very light weighted database so, it is easy to use it as an embedded software with devices like televisions, Mobile phones, cameras, home electronic devices, etc.
2. **Better Performance**
   Reading and writing operations are very fast for SQLite database. It is almost 35% faster than File system. It only loads the data which is needed, rather than reading the entire file and hold it in memory. If you edit small parts, it only overwrites the parts of the file which was changed.

3. **No Installation Needed**
   SQLite is very easy to learn. You don't need to install and configure it. Just download SQLite libraries in your computer and it is ready for creating the database.
4. **Reliable**
   It updates your content continuously so, little or no work is lost in a case of power failure or crash. SQLite is less bugs prone rather than custom written file I/O codes. SQLite queries are smaller than equivalent procedural codes so, chances of bugs are minimal.
5. **Portable**
   SQLite is portable across all 32-bit and 64-bit operating systems and big- and little-endian architectures. Multiple processes can be attached with same application file and can read and write without interfering each other. It can be used with all programming languages without any compatibility issue.
6. **Accessible**
   SQLite database is accessible through a wide variety of third-party tools. SQLite database's content is more likely to be recoverable if it has been lost. Data lives longer than code.
7. **Reduce Cost and Complexity**
   It reduces application cost because content can be accessed and updated using concise SQL queries instead of lengthy and error-prone procedural queries. SQLite can be easily extended in in future releases just by adding new tables and/or columns. It also preserves the backwards compatibility.

Following are the **disadvantages** of using SQLite:

1. SQLite is used to handle low to medium traffic HTTP requests.
2. Database size is restricted to 2GB in most cases.

Establishing Connection

To use SQLite database in android two classes are used for creating database and manipulating data. They are **SQLiteOpenHelper** and **SQLiteDatabase** class**.**

The **SQLiteOpenHelper** class is used for database creation and version management. For performing any database operation, you have to provide the implementation of **onCreate()** and **onUpgrade()** methods of SQLiteOpenHelper class.

There are many methods in SQLiteOpenHelper class. Some of them are as follows:

Method	Description
public abstract void onCreate(SQLiteDatabase db)	called only once when database is created for the first time.
public abstract void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)	called when database needs to be upgraded.
public synchronized void close ()	closes the database object.
public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion)	called when database needs to be downgraded.

SQLiteDatabase class contains methods to be performed on SQLite database such as create, update, delete, select etc. There are many methods in SQLiteDatabase class. Some of them are as follows:

Method	Description
void execSQL(String sql)	executes the sql query not select query. Basically used for creating database and tables.
long insert(String table, String nullColumnHack, ContentValues values)	inserts a record on the database. The table specifies the table name, nullColumnHack doesn't allow completely null values. If second argument is null, android will store null values if values are empty. The third argument specifies the values to be stored.
int update(String table, ContentValues values, String whereClause, String[] whereArgs)	updates a row.
Cursor rawQuery(String query, String[] selectionArgs)	returns a cursor over the result set.

Creating Database and Tables

Following code snippet will create SQLite Database and Table in android:

```
public class MyDbHelper extends SQLiteOpenHelper{
    // Database Version
    private static final int DATABASE_VERSION = 1;
    // Database Name
    private static final String DATABASE_NAME = "mydb";

    public MyDbHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    // Creating Tables
    @Override
    public void onCreate(SQLiteDatabase db) {
        // create notes table
        String createQuery="CREATE TABLE mytable(id INTEGER PRIMARY
                        KEY,name TEXT,address TEXT)";
        db.execSQL(createQuery);
    }

    // Upgrading database
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
                newVersion) {
        // Drop older table if existed
        db.execSQL("DROP TABLE IF EXISTS " + DATABASE_NAME);
        // Create tables again
        onCreate(db);
    }
}
```

Data Manipulation

Following code snippet can be used to perform various manipulation operation like insert, update, select and delete.

```
//code to insert data
public void insertData(int id,String name, String address){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues=new ContentValues();
    contentValues.put("id",id);
    contentValues.put("name",name);
    contentValues.put("address",address);
    //inserting row
    db.insert("mytable",null,contentValues);
    db.close();
}

//code to select data
public Cursor selectData(){
     SQLiteDatabase db=this.getReadableDatabase();
     String query="SELECT * FROM mytable";
     Cursor cursor=db.rawQuery(query,null);
     return cursor;
}

//code to update data
public void updateData(String id,String name, String address){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues=new ContentValues();
    contentValues.put("name",name);
    contentValues.put("address",address);
    //updating row
    db.update("mytable",contentValues,"id=?",new String[]{id});
    db.close();
}

//code to delete data
public void deleteData(String id) {
    SQLiteDatabase db = this.getWritableDatabase();
    //deleting row
    db.delete("mytable","id=?",new String[]{id});
}
```

Following example will demonstrate the use of SQLite in Android. In this example, we are creating three columns id, name and address in database table. Also we are creating effective UI for showing different data manipulation operations.

So, let's start by designing UI.

sqlite_example.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:orientation="vertical"
    android:layout_margin="5dp"
    android:layout_height="match_parent">
```

```
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Id"
        android:inputType="number"
        android:id="@+id/edtId" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Name"
        android:layout_below="@+id/edtId"
        android:id="@+id/edtName" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Address"
        android:layout_below="@+id/edtName"
        android:id="@+id/edtAddress" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edtAddress"
        android:id="@+id/btnInsert"
        android:text="Insert" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edtAddress"
        android:layout_toRightOf="@+id/btnInsert"
        android:id="@+id/btnSelect"
        android:layout_marginLeft="10dp"
        android:text="Select" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edtAddress"
        android:layout_toRightOf="@+id/btnSelect"
        android:id="@+id/btnUpdate"
        android:layout_marginLeft="10dp"
        android:text="Update" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edtAddress"
        android:layout_toRightOf="@+id/btnUpdate"
        android:id="@+id/btnDelete"
        android:layout_marginLeft="10dp"
        android:text="Delete" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Selected Data:"
        android:layout_below="@+id/btnSelect"
```

```
        android:layout_marginTop="10dp"
        android:id="@+id/txtData"
        android:textSize="18sp" />

</RelativeLayout>
```

Now we are creating Helper class for SQLite operations by extending SQLiteOpenHelpter as follows:

```
public class MyDbHelper extends SQLiteOpenHelper{
    // Database Version
    private static final int DATABASE_VERSION = 1;
    // Database Name
    private static final String DATABASE_NAME = "mydb";

    public MyDbHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    // Creating Tables
    @Override
    public void onCreate(SQLiteDatabase db) {
        // create notes table
        String createQuery="CREATE TABLE mytable(id INTEGER PRIMARY
                KEY,name TEXT, address TEXT)";
        db.execSQL(createQuery);
    }

    // Upgrading database
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
        newVersion) {
        // Drop older table if existed
        db.execSQL("DROP TABLE IF EXISTS " + DATABASE_NAME);
        // Create tables again
        onCreate(db);
    }

    //code to insert data
    public void insertData(int id,String name, String address){
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues=new ContentValues();
        contentValues.put("id",id);
        contentValues.put("name",name);
        contentValues.put("address",address);
        //inserting row
        db.insert("mytable",null,contentValues);
        db.close();
    }

    //code to select data
    public Cursor selectData(){
         SQLiteDatabase db=this.getReadableDatabase();
         String query="SELECT * FROM mytable";
         Cursor cursor=db.rawQuery(query,null);
         return cursor;
    }

    //code to update data
    public void updateData(String id,String name, String address){
```

```
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues=new ContentValues();
        contentValues.put("name",name);
        contentValues.put("address",address);
        //updating row
        db.update("mytable",contentValues,"id=?",new String[]{id});
        db.close();
    }

    //code to delete data
    public void deleteData(String id) {
        SQLiteDatabase db = this.getWritableDatabase();
        //deleting row
        db.delete("mytable","id=?",new String[]{id});
    }
}
```

Now finally we are creating Activity named as "**SqliteExample.java**" to perform various data manipulation operations:

```
public class SqliteExample extends AppCompatActivity {
    EditText edtId,edtName,edtAddress;
    Button btnInsert,btnSelect,btnUpdate,btnDelete;
    TextView txtData;
    MyDbHelper myDbHelper;
    @Override
    protected void onCreate(Bundle b){
        super.onCreate(b);
        setContentView(R.layout.sqlite_example);

        //creating object of MyDbHelper class
        myDbHelper=new MyDbHelper(this);

        edtId=findViewById(R.id.edtId);
        edtName=findViewById(R.id.edtName);
        edtAddress=findViewById(R.id.edtAddress);
        btnInsert=findViewById(R.id.btnInsert);
        btnUpdate=findViewById(R.id.btnUpdate);
        btnSelect=findViewById(R.id.btnSelect);
        btnDelete=findViewById(R.id.btnDelete);
        txtData=findViewById(R.id.txtData);

        btnInsert.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                int id=Integer.parseInt(edtId.getText().toString());
                String name=edtName.getText().toString();
                String address=edtAddress.getText().toString();
                //calling insert function
                myDbHelper.insertData(id,name,address);
                Toast.makeText(getApplicationContext(),"Data Inserted
                        Successfully !", Toast.LENGTH_SHORT).show();
            }
        });

        btnSelect.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //retrieving data
                int id=0;
```

```
                String name="",address="";
                //calling select function
                Cursor cursor=myDbHelper.selectData();
                while (cursor.moveToNext()){
                    id=cursor.getInt(0);
                    name=cursor.getString(1);
                    address=cursor.getString(2);
                }
                //displaying data in TextView
                txtData.setText("Id="+id+"\t Name="+name+"\tAddress="
                                                +address);
            }
        });

        btnUpdate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String id=edtId.getText().toString();
                String name=edtName.getText().toString();
                String address=edtAddress.getText().toString();
                //calling insert function
                myDbHelper.updateData(id,name,address);
                Toast.makeText(getApplicationContext(),"Data Updated
                        Successfully !",Toast.LENGTH_SHORT).show();
            }
        });

        btnDelete.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String id=edtId.getText().toString();
                //calling delete function
                myDbHelper.deleteData(id);
                Toast.makeText(getApplicationContext(),"Data Deleted
                        Successfully !",Toast.LENGTH_SHORT).show();
            }
        });

    }
}
```

Above code produces following output:

Figure 7-1. Output demonstrating various operation using SQLite

Displaying data in ListView

In above example, we have created TextView for displaying data. The problem with above TextView is that we are able to display only one row at a time. To overcome this problem, we can use widgets like ListView, GridView and RecyclerView to display multiple rows at a time.

Here, in this example we are using ListView widget to display our data.

list_items.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:id="@+id/relat"
    android:layout_height="wrap_content">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_margin="10dp"
        android:text="Id"
        android:textStyle="bold"
```

```
        android:id="@+id/txtId" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:id="@+id/txtName"
        android:textStyle="bold"
        android:layout_toRightOf="@+id/txtId"
        android:layout_marginTop="10dp"
        android:text="Name" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:id="@+id/txtAddress"
        android:text="Address"
        android:layout_marginLeft="10dp"
        android:layout_below="@+id/txtName" />

</RelativeLayout>
```

sqlite_example.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:orientation="vertical"
    android:layout_margin="5dp"
    android:layout_height="match_parent">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Id"
        android:inputType="number"
        android:id="@+id/edtId" />

   <EditText
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:hint="Enter Name"
       android:layout_below="@+id/edtId"
       android:id="@+id/edtName" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Address"
        android:layout_below="@+id/edtName"
        android:id="@+id/edtAddress" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edtAddress"
        android:id="@+id/btnInsert"
        android:text="Insert" />
```

```
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edtAddress"
        android:layout_toRightOf="@+id/btnInsert"
        android:id="@+id/btnSelect"
        android:layout_marginLeft="10dp"
        android:text="Select" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edtAddress"
        android:layout_toRightOf="@+id/btnSelect"
        android:id="@+id/btnUpdate"
        android:layout_marginLeft="10dp"
        android:text="Update" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edtAddress"
        android:layout_toRightOf="@+id/btnUpdate"
        android:id="@+id/btnDelete"
        android:layout_marginLeft="10dp"
        android:text="Delete" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Selected Data:"
        android:layout_below="@+id/btnSelect"
        android:layout_marginTop="10dp"
        android:id="@+id/txtData"
        android:textSize="18sp" />

    <ListView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/mylist"
        android:layout_below="@+id/txtData" />

</RelativeLayout>
```

ListAdapter.java

```
public class ListAdapter extends ArrayAdapter<String> {
    Activity context;
    ArrayList<Integer> id;
    ArrayList<String> name;
    ArrayList<String> address;

    public ListAdapter(Activity context, ArrayList<Integer> id,
                ArrayList<String> name, ArrayList<String> address) {
        //ArrayAdapter needs String so we are supplying title
        super(context, R.layout.list_items,name);
        this.context=context;
        this.id=id;
        this.name=name;
```

```
        this.address=address;
    }

    public View getView(int position, View view, ViewGroup parent) {
        LayoutInflater inflater=context.getLayoutInflater();
        View rowView=inflater.inflate(R.layout.list_items, null,true);
        //wiring widgets
        TextView txtId = (TextView) rowView.findViewById(R.id.txtId);
        TextView txtName = (TextView) rowView.findViewById(R.id.txtName);
        TextView txtAddress = (TextView) rowView.findViewById
                              (R.id.txtAddress);

        txtId.setText(id.get(position).toString());
        txtName.setText(name.get(position).toString());
        txtAddress.setText(address.get(position).toString());

        return rowView;

    };
}
```

SqliteExample.java

```
public class SqliteExample extends AppCompatActivity {
    EditText edtId,edtName,edtAddress;
    Button btnInsert,btnSelect,btnUpdate,btnDelete;
    MyDbHelper myDbHelper;
    ListView listView;
    @Override
    protected void onCreate(Bundle b){
        super.onCreate(b);
        setContentView(R.layout.sqlite_example);

        //creating object of MyDbHelper class
        myDbHelper=new MyDbHelper(this);

        edtId=findViewById(R.id.edtId);
        edtName=findViewById(R.id.edtName);
        edtAddress=findViewById(R.id.edtAddress);
        btnInsert=findViewById(R.id.btnInsert);
        btnUpdate=findViewById(R.id.btnUpdate);
        btnSelect=findViewById(R.id.btnSelect);
        btnDelete=findViewById(R.id.btnDelete);
        listView=findViewById(R.id.mylist);

        btnInsert.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                int id=Integer.parseInt(edtId.getText().toString());
                String name=edtName.getText().toString();
                String address=edtAddress.getText().toString();
                //calling insert function
                myDbHelper.insertData(id,name,address);
                Toast.makeText(getApplicationContext(),"Data Inserted
                        Successfully !",Toast.LENGTH_SHORT).show();
            }
        });

        btnSelect.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View view) {
                //creating dynamic array using Arraylist
                //array is not suitable for this example
                ArrayList<Integer> id=new ArrayList<>();
                ArrayList<String> name=new ArrayList<>();
                ArrayList<String> address=new ArrayList<>();
                //calling select function
                Cursor cursor=myDbHelper.selectData();
                while (cursor.moveToNext()){
                    id.add(cursor.getInt(0));
                    name.add(cursor.getString(1));
                    address.add(cursor.getString(2));
                }
                ListAdapter adapter=new ListAdapter
                                (SqliteExample.this,id,name,address);
                listView.setAdapter(adapter);
            }
        });

        btnUpdate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String id=edtId.getText().toString();
                String name=edtName.getText().toString();
                String address=edtAddress.getText().toString();
                //calling insert function
                myDbHelper.updateData(id,name,address);
                Toast.makeText(getApplicationContext(),"Data Updated
                        Successfully !",Toast.LENGTH_SHORT).show();
            }
        });

        btnDelete.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String id=edtId.getText().toString();
                //calling delete function
                myDbHelper.deleteData(id);
                Toast.makeText(getApplicationContext(),"Data Deleted
                        Successfully !",Toast.LENGTH_SHORT).show();
            }
        });

    }
}
```

Above code produces following output:

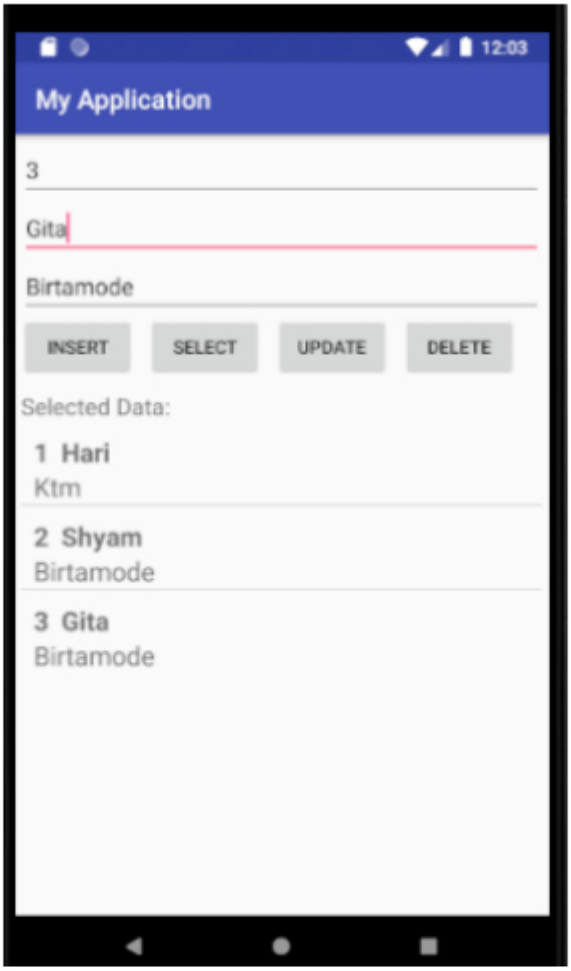


Figure 7-2. Output demonstrating displaying retrieved data in ListView

Displaying data in RecyclerView

Like in ListView we can also plot retrieved data in RecyclerView as follows:

sqlite_example.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:orientation="vertical"
    android:layout_margin="5dp"
    android:layout_height="match_parent">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Id"
        android:inputType="number"
        android:id="@+id/edtId" />

   <EditText
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:hint="Enter Name"
```

```
        android:layout_below="@+id/edtId"
        android:id="@+id/edtName" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Address"
        android:layout_below="@+id/edtName"
        android:id="@+id/edtAddress" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edtAddress"
        android:id="@+id/btnInsert"
        android:text="Insert" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edtAddress"
        android:layout_toRightOf="@+id/btnInsert"
        android:id="@+id/btnSelect"
        android:layout_marginLeft="10dp"
        android:text="Select" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edtAddress"
        android:layout_toRightOf="@+id/btnSelect"
        android:id="@+id/btnUpdate"
        android:layout_marginLeft="10dp"
        android:text="Update" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edtAddress"
        android:layout_toRightOf="@+id/btnUpdate"
        android:id="@+id/btnDelete"
        android:layout_marginLeft="10dp"
        android:text="Delete" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Selected Data:"
        android:layout_below="@+id/btnSelect"
        android:layout_marginTop="10dp"
        android:id="@+id/txtData"
        android:textSize="18sp" />

    <android.support.v7.widget.RecyclerView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/recyclerview"
        android:layout_below="@+id/txtData" />

</RelativeLayout>
```

DataModel.java

```
public class DataModel {
    private int id;
    private String name;
    private String address;

    public DataModel(int id, String name, String address){
        this.id=id;
        this.name=name;
        this.address=address;
    }

    public int getId(){
        return id;
    }

    public String getName(){
        return name;
    }

    public String getAddress(){
        return address;
    }
}
```

RecyclerViewAdapter.java

```
public class RecyclerViewAdapter extends
RecyclerView.Adapter<RecyclerViewAdapter.ViewHolder> {
    Activity context;
    ArrayList<DataModel> data;

    public RecyclerViewAdapter(Activity context, ArrayList<DataModel>
            data){
        this.context=context;
        this.data=data;
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType)
{
        LayoutInflater layoutInflater = LayoutInflater.from(context);
        View listItem= layoutInflater.inflate(R.layout.list_items,
                parent, false);
        ViewHolder viewHolder = new ViewHolder(listItem);
        return viewHolder;
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        final DataModel current=data.get(position);
        holder.txtId.setText(current.getId()+"");
        holder.txtName.setText(current.getName());
        holder.txtAddress.setText(current.getAddress());
    }
```

```
    @Override
    public int getItemCount() {
        return data.size();
    }

    public static class ViewHolder extends RecyclerView.ViewHolder {
        TextView txtId,txtName,txtAddress;
        ImageView imageView;
        public ViewHolder(View itemView) {
            super(itemView);
            txtId = itemView.findViewById(R.id.txtId);
            txtName = itemView.findViewById(R.id.txtName);
            txtAddress = itemView.findViewById(R.id.txtAddress);
        }
    }
}
```

SqliteExample.java

```
public class SqliteExample extends AppCompatActivity {
    EditText edtId,edtName,edtAddress;
    Button btnInsert,btnSelect,btnUpdate,btnDelete;
    MyDbHelper myDbHelper;
    RecyclerView recyclerView;
    RecyclerView.Adapter recyclerViewAdapter;
    RecyclerView.LayoutManager layoutManager;
    @Override
    protected void onCreate(Bundle b){
        super.onCreate(b);
        setContentView(R.layout.sqlite_example);

        //creating object of MyDbHelper class
        myDbHelper=new MyDbHelper(this);

        edtId=findViewById(R.id.edtId);
        edtName=findViewById(R.id.edtName);
        edtAddress=findViewById(R.id.edtAddress);
        btnInsert=findViewById(R.id.btnInsert);
        btnUpdate=findViewById(R.id.btnUpdate);
        btnSelect=findViewById(R.id.btnSelect);
        btnDelete=findViewById(R.id.btnDelete);
        recyclerView=findViewById(R.id.recyclerview);

        btnInsert.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                int id=Integer.parseInt(edtId.getText().toString());
                String name=edtName.getText().toString();
                String address=edtAddress.getText().toString();
                //calling insert function
                myDbHelper.insertData(id,name,address);
                Toast.makeText(getApplicationContext(),"Data Inserted
                        Successfully !",Toast.LENGTH_SHORT).show();
            }
        });

        btnSelect.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
```

```
                ArrayList<DataModel> data=new ArrayList<>();
                //calling select function
                Cursor cursor=myDbHelper.selectData();
                while (cursor.moveToNext()){
                   int id=cursor.getInt(0);
                   String name=cursor.getString(1);
                   String address=cursor.getString(2);
                   DataModel dataModel=new DataModel(id,name,address);
                   data.add(dataModel);
                }
                //plotting in recyclerview
                layoutManager=new LinearLayoutManager
                                        (SqliteExample.this);
                recyclerView.setLayoutManager(layoutManager);
                recyclerViewAdapter=new RecyclerViewAdapter
                                (SqliteExample.this,data);
                recyclerView.setAdapter(recyclerViewAdapter);
            }
        });

        btnUpdate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String id=edtId.getText().toString();
                String name=edtName.getText().toString();
                String address=edtAddress.getText().toString();
                //calling insert function
                myDbHelper.updateData(id,name,address);
                Toast.makeText(getApplicationContext(),"Data Updated
               Successfully !",Toast.LENGTH_SHORT).show();
            }
        });

        btnDelete.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String id=edtId.getText().toString();
                //calling delete function
                myDbHelper.deleteData(id);
                Toast.makeText(getApplicationContext(),"Data Deleted
                       Successfully !",Toast.LENGTH_SHORT).show();
            }
        });

    }
}
```

Above code produces following output:

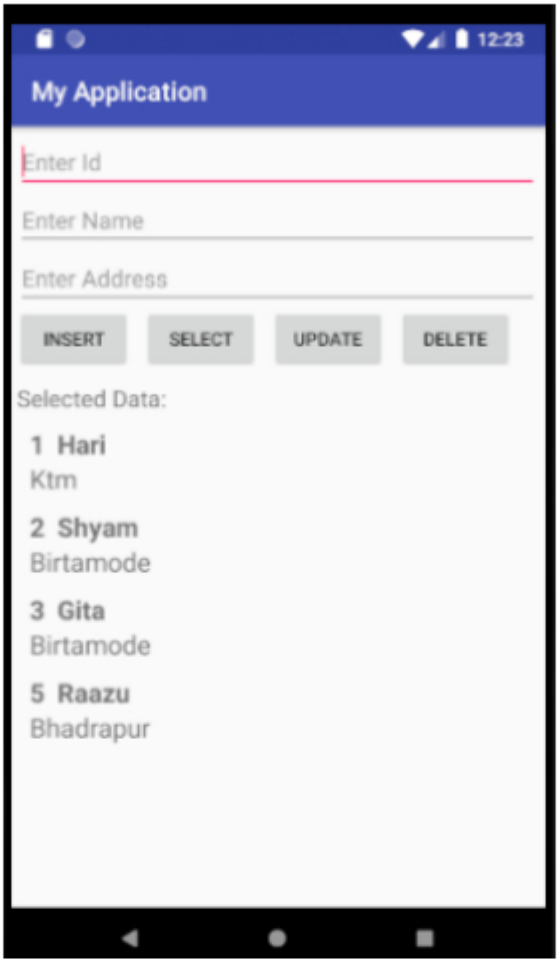


Figure 7-3. Output demonstrating displaying retrieved data in RecyclerView

Introduction to API

API is an abbreviation for Application Programming Interface which is a collection of communication protocols and subroutines used by various programs to communicate between them. A programmer can make use of various API tools to make its program easier and simpler. Also, an API facilitates the programmers with an efficient way to develop their software programs.

Thus in simpler terms, an API helps two programs or applications to communicate with each other by providing them with necessary tools and functions. It takes the request from the user and sends it to the service provider and then again sends the result generated from the service provider to the desired user.

A developer extensively uses API's in his software to implement various features by using an API call without writing the complex codes for the same. We can create an API for an operating system, database systems, hardware system, for a JavaScript file or similar object oriented files. Also, an API is similar to a GUI (Graphical User Interface) with one major difference. Unlike GUI's, an API helps the software developers to access the web tools while a GUI helps to make a program easier to understand by the users.

Real life example of an API:

Suppose, we are searching for a hotel room on an online website. In this case, you have a vast number of options to choose from and this may include the hotel location, the check-in and check-out dates, price, accommodation details and many more factors. So in order to book the room online, you need to interact with the hotel booking's website which in further will let you know if there is a room available on that particular date or not and at what price.

Now in the above example, the API is the interface that actually communicates in between. It takes the request of the user to the hotel booking's website and in turn returns back the most relevant data from the website to the intended user. Thus, we can see from this example how an API works and it has numerous applications in real life from switching on mobile phones to maintaining a large amount of databases from any corner of the world.

Types of API

There are various kinds of API's available according to their uses and applications like the Browser API which is created for the web browsers to abstract and to return the data from surroundings or the Third party API's, for which we have to get the codes from other sites on the web (e.g. Facebook, Twitter).

There are three basic forms of API-

1. **Web APIs:**
   A Web API also called as Web Services is an extensively used API over the web and can be easily accessed using the HTTP protocols. A Web API is an open source interface and can be used by a large number of clients through their phones, tablets. or PC's.

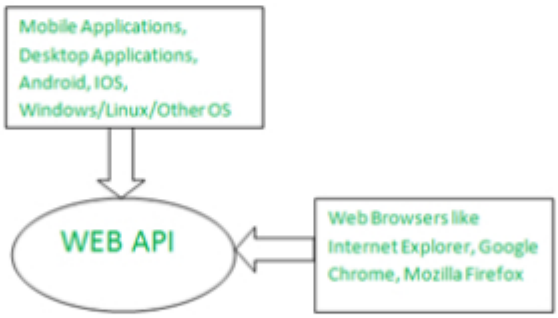


Figure 7-4. Web API

2. **Local APIs:**
   In this types of API, the programmers get the local middleware services. TAPI (Telephony Application Programming Interface), .NET are common examples of Local API's.
3. **Program APIs:**
   It makes a remote program appears to be local by making use of RPC's (Remote Procedural Calls). SOAP is a well-known example of this type of API.

Few other types of APIs:

- **SOAP (SIMPLE OBJECT ACCESS PROTOCOL):** It defines messages in XML format used by web applications to communicate with each other.
- **REST (Representational State Transfer):** It makes use of HTTP to GET, POST, PUT, or DELETE data. It is basically used to take advantage of the existing data.
- **JSON-RPC:** It use JSON for data transfer and is a light-weight remote procedural call defining few data structure types.
- **XML-RPC:** It is based on XML and uses HTTP for data transfer. This API is widely used to exchange information between two or more networks.

Advantages and Disadvantages of API

Following are the advantages of APIs:

- **Efficiency:** API produces efficient, quicker and more reliable results than the outputs produced by human beings in an organization.
- **Flexible delivery of services:** API provides fast and flexible delivery of services according to developer's requirements.
- **Integration:** The best feature of API is that it allows movement of data between various sites and thus enhances integrated user experience.
- **Automation:** As API makes use of robotic computers rather than humans, it produces better and automated results.
- **New functionality**: While using API the developers find new tools and functionality for API exchanges.

Following are the disadvantages of APIs:

- **Cost:** Developing and implementing API is costly at times and requires high maintenance and support from developers.
- **Security issues:** Using API adds another layer of surface which is then prone to attacks, and hence the security risk problem is common in API's.

Introduction to JSON

JSON — short for **JavaScript Object Notation** — is a format for sharing data. As its name suggests, JSON is derived from the JavaScript programming language, but it's available for use by many languages including Python, Ruby, PHP, and Java.

JSON uses the .json extension when it stands alone. When it's defined in another file format (as in .html), it can appear inside of quotes as a JSON string, or it can be an object assigned to a variable. This format is easy to transmit between web server and client or browser.

Very readable and lightweight, JSON offers a good alternative to XML and requires much less formatting. This informational guide will get you up to speed with the data you can use in JSON files, and the general structure and syntax of this format.

A JSON object looks something like this:

```
{
  "sid" : "111",
  "name" : "Raazu Poudel",
  "address" : "Birtamode"
}
```

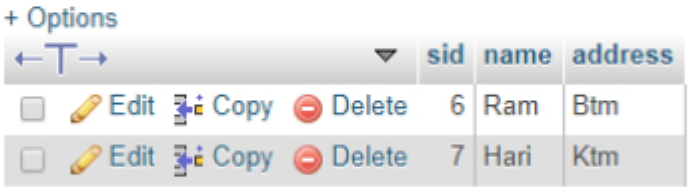
Although this is a very short example, and JSON could be many lines long, this shows that the format is generally set up with two curly braces (or curly brackets) that look like this { } on either end of it, and with key-value pairs populating the space between. Most data used in JSON ends up being encapsulated in a JSON object.

Key-value pairs have a colon between them as in "key" : "value". Each key-value pair is separated by a comma, so the middle of a JSON looks like this: "key" : "value", "key" : "value", "key": "value". In our example above, the first key-value pair is "sid" : "111".

Encoding and Decoding JSON

Encoding using PHP

In this example, we are using **PHP language** for creating JSON. At first we select data from table of the database and after selecting data we **encode** than data in JSON format. We are using localhost as a server for this purpose.



getdata.php

```
<?php
$server="localhost";
$username="root";
$password="";
$database="bcatest";

// Create connection
$conn = new mysqli($server, $username, $password,$database);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
```

```
//retrieving data
 $sql = "SELECT * FROM student";
 $result = $conn->query($sql);

//encoding data in json format
$json = array();
if($result->num_rows>0){
        while ($row = $result->fetch_array()) {
                $json["data"][]=array("sid"=>$row["sid"],"name"=>$row["name"],
                                "address"=>$row["address"]);
        }
}

echo json_encode($json);
$conn->close();
?>
```

Testing in browser:

Decoding JSON using in mobile application using Java

DecodingJSON.java

```
public class DecodingJSON extends Activity{
    @Override
    protected void onCreate(Bundle b){
        super.onCreate(b);
        decodeJson();
    }

    public void decodeJson(){
        try{

            //copy paste json string
            String json_data="{\n" +
                    "  \"data\": [\n" +
                    "    {\n" +
                    "      \"sid\": \"6\",\n" +
                    "      \"name\": \"Ram\",\n" +
                    "      \"address\": \"Btm\"\n" +
                    "    },\n" +
                    "    {\n" +
                    "      \"sid\": \"7\",\n" +
                    "      \"name\": \"Hari\",\n" +
                    "      \"address\": \"Ktm\"\n" +
                    "    }\n" +
```

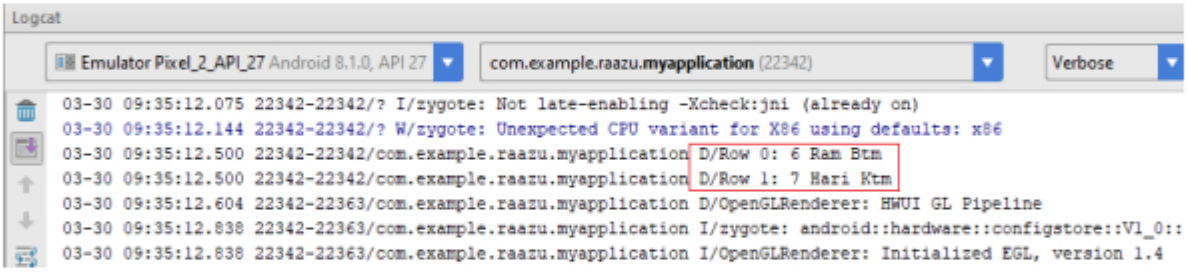
```
                    "  ]\n" +
                    "}";

            //decoding JSON
            JSONObject result=new JSONObject(json_data);
            JSONArray data=result.getJSONArray("data");

            for(int i=0;i<data.length();i++){
                //fetching each row
                JSONObject student=data.getJSONObject(i);
                int sid=student.getInt("sid");
                String name=student.getString("name");
                String address=student.getString("address");

                //displaying data
                Log.d("Row "+i,sid+" "+name+" "+address);
            }
        }catch (Exception ex){
            Log.d("exception",ex.toString());
        }
    }
}
```

Above code produces following output in Logcat:



Retrieving Contents Form Remote Server

For communicating with server, we are using **Volley Library** in our application. **Volley** is an HTTP library that makes networking for Android apps easier and most importantly, faster.

Volley offers the following benefits:

- Automatic scheduling of network requests.
- Multiple concurrent network connections.
- Transparent disk and memory response caching.
- Support for request prioritization.
- Cancellation request API. You can cancel a single request, or you can set blocks or scopes of requests to cancel.
- Ease of customization, for example, for retry and back off.
- Strong ordering that makes it easy to correctly populate your UI with data fetched asynchronously from the network.
- Debugging and tracing tools.

To use volley in your application, we need to add following dependency in our project:

```
dependencies {
    ...
    implementation 'com.android.volley:volley:1.1.1'
}
```

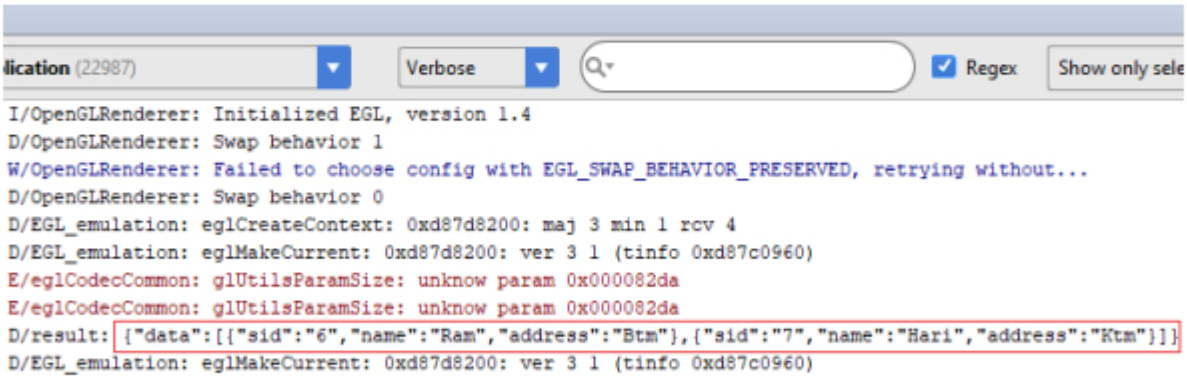
Also, to use Volley, you must add the android.permission.INTERNET permission to your app's manifest. Without this, your app won't be able to connect to the network.

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

```
public void volleyRequest(){
    // Instantiate the RequestQueue.
    RequestQueue queue = Volley.newRequestQueue(this);
    //url for localhost
    String url ="http://10.0.2.2/myproject/getdata.php";

    // Request a string response from the provided URL.
 StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    //displaying response string in logcat
                    Log.d("result",response);
                }
            }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            //displaying error response message
            Log.d("exception",error.toString());
        }
    });

    // Add the request to the RequestQueue.
    queue.add(stringRequest);
}
```



Displaying Retrieved data in RecyclerView

json_example.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_margin="5dp"
    android:layout_height="match_parent">

    <android.support.v7.widget.RecyclerView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/recyclerview" />

</RelativeLayout>
```

list_items.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:id="@+id/relat"
    android:layout_height="wrap_content">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:layout_margin="10dp"
        android:text="Id"
        android:textStyle="bold"
        android:id="@+id/txtId" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:id="@+id/txtName"
        android:textStyle="bold"
        android:layout_toRightOf="@+id/txtId"
        android:layout_marginTop="10dp"
        android:text="Name" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:id="@+id/txtAddress"
        android:text="Address"
        android:layout_marginLeft="10dp"
        android:layout_below="@+id/txtName" />

</RelativeLayout>
```

DataModel.java

```
public class DataModel {
    private int id;
    private String name;
    private String address;

    public DataModel(int id, String name, String address){
        this.id=id;
        this.name=name;
        this.address=address;
    }

    public int getId(){
        return id;
    }

    public String getName(){
        return name;
    }

    public String getAddress(){
        return address;
    }
}
```

RecyclerViewAdapter.java

```
public class RecyclerViewAdapter extends
RecyclerView.Adapter<RecyclerViewAdapter.ViewHolder> {
    Activity context;
    ArrayList<DataModel> data;

    public RecyclerViewAdapter(Activity context, ArrayList<DataModel>
            data){
        this.context=context;
        this.data=data;
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType)
{
        LayoutInflater layoutInflater = LayoutInflater.from(context);
        View listItem= layoutInflater.inflate(R.layout.list_items,
                parent, false);
        ViewHolder viewHolder = new ViewHolder(listItem);
        return viewHolder;
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        final DataModel current=data.get(position);
        holder.txtId.setText(current.getId()+"");
        holder.txtName.setText(current.getName());
        holder.txtAddress.setText(current.getAddress());
    }
```

```
    @Override
    public int getItemCount() {
        return data.size();
    }

    public static class ViewHolder extends RecyclerView.ViewHolder {
        TextView txtId,txtName,txtAddress;
        ImageView imageView;
        public ViewHolder(View itemView) {
            super(itemView);
            txtId = itemView.findViewById(R.id.txtId);
            txtName = itemView.findViewById(R.id.txtName);
            txtAddress = itemView.findViewById(R.id.txtAddress);
        }
    }
}
```

JsonExample.java

```
public class JsonExample extends AppCompatActivity{
    RecyclerView recyclerView;
    RecyclerView.Adapter recyclerAdapter;
    RecyclerView.LayoutManager layoutManager;
    @Override
    protected void onCreate(Bundle b){
        super.onCreate(b);
        setContentView(R.layout.json_example);
        recyclerView=findViewById(R.id.recyclerview);
        volleyRequest();
    }

    public void volleyRequest(){
        // Instantiate the RequestQueue.
        RequestQueue queue = Volley.newRequestQueue(this);
        //url for localhost
        String url ="http://10.0.2.2/myproject/getdata.php";

        // Request a string response from the provided URL.
        StringRequest stringRequest = new StringRequest
          (Request.Method.GET, url,
                new Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {
                        //passing data for decoding
                        decodeJson(response);
                    }
                }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                //displaying error response message
                Log.d("exception",error.toString());
            }
        });

        // Add the request to the RequestQueue.
        queue.add(stringRequest);
    }
```

```
public void decodeJson(String response){
        try{
            ArrayList<DataModel> data=new ArrayList<>();
            JSONObject result=new JSONObject(response);
            JSONArray array=result.getJSONArray("data");
            for(int i=0;i<array.length();i++){
                //fetching each row
                JSONObject student=array.getJSONObject(i);
                int sid=student.getInt("sid");
                String name=student.getString("name");
                String address=student.getString("address");
                DataModel dataModel=new DataModel(sid,name,address);
                data.add(dataModel);
            }

            //plotting data in recyclerview
            layoutManager=new LinearLayoutManager(this);
            recyclerView.setLayoutManager(layoutManager);
            recyclerAdapter=new RecyclerViewAdapter
                                        (JsonExample.this,data);
            recyclerView.setAdapter(recyclerAdapter);

        }catch (Exception ex){
            Log.d("exception",ex.toString());
        }
    }
}
```

Above code produces following output:

Figure 7-5. Output Displaying retrieved contents from server in RecyclerView

Sending Contents to Remote Server

send_data.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:orientation="vertical"
    android:layout_margin="5dp"
    android:layout_height="match_parent">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Id"
        android:inputType="number"
        android:id="@+id/edtId" />

   <EditText
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:hint="Enter Name"
       android:layout_below="@+id/edtId"
       android:id="@+id/edtName" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter Address"
        android:layout_below="@+id/edtName"
        android:id="@+id/edtAddress" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edtAddress"
        android:id="@+id/btnSubmit"
        android:layout_centerHorizontal="true"
        android:text="Submit" />

</RelativeLayout>
```

SendData.java

```
public class SendData extends AppCompatActivity {
    EditText edtId,edtName,edtAddress;
    Button btnSubmit;
    @Override
    protected void onCreate(Bundle b){
        super.onCreate(b);
        setContentView(R.layout.send_data);

        edtId=findViewById(R.id.edtId);
        edtName=findViewById(R.id.edtName);
        edtAddress=findViewById(R.id.edtAddress);
        btnSubmit=findViewById(R.id.btnSubmit);
```

```
        btnSubmit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                volleyRequest();
            }
        });

    }

    public void volleyRequest(){
        // Instantiate the RequestQueue.
        RequestQueue queue = Volley.newRequestQueue(this);
        //url for localhost
        String url ="http://10.0.2.2/myproject/setdata.php";
        // Request a string response from the provided URL.
        StringRequest stringRequest = new StringRequest
          (Request.Method.POST, url,
                new Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {
                        //displaying response message
                        Toast.makeText(getApplicationContext(),response,
                                Toast.LENGTH_LONG).show();
                    }
                }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                //displaying error response message
                Log.d("exception",error.toString());
            }
        }){
            //sending data to server
            @Override
            protected HashMap<String,String> getParams(){
                HashMap<String,String> params = new HashMap<>();
                params.put("sid",edtId.getText().toString());
                params.put("name",edtName.getText().toString());
                params.put("address",edtAddress.getText().toString());
                return params;
            }

        };;
        // Add the request to the RequestQueue.
        queue.add(stringRequest);
    }
}
```

setdata.php

```
<?php

//getting post data
$sid=$_POST["sid"];
$name=$_POST["name"];
$address=$_POST["address"];
```

```
//server credentials
$server="localhost";
$username="root";
$password="";
$database="bcatest";
// Create connection
$conn = new mysqli($server, $username, $password,$database);
// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

//inserting data
 $sql = "INSERT INTO student(sid,name,address)
VALUES('".$sid."','".$name."','".$address."')";
 $result = $conn->query($sql);

 if($result)
        echo"Data Inserted Successfully !";
 else
        echo"Failed to Insert Data !";
$conn->close();
?>
```

Above code produces following output:

Figure 7-6. Sending data to remote server

Implementing Google Maps in Android Application

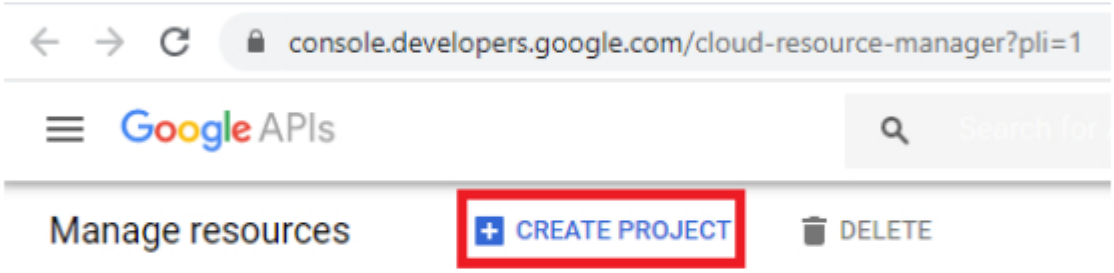
Android allows us to integrate Google Maps in our application. For this Google provides us a library via Google Play Services for using maps. **In order to use the Google Maps API, you must register your application on the Google Developer Console and enable the API**.

Steps for Getting Google Maps API Key

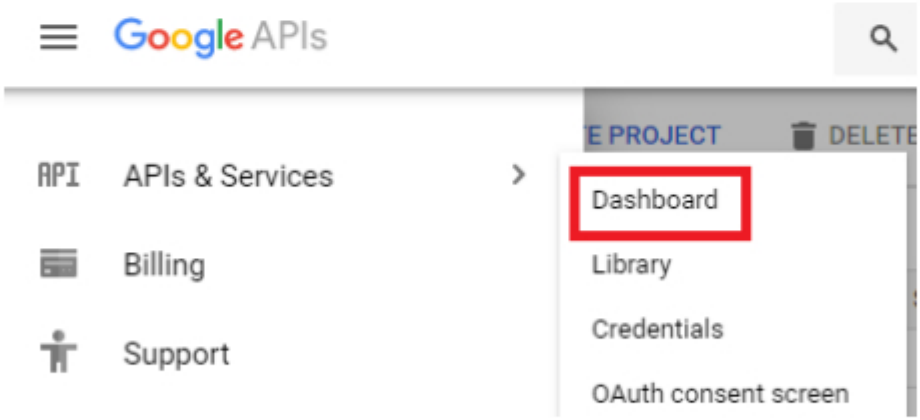
An API key is needed to access the Google Maps servers. This key is free and you can use it with any of your applications. If you haven't created project, you can follow the below steps to get started:

Step 1: Open Google developer console and sign in with your Gmail account: https://console.developers.google.com/project

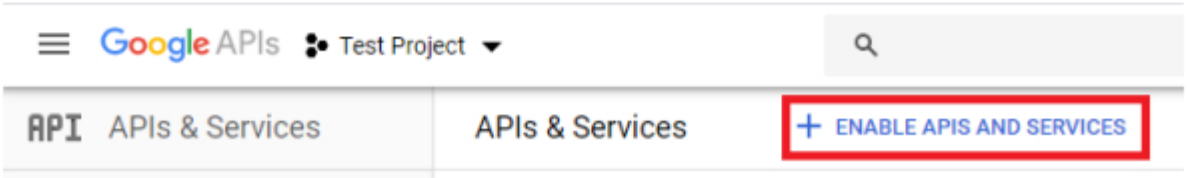
Step 2: Now create new project. You can create new project by clicking on the **Create Project** button and give name to your project.



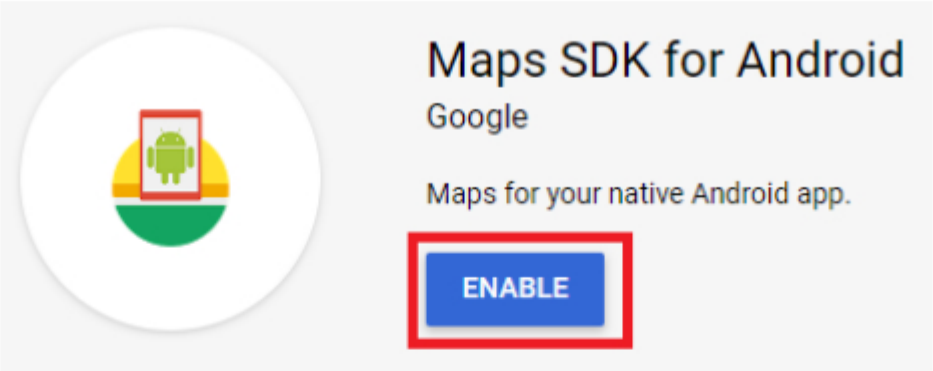
Step 3: Now click on APIs & Services and open **Dashboard** from it.



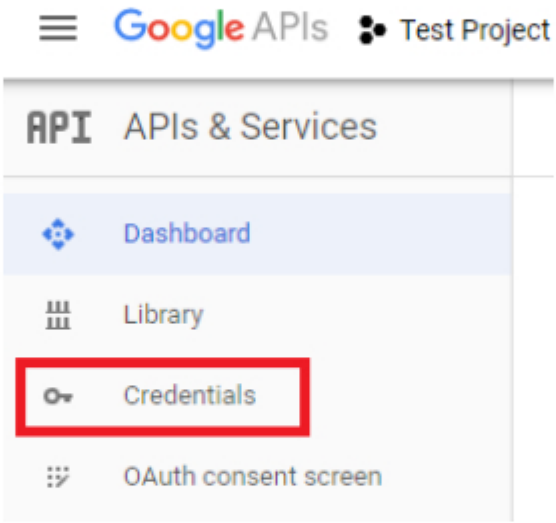
Step 4: Now click **Enable APIS AND SERICES**.



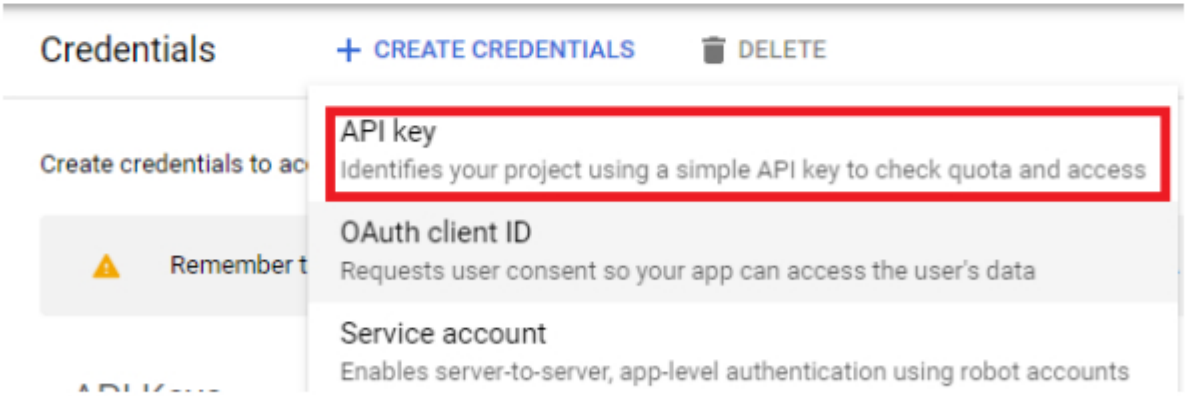
Step 5: Now search for Google Map Android API and enable it.



Step 6: Now refresh page and go to **Credentials.**



Step 7: Now click on Create credentials and choose API key. It will create API key to integrate maps in your application.



Step 8: Now API your API key will be generated. Copy it and save it somewhere as we will need it when implementing Google Map in our Android project.

Developing Google Maps Application in Android

Following are the steps for integrating google maps in your application:

Step 1: Add following permission in your **manifest** file.

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

Step 2: Add your generated API Key in your **manifest** file inside application tag as follows.

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="AIzaSyCWIdlyqlFhC7lOthG164H42heV1F7N3v0" />
```

Step 3: Add following dependency in **build.gradle** file .

```
dependencies {
    . . .
    . . .
    implementation 'com.google.android.gms:play-services-maps:15.0.1'
}
```

Step 4: Now create layout resource file to display google map.

map_activity.xml

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

    <fragment xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        tools:context=".MapsActivity"
        android:layout_width="match_parent"
```

```
        android:layout_height="match_parent" />

</RelativeLayout>
```

Step 5: Now finally create an Activity by implementing **OnMapReadyCallback** interface.

MapsActivity.java

```
public class MapsActivity extends AppCompatActivity implements
                                        OnMapReadyCallback {
    GoogleMap mMap;
    @Override
    protected void onCreate(Bundle b){
        super.onCreate(b);
        setContentView(R.layout.map_activity);

        // Obtain the SupportMapFragment and get notified when the map is
               ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment).
               getSupportFragmentManager().findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        // Adding latitude and longitude
        LatLng location = new LatLng(26.644096, 87.989391);
        //Adding red marker to point location
        mMap.addMarker(new MarkerOptions().position(location).
                              title("Marker in Birtamode"));
        //Moving camera to desired location
        mMap.moveCamera(CameraUpdateFactory.newLatLng(location));
        //Adding zoom effect
        mMap.animateCamera(CameraUpdateFactory.zoomTo(12.0f) );
    }
}
```

Following output will be produced after running application:

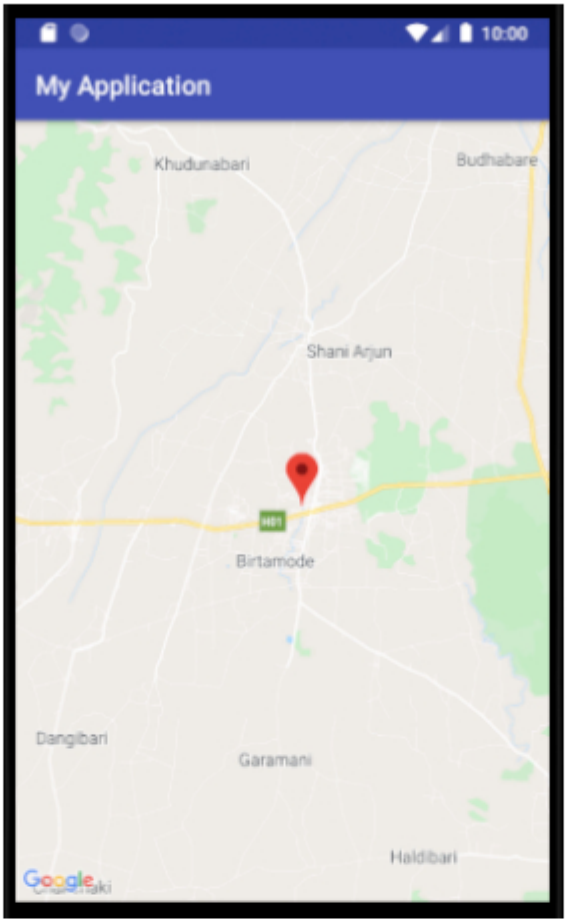


Figure 7-7. Output Demonstrating Google Map.

Procedure for Publishing Application in Google Play Store

To upload your application in google play store you must perform following tasks:

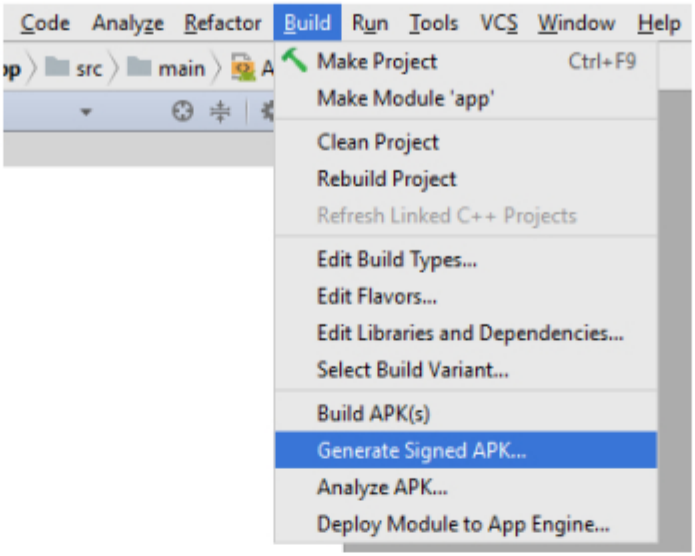
- First generate **signed APK** of your application.
- Upload signed APK in **google play console**.

Generating Signed APK for App Release

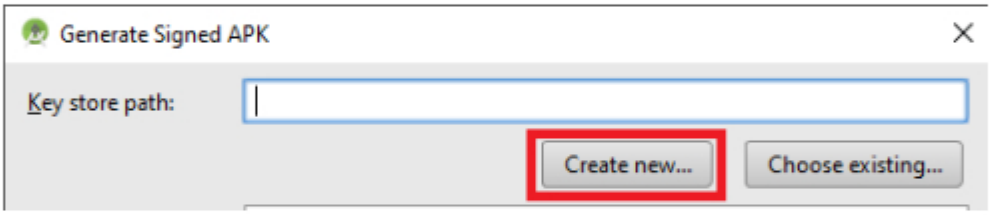
Following are the steps used to generate signed APK:

Note: Don't use **com.example….** as a package name because **example** is not supported by play store. Instead you can rename your package name.

Step 1: Go to menu and select **Build -> Generate Signed APK**.

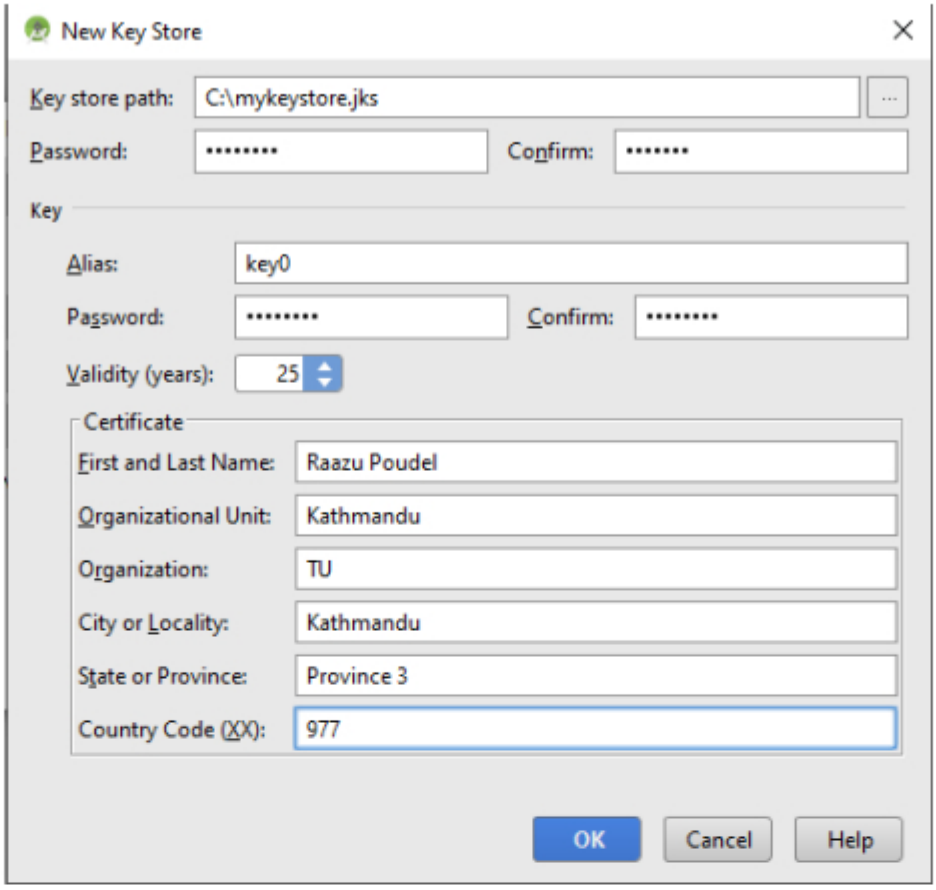


Step 2: Now click **Create new keystore file**. This keystore file is necessary to upload new app and for updating existing app. If you have already created keystore file, then you can click **Choose existing keystore file**.

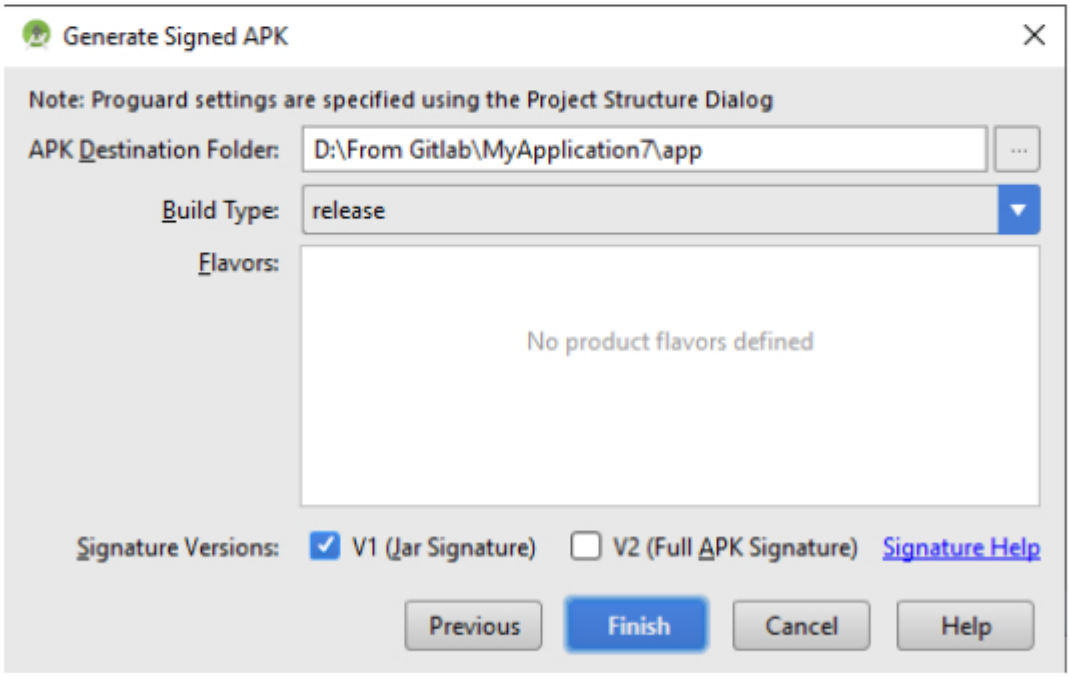


Step 3: Now fill up information for creating new keystore file and click ok. Example is shown below.

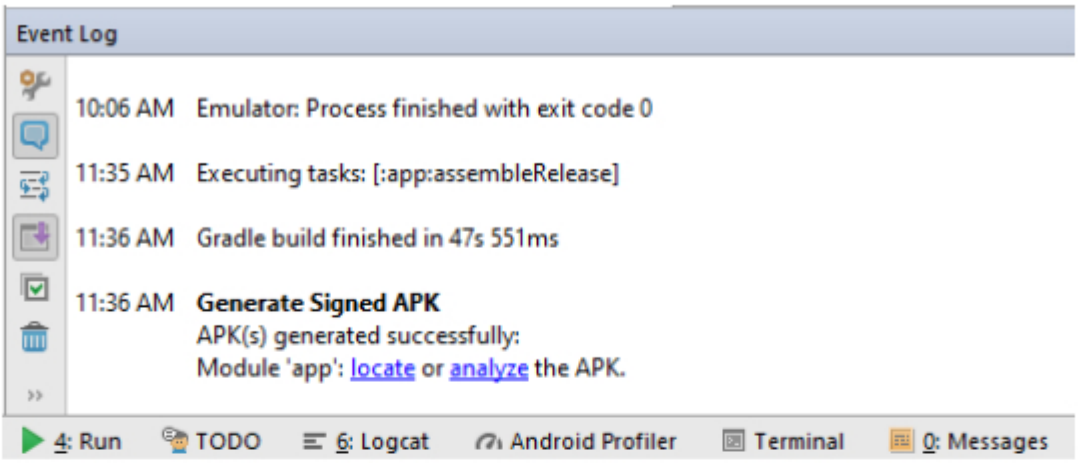
Note: Please remember keystore file, username and password because if it is lost, then you will be unable to update your application.



Step 4: Now select **V1 (Jar Signature)** and click **Finish**. This will create signed APK for release.



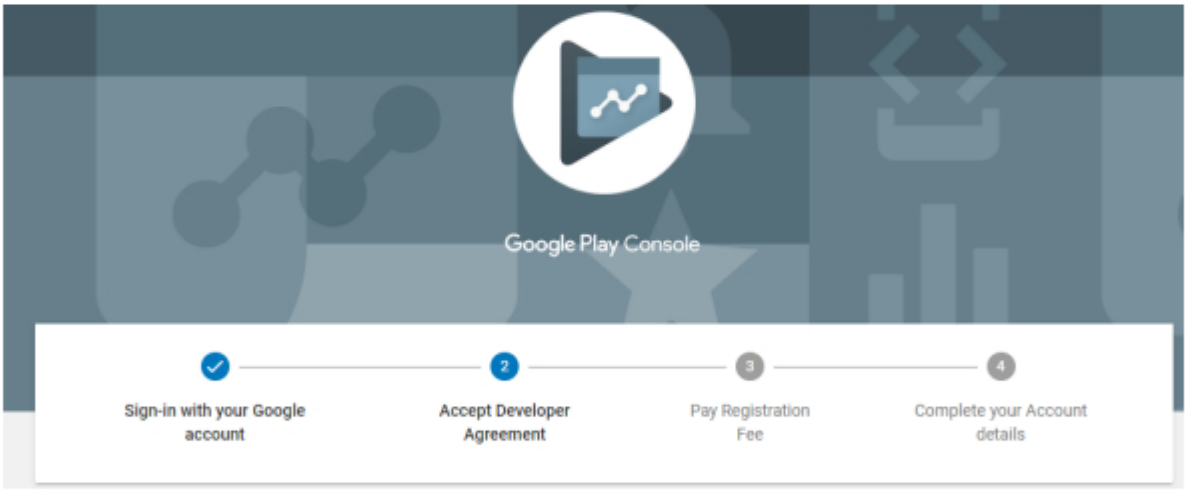
Step 5: Now click Event Log and locate to the folder where signed APK has been placed. You can get your signed APK file inside **release** folder.



Upload Signed APK in Play Store

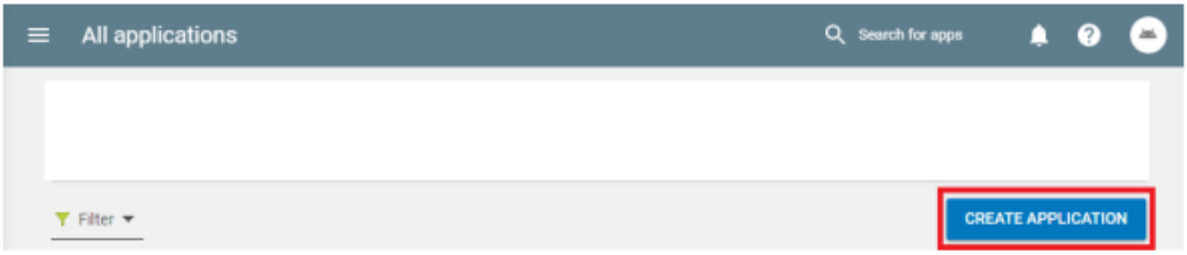
Following steps are used to upload signed APK to play store.

Step 1: Visit https://play.google.com/apps/publish/ to upload your app in play store. Following page will be displayed.

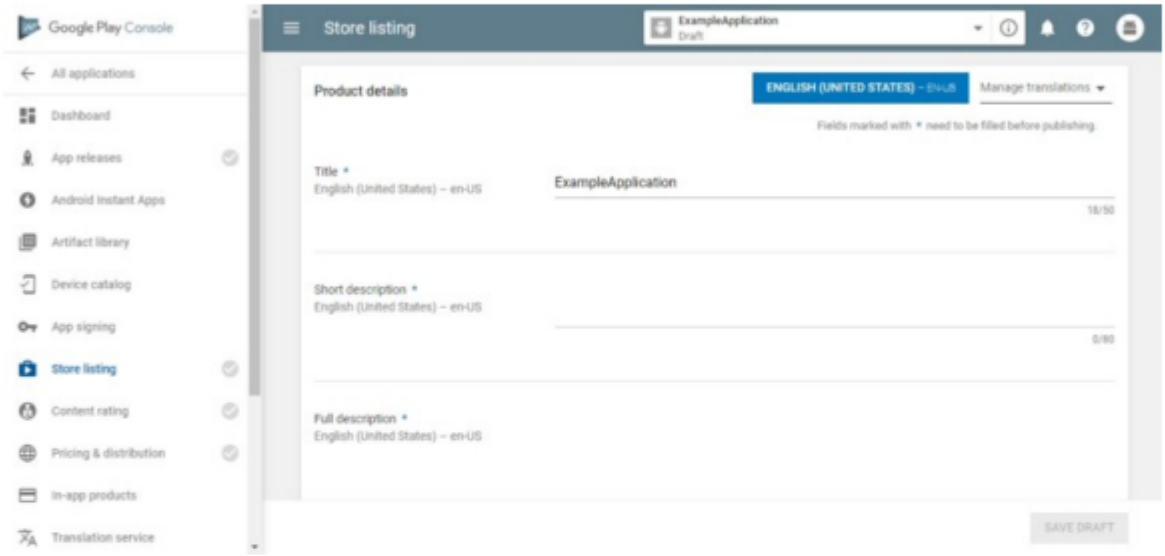


Here you need to create your developer account for lifetime by paying $25 using your credit card. There is no limitation for number of apps. You can upload multiple apps using a single account.

Step 2: Once you've created developer account following page will be displayed on screen. Click **Create Application**.



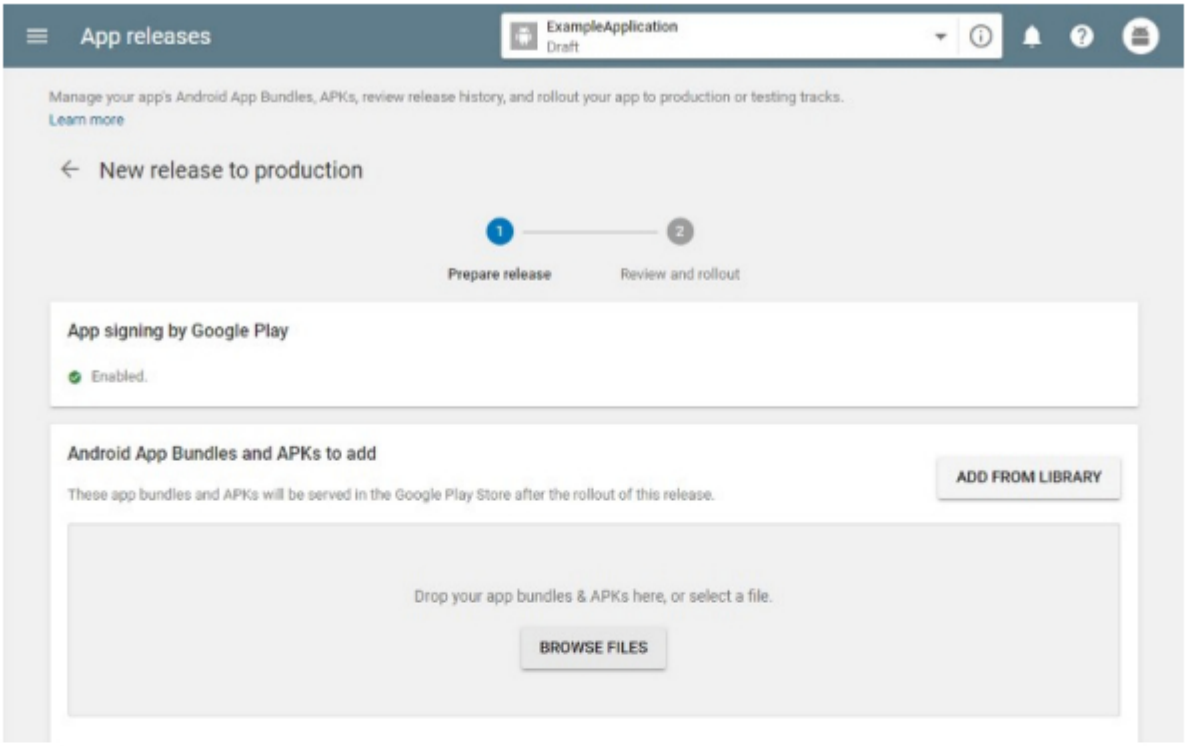
Step 3: Next, give your application a title and the default language. You will now be moved to the **Store listing** screen.



In store listing page you must add everything related to how your app will look in the store. Following information must be added in this page:

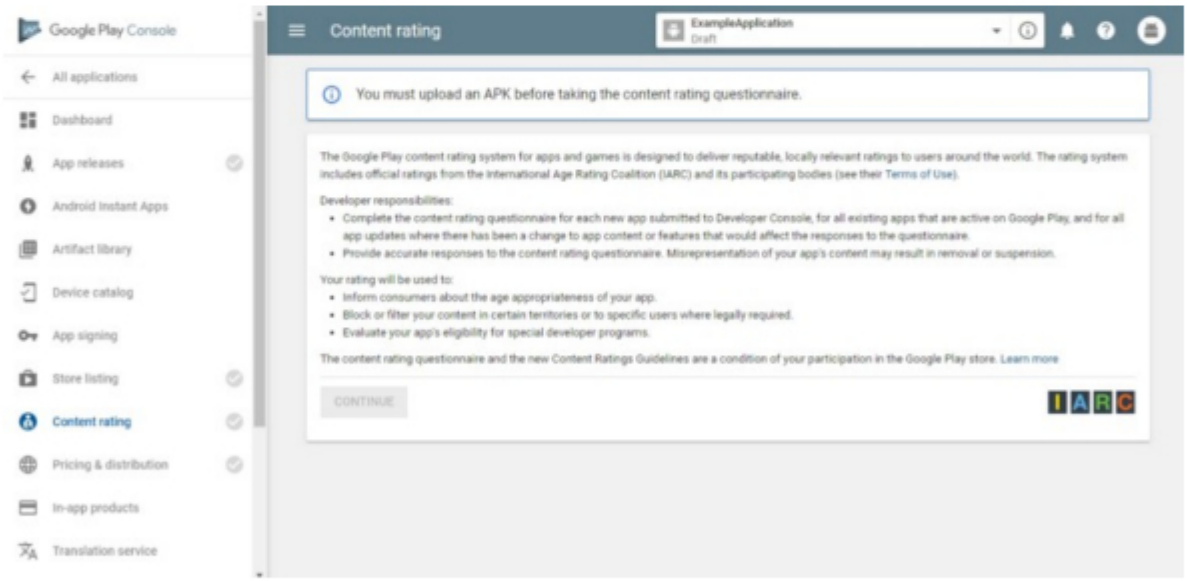
- A short description of your application
- A full description
- Screenshots (at least 2)
- A Hi Resolution Icon
- A Feature Graphic
- The application's type and its category
- Your email
- Your privacy policy (if you don't have one, check the Not submitting a privacy policy checkbox)

Step 4: Now go to the **App Releases** page and click **Create Release** button. Following page will be displayed on screen.



Here you can upload your signed APK. Here you can also give the release a name and specify what is new in this release. ***Pay special attention to whatever you type in what is new in this release because it will appear in the Play store under the What's New section***. Once you are done, you can press the ***Review*** button on the lower right corner of the page.

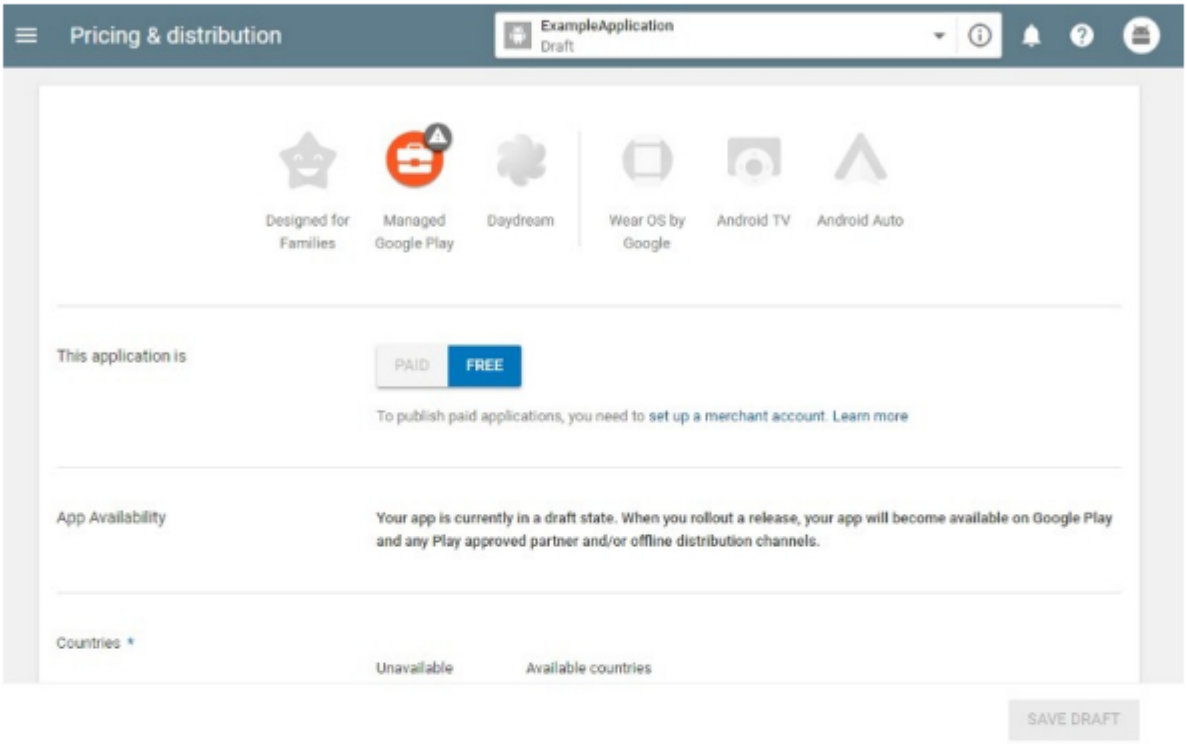
Step 5: Next, Click **Content Rating**.



After pressing the Continue button, you will be asked several questions regarding the content of your application. Just follow them in order, filling one by one.

Step 6: Last but not least, go to ***Pricing & distribution*** page. Be aware to fill in all the required fields. In this last page, you can decide on:

- If your application will be free or not
- Which countries your application will be available to
- If it contains ads
- User programs



Step 7: After filling up all the information regarding Store Listings, Content Rating, Pricing & Distribution etc. Now you can go to **App Release page** to publish your application. To publish app in play store, click **Rollout** Button.

Exercise

1. What do you mean by SQLite? Explain its features.
2. Explain advantages and disadvantages of using SQLite.
3. Differentiate SQLite and SQL with example.
4. How can you establish connection using SQLite in android? Explain with the help of suitable example.
5. Explain the process of creating Database and Tables using SQLite in android.
6. Develop an android application to demonstrate basic crud operation using SQLite.
7. Develop a to-do list application using SQLite.
8. Develop a phone directory application using SQLite.
9. Develop an android application which track your daily income and expenses. Also your app should display daily savings.
10. What do you mean by API? Explain its types.
11. Define JSON. Explain JSON as a tool for creating API with example.

12. Explain the advantages of JSON over XML.
13. How do your communication your application with remote server? Explain with the help of suitable example.
14. Develop an android application that demonstrates retrieval of contents from remote server.
15. Develop an android application that demonstrates sending of contents to remote server.
16. Develop a phone directory application (Phone records must be stored in remote server).
17. Explain the procedure for generating API key for displaying google map in your application.
18. Develop an android application that displays google map.
19. Explain the procedure for generating signed APK.
20. Explain the procedure for publishing signed APK in play store.