

Unit-7: Advance Android Concept

Local Database with SQLite

SQLite is a open source SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation.

SQLite 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.

Android has built in SQLite database implementation. It is available locally over the device(mobile & tablet) and contains data in text format. It carry light weight data and is suitable with many languages. So, it doesn't required any administration or setup procedure of the database.

For creating, updating and other operations you need to create a subclass of SQLiteOpenHelper class. SQLiteOpenHelper is a helper class to manage database creation and version management. It provides two methods....

1. onCreate(SQLiteDatabase db),
2. onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion).

SQLiteOpenHelper

The SQLiteOpenHelper is responsible for opening the database if it exists, creating the database if it does not exist and upgrading if required. The SQLiteOpenHelper only require the DATABASE_NAME to create database. After extending SQLiteOpenHelper you will need to implement its methods onCreate, onUpgrade and constructor.

onCreate method

onCreate(SQLiteDatabase sqLiteDatabase) method is called only once throughout the application lifecycle. It will be called whenever there is a first call to getReadableDatabase() or getWritableDatabase() function available in super SQLiteOpenHelper class. So SQLiteOpenHelper class call the onCreate() method after creating database and instantiate SQLiteDatabase object. Database name is passed in constructor call.

onUpgrade method

onUpgrade(SQLiteDatabase db,int oldVersion, int newVersion) is only called whenever there is a updation in existing version. So to update a version we have to increment the value of version variable passed in the superclass constructor.

In the onUpgrade method we can write queries to perform whatever action is required. In most examples you will see that existing table(s) are being dropped and again the onCreate() method is being called to create tables again. But it's not mandatory to do so and it all depends upon your requirements.

We have to change the database version if we have added a new row in the database table. If we have a requirement that we don't want to lose existing data in the table then we can write an alter table query in the onUpgrade(SQLiteDatabase db,int oldVersion, int newVersion) method.

Example:**activity_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/activity_main">
    <TableRow>
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="User Registration:"
            android:textSize="@android:dimen/app_icon_size"
            android:layout_column="1"
            android:layout_marginLeft="50dp"
            android:textStyle="italic"
            android:textColor="@color/cardview_dark_background"
        />
    </TableRow>
```

```
<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="username:"
        android:textSize="@android:dimen/app_icon_size"
        android:layout_column="1"/>

    <EditText
        android:id="@+id/editName"
        android:layout_width="80dp"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:layout_column="2"
        android:hint="Enter Name"
        android:textSize="@android:dimen/app_icon_size"/>
</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_column="1"
        android:text="password:"
        android:textSize="@android:dimen/app_icon_size"/>
```

```
<EditText
    android:id="@+id/editPass"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:hint="Enter Password"
    android:layout_column="2"
    android:textSize="@android:dimen/app_icon_size"/>
</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="add_user"
        android:layout_column="1"
        android:textSize="@android:dimen/app_icon_size"/>

</TableRow>
<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="view_data"
        android:layout_column="1"
        android:textSize="@android:dimen/app_icon_size"/>

</TableRow>
```

```

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <EditText
        android:id="@+id/editText3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_column="1"
        android:textSize="40dp"
        android:hint="Current Name" />
</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <EditText
        android:id="@+id/editText5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="New Name"
        android:textSize="40dp"
        android:layout_column="1" />
</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <Button
        android:text="update Name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button3"
        android:layout_column="1"
        android:textSize="@android:dimen/app_icon_size"/>
</TableRow>

```

```
<TableRow
    android:layout_height="wrap_content"
    android:layout_width="match_parent">
    <EditText
        android:id="@+id/editText6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Enter Name to Delete Data"
        android:inputType="text"
        android:textSize="40dp"
        android:layout_column="1" />
</TableRow>

<TableRow
    android:layout_height="wrap_content"
    android:layout_width="match_parent">
    <Button
        android:id="@+id/button4"
        android:text="delete Name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_column="1"
        android:textSize="@android:dimen/app_icon_size"/>
</TableRow>
</TableLayout>
```

MainActivity.java

```
package com.example.myapplication;

import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {
    EditText Name, Pass, updateold, updatenew, delete;
    myDbAdapter helper;

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

        Name = findViewById(R.id.editName);
        Pass = findViewById(R.id.editPass);
        updateold = findViewById(R.id.editText3);
        updatenew = findViewById(R.id.editText5);
        delete = findViewById(R.id.editText6);

        helper = new myDbAdapter(this);

        Button addButton = findViewById(R.id.button);
        addButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                addUser(v);
            }
        });

        Button viewButton = findViewById(R.id.button2);
        viewButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                viewdata(v);
            }
        });
    }
}
```

```

    }
});

Button updateButton = findViewById(R.id.button3);
updateButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        update(v);
    }
});

Button deleteButton = findViewById(R.id.button4);
deleteButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        delete(v);
    }
});
}

public void addUser(View view) {
    String t1 = Name.getText().toString();
    String t2 = Pass.getText().toString();
    if (t1.isEmpty() || t2.isEmpty()) {
        Message.message(getApplicationContext(), "Enter Both Name and
        Password");
    } else {
        long id = helper.insertData(t1, t2);
        if (id <= 0) {
            Message.message(getApplicationContext(), "Insertion Unsuccessful");
            Name.setText("");
            Pass.setText("");
        } else {
            Message.message(getApplicationContext(), "Insertion Successful");
            Name.setText("");
            Pass.setText("");
        }
    }
}
}

```



```
public void viewdata(View view) {
    String data = helper.getData();
    Message.message(this, data);
}

public void update(View view) {
    String u1 = updateold.getText().toString();
    String u2 = updatenew.getText().toString();
    if (u1.isEmpty() || u2.isEmpty()) {
        Message.message(getApplicationContext(), "Enter Data");
    } else {
        int a = helper.updateName(u1, u2);
        if (a <= 0) {
            Message.message(getApplicationContext(), "Unsuccessful");
            updateold.setText("");
            updatenew.setText("");
        } else {
            Message.message(getApplicationContext(), "Updated");
            updateold.setText("");
            updatenew.setText("");
        }
    }
}

public void delete(View view) {
    String unname = delete.getText().toString();
    if (unname.isEmpty()) {
        Message.message(getApplicationContext(), "Enter Data");
    } else {
        int a = helper.delete(unname);
        if (a <= 0) {
            Message.message(getApplicationContext(), "Unsuccessful");
            delete.setText("");
        } else {
            Message.message(this, "DELETED");
            delete.setText("");
        }
    }
}
}
```

MyDbAdapter.java

```
package com.example.myapplication;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

class myDbAdapter {
    myDbHelper myhelper;

    public myDbAdapter(Context context) {
        myhelper = new myDbHelper(context);
    }

    public long insertData(String name, String pass) {
        SQLiteDatabase dbb = myhelper.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(myDbHelper.NAME, name);
        contentValues.put(myDbHelper.MyPASSWORD, pass);
        return dbb.insert(myDbHelper.TABLE_NAME, null, contentValues);
    }

    public String getData() {
        SQLiteDatabase db = myhelper.getWritableDatabase();
        String[] columns = {myDbHelper.UID, myDbHelper.NAME,
myDbHelper.MyPASSWORD};
        Cursor cursor = db.query(myDbHelper.TABLE_NAME, columns, null, null,
null, null, null);
        StringBuffer buffer = new StringBuffer();

        int indexUID = cursor.getColumnIndex(myDbHelper.UID);
        int indexName = cursor.getColumnIndex(myDbHelper.NAME);
        int indexPassword = cursor.getColumnIndex(myDbHelper.MyPASSWORD);

        while (cursor.moveToNext()) {
            if (indexUID != -1 && indexName != -1 && indexPassword != -1) {
                int cid = cursor.getInt(indexUID);
```

```

        String name = cursor.getString(indexName);
        String password = cursor.getString(indexPassword);
        buffer.append(cid + " " + name + " " + password + "\n");
    }
}
cursor.close(); // Close the cursor to avoid memory leaks
return buffer.toString();
}

public int delete(String uname) {
    SQLiteDatabase db = myhelper.getWritableDatabase();
    String[] whereArgs = {uname};
    return db.delete(myDbHelper.TABLE_NAME, myDbHelper.NAME + " = ?",
whereArgs);
}

public int updateName(String oldName, String newName) {
    SQLiteDatabase db = myhelper.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put(myDbHelper.NAME, newName);
    String[] whereArgs = {oldName};
    return db.update(myDbHelper.TABLE_NAME, contentValues,
myDbHelper.NAME + " = ?", whereArgs);
}

static class myDbHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "myDatabase"; // Database
Name
    private static final String TABLE_NAME = "myTable"; // Table Name
    private static final int DATABASE_Version = 1; // Database Version
    private static final String UID = "_id"; // Column I (Primary Key)
    private static final String NAME = "Name"; // Column II
    private static final String MyPASSWORD = "Password"; // Column III

    private static final String CREATE_TABLE = "CREATE TABLE " +
TABLE_NAME +
        "(" + UID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
NAME + " VARCHAR(255), " + MyPASSWORD + " VARCHAR(225));";

    private static final String DROP_TABLE = "DROP TABLE IF EXISTS " +

```

```
TABLE_NAME;
private Context context;

public myDbHelper(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_Version);
    this.context = context;
}

public void onCreate(SQLiteDatabase db) {
    try {
        db.execSQL(CREATE_TABLE);
    } catch (Exception e) {
        Message.message(context, "" + e);
    }
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    try {
        Message.message(context, "OnUpgrade");
        db.execSQL(DROP_TABLE);
        onCreate(db);
    } catch (Exception e) {
        Message.message(context, "" + e);
    }
}
}
```

Message.java

```
package com.example.myapplication;
```

```
import android.content.Context;
```

```
import android.widget.Toast;
```

```
public class Message {
```

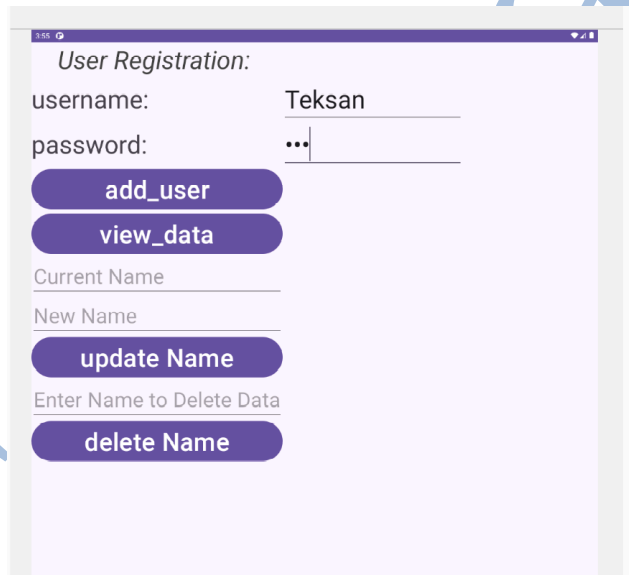
```
    public static void message(Context context, String message) {
```

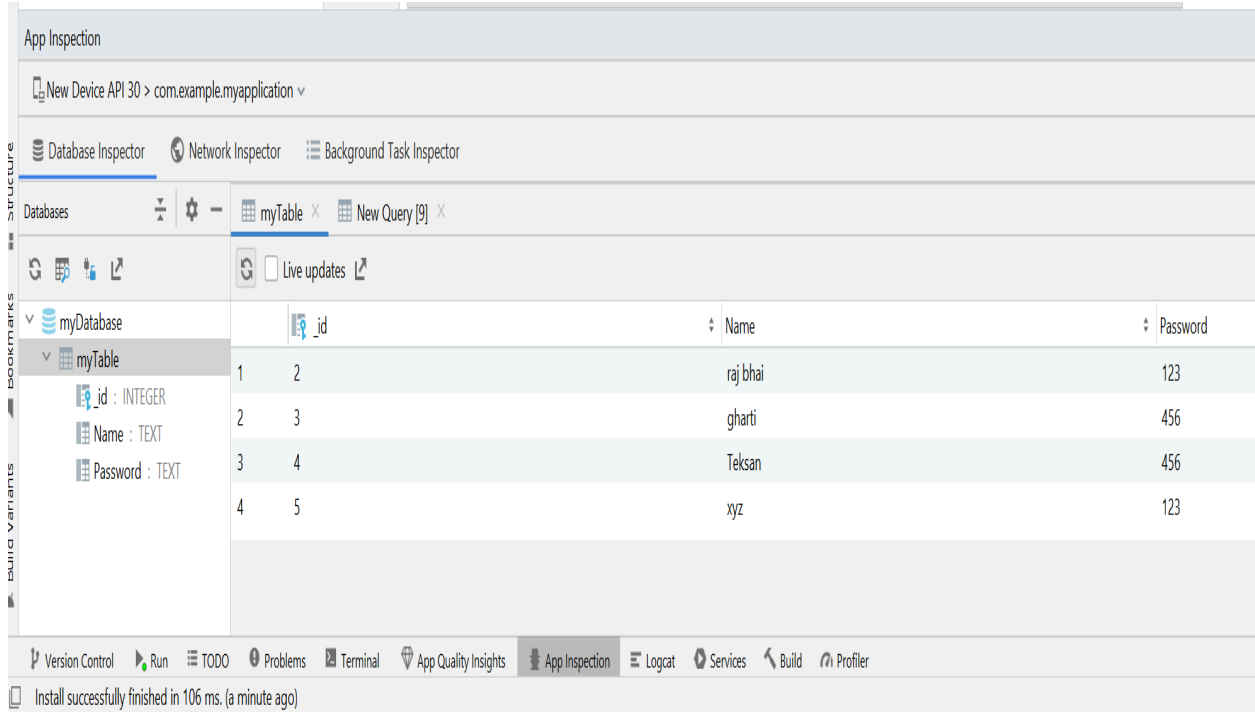
```
        Toast.makeText(context, message, Toast.LENGTH_SHORT).show();
```

```
    }
```

```
}
```

Output:





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.

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).

Some example of third party APIs are:

- YouTube API - Allows you to display videos on a web site.
- Twitter API - Allows you to display Tweets on a web site.
- Facebook API - Allows you to display Facebook info on a web site.

Types of APIs:

There are three basic forms of API

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. Open source, accessible by a large number of clients (phones, tablets, PCs). Examples: REST, SOAP.

Local APIs:

In this type of API, the programmers get the local middleware services. Local APIs provide local middleware services to programmers. Used for interacting with software on the same machine. TAPI (Telephony Application Programming Interface), .NET are common examples of Local API's.

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.

Above are the various types and forms of API's extensively used over web network to exchange information and to enhance communication between them.

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.

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 or JavaScript Object Notation is a lightweight text-based open standard designed for human-readable data interchange. The JSON format was originally specified by Douglas Crockford, and is described in RFC 4627. The official Internet media type for JSON is application/json. The JSON filename extension is .json. It can be use within various programming languages such as PHP, PERL, Python, Ruby, Java, etc.

JSON (JavaScript Object Notation) is a straightforward data exchange format to interchange the server's data, and it is a better alternative for XML. This is because JSON is a lightweight and structured language. Android supports all the JSON classes such as JSONStringer, JSONObject, JSONArray, and all other forms to parse the JSON data and fetch the required information by the program.

JSON's main advantage is that it is a language-independent, and the JSON object will contain data like a key/value pair. In general, JSON nodes will start with a square bracket ([]) or with a curly bracket ({}). The square and curly bracket's primary difference is that the square bracket ([]) represents the beginning of a JSONArray node. Whereas, the curly bracket ({} represents a JSONObject. So one needs to call the appropriate method to get the data. Sometimes JSON data start with [. We then need to use the getJSONArray() method to get the data. Similarly, if it starts with {, then we need to use the getJSONObject() method. The syntax of the JSON file is as following:

Example:

Let's take example of json array.

```
{ "user" :
  [
    {"name":"Rohan","designation":"Manager","Location":"KTM"},
    {"name":"sujita","designation":"CEO","Location":"Butwal"},
    {"name":"Maila bhai","designation":"Programmer","Location":"Gorkha"},
  ]
}
```

Same can be written as

```
return "{ \"users\" :\" +
    \"[\" +
    \"{\\\"name\\\":\\\"Rohan\\\",\" +
    \\\"designation\\\":\\\"Manager\\\",\" +
    \\\"location\\\":\\\"KTM\\\"},\" +

    \"{\\\"name\\\":\\\"Sujita\\\",\" +
    \\\"designation\\\":\\\"CEO\\\",\" +
    \\\"location\\\":\\\"Butwal\\\"},\" +

    \"{\\\"name\\\":\\\"Maila bhai\\\",\" +
    \\\"designation\\\":\\\"Programmer\\\",\" +
    \\\"location\\\":\\\"Gorkha\\\"}\" +

    \"] }\";
```

Hence,

- JSON stands for JavaScript Object Notation.
- JSON is lightweight data-interchange format.
- JSON is easy to read and write than XML.
- JSON is language independent.
- JSON supports array, object, string, number and values.

JSON syntax is basically considered as a subset of JavaScript syntax; it includes the following

- Data is represented in name/value pairs.
- Curly braces hold objects and each name is followed by ':'(colon), the name/value pairs are separated by , (comma).
- Square brackets hold arrays and values are separated by ,(comma).

Program: Example of android JSON parsing

Activity_main.xml

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

<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical" >

<!--This listView will display the list items-->
<ListView
    android:id="@+id/user_list"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:dividerHeight="1dp" />

</LinearLayout>
```

MainActivity.java

```
package com.example.myapplication;

import android.os.Bundle;
import android.util.Log;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import androidx.appcompat.app.AppCompatActivity;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;

public class MainActivity extends AppCompatActivity {
    // JSON object in the form of input stream
    private String getListData() {
        return "{ \"users\" : " +
            "[" +
            "{ \"name\" : \"Rohan\" , " +
            "\"designation\" : \"Manager\" , " +
            "\"location\" : \"KTM\" } , " +

            "{ \"name\" : \"Sujita\" , " +
            "\"designation\" : \"CEO\" , " +
            "\"location\" : \"Butwal\" } , " +

            "{ \"name\" : \"Maila bhai\" , " +
            "\"designation\" : \"Programmer\" , " +
            "\"location\" : \"Gorkha\" } " +

            " ] }";
    }

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

        // private string declared in the latter section of the program
```

```
String jsonStr = getListData();
try {
    // Create a userList string hashmap arraylist
    ArrayList<HashMap<String,String>> userList = new ArrayList<>();

    // Declaring the ListView from the layout file
    ListView lv = findViewById(R.id.user_list);

    // Initializing the JSON object and extracting the information
    JSONObject jsonObj = new JSONObject(jsonStr);
    JSONArray jsonArray = jsonObj.getJSONArray("users");
    for (int i = 0; i < jsonArray.length(); i++) {
        HashMap<String, String> user = new HashMap<>();
        JSONObject obj = jsonArray.getJSONObject(i);
        user.put("name", obj.getString("name"));
        user.put("designation", obj.getString("designation"));
        user.put("location", obj.getString("location"));
        userList.add(user);
    }

    // ListAdapter to broadcast the information to the list elements
    ListAdapter adapter = new SimpleAdapter(
        this, userList, R.layout.list_row,
        new String[]{"name", "designation", "location"},
        new int[]{R.id.name, R.id.designation, R.id.location}
    );
    lv.setAdapter(adapter);
} catch (JSONException ex) {
    Log.e("JsonParser Example", "unexpected JSON exception", ex);
}
}
```

List_row.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="5dip">
```

```

<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
>
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Name"
    android:textSize="30dp"
    android:textStyle="bold"
    android:layout_margin="10dp"
    android:padding="10dp"
    android:layout_column="1"/>
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Designation"
    android:textSize="30dp"
    android:textStyle="bold"
    android:layout_column="2"
    android:layout_margin="10dp"
    android:padding="10dp"/>
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Address"
    android:textSize="30dp"
    android:textStyle="bold"
    android:layout_column="3"
    android:layout_margin="10dp"
    android:padding="10dp"/>
</TableRow>
<TableRow
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
>
<!--TextView to display the name-->
<TextView
    android:id="@+id/name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="30dp"
    android:textStyle="bold"

```

```
android:layout_column="1"  
android:layout_margin="10dp"  
android:padding="10dp"/>
```

```
<!--TextView to display the designation-->
```

```
<TextView  
    android:id="@+id/designation"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:textSize="30dp"  
    android:layout_column="2"  
    android:layout_margin="10dp"  
    android:padding="10dp"/>
```

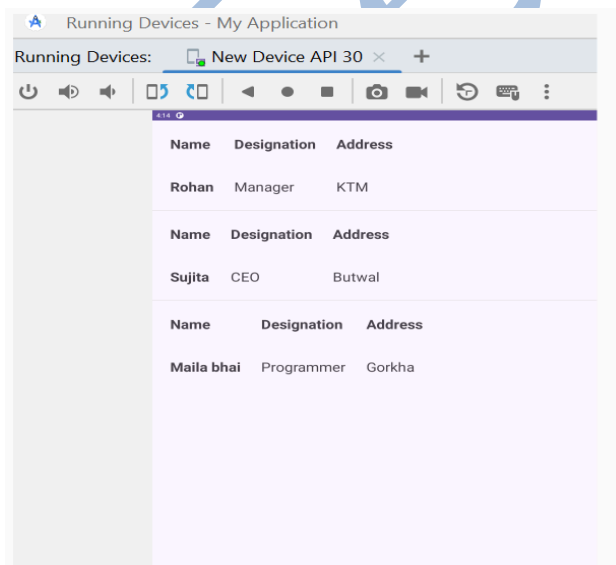
```
<!--TextView to display the location-->
```

```
<TextView  
    android:id="@+id/location"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_column="3"  
    android:textSize="30dp"  
    android:layout_margin="10dp"  
    android:padding="10dp"/>
```

```
</TableRow>
```

```
</TableLayout>
```

OutPut:



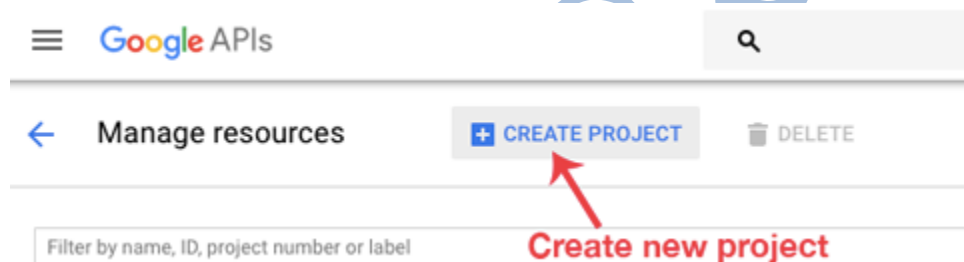
Implementing Google map in Application:

Android allows us to integrate Google map in your application. For this Google provides library via Google Play Service for using maps. To use Google Map we have to register our application on the Google Developer Console and get Google Map API.

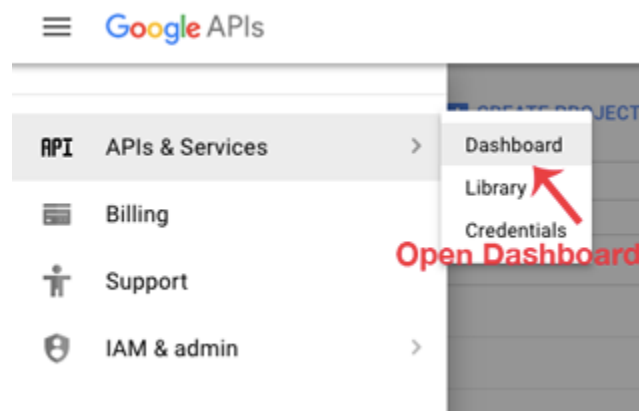
First step is to get Google Map API.

Step 1: Open Google developer console and Signin with your gmail account: <https://console.developers.google.com/project>

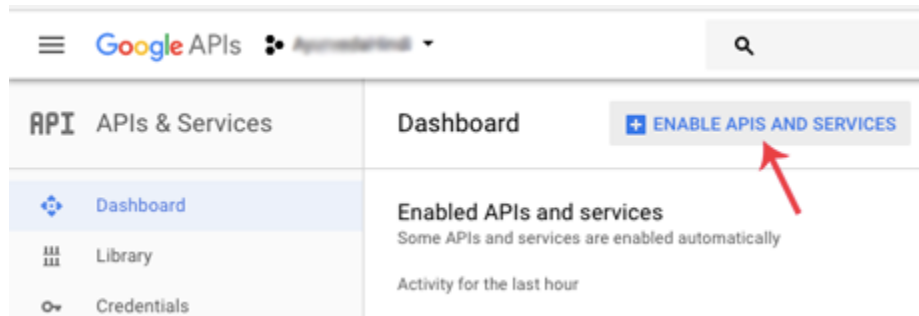
Step 2: Now create new project. You can create new project by clicking on the **Create Project** button and give name to your project.



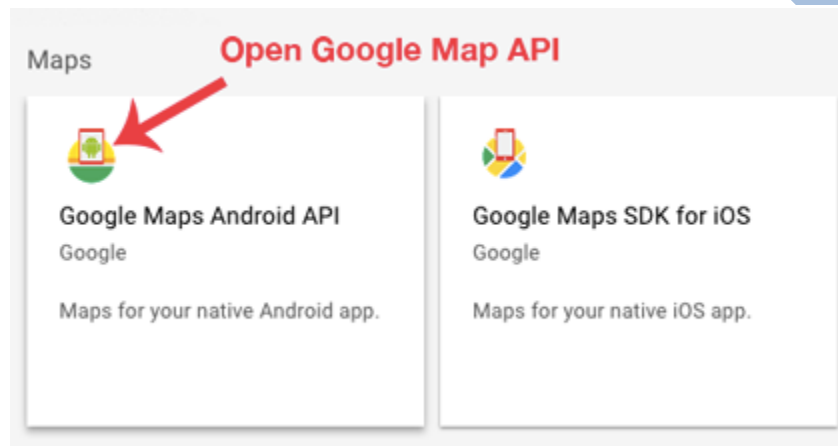
Step 3: Now click on APIs & Services and open Dashboard from it.



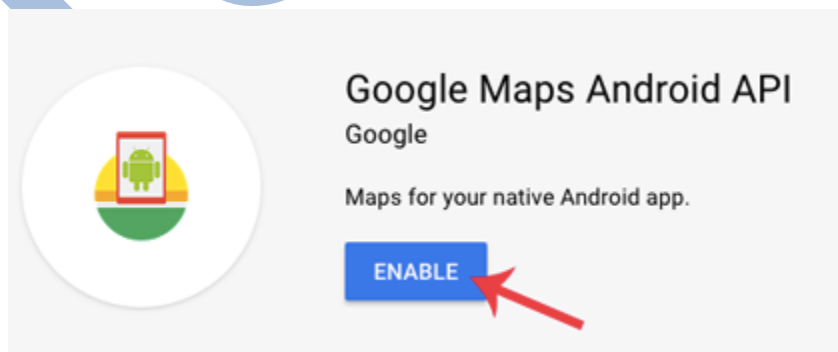
Step 4: In this open Enable APIS AND SERICES.



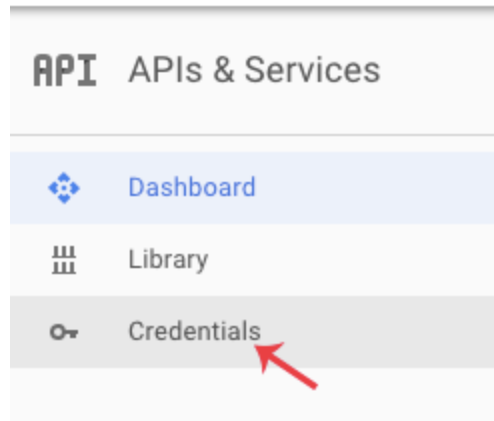
Step 5: Now open Google Map Android API.



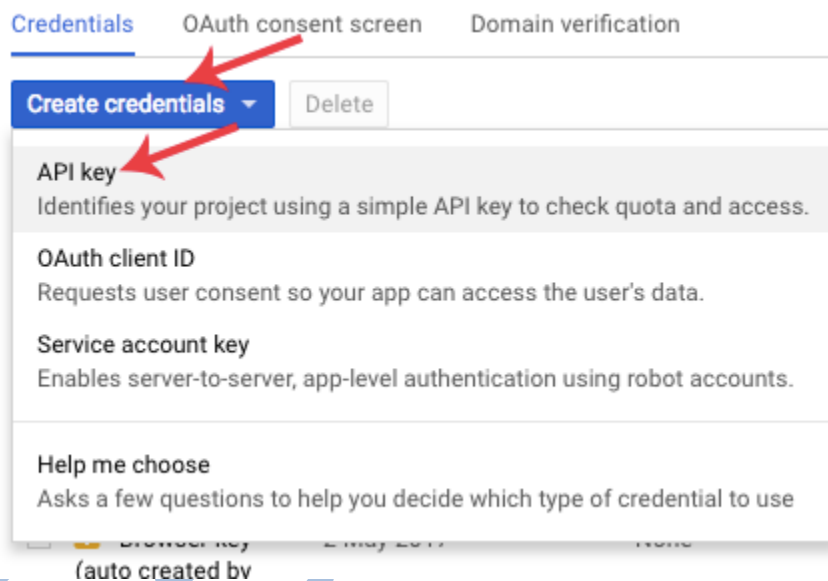
Step 6: Now enable the Google Maps Android API.



Step 6: Now go to Credentials



Step 7: Here click on Create credentials and choose API key



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.

API key created

Use this key in your application by passing it with the `key=API_KEY` parameter.

Your API key

AIzaSyCJg2QlAFtIWKH0g_bhJpdv13M4eH0u3Y



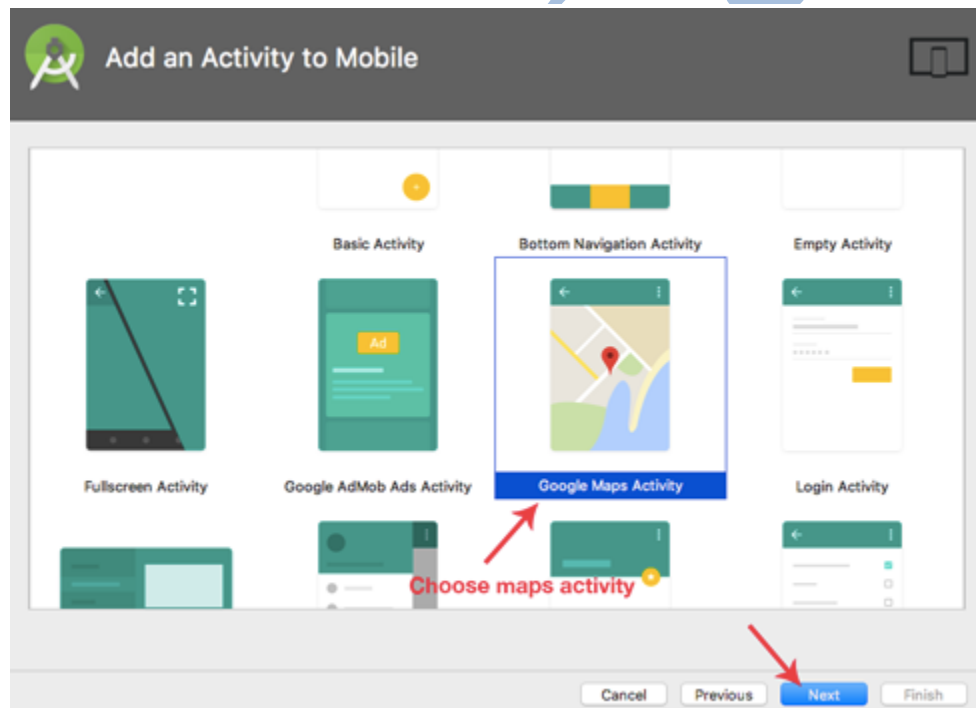
⚠ Restrict your key to prevent unauthorised use in production.

[CLOSE](#) [RESTRICT KEY](#)

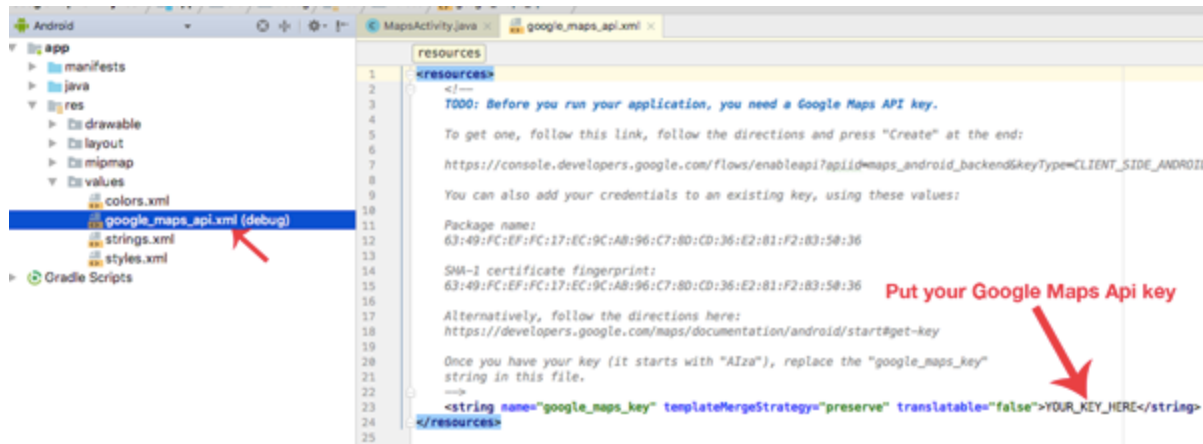
Now, Create an application

Step 1: Create a New Android Project and name it GoogleMaps.

Step 2: Now select Google Maps Activity and then click Next and finish.



Step 3: Now open `google_maps_api.xml` (debug) in values folder



Step 4: Here enter your Google Maps API key in place of YOUR_KEY_HERE.

```
<resources>

    <string name="google_maps_key" templateMergeStrategy="preserve"
translatable="false">AIzaSyDV2_xy58r15K6TskZy4KWMuhUDVq67jqM</string>

</resources>
```

Step 5: Now open build.gradle and add compile 'com.google.android.gms:play-services:8.4.0' in dependencies

build.gradle

```
apply plugin: 'com.android.application'

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])

    androidTestCompile('com.android.support.test.espresso:espresso-core:2.
2.2', {
        exclude group: 'com.android.support', module:
'support-annotations'
```

```
})  
compile 'com.google.android.gms:play-services:8.4.0'  
}
```

Step 6: Now create activity as well as xml file to implement it.

Assignment: Develop an android app to implement the Google map.

End of unit-7