UNIT 5

DATABASE

MSc CS

GAC-CBE

Contents

- Files and Database
- SQLite on Android
- Loading Asynchronous Data
- Map API

Files

All Android devices have two file storage areas: "internal" and "external" storage

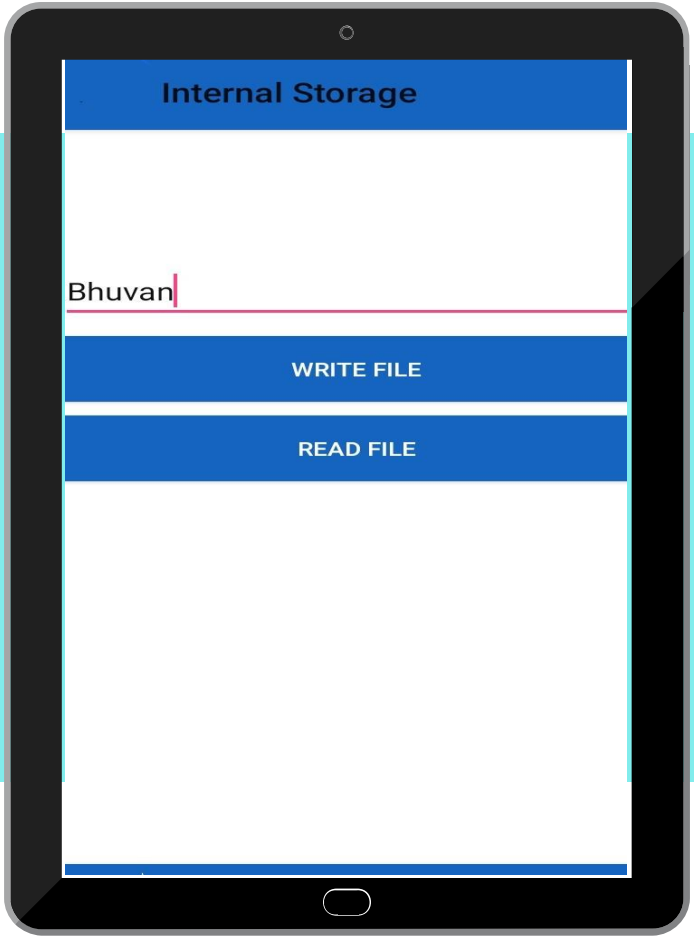
Internal VS External Storage

Internal storage

- Always available.
- Only user app can access files. Specifically, Other apps cannot browse user's internal directories and do not have read or write access unless user explicitly set the files to be readable or writable.
- When the user uninstalls the app, the system removes all the app's files from internal storage.
- Internal storage is best when neither the user nor other apps can access the files.

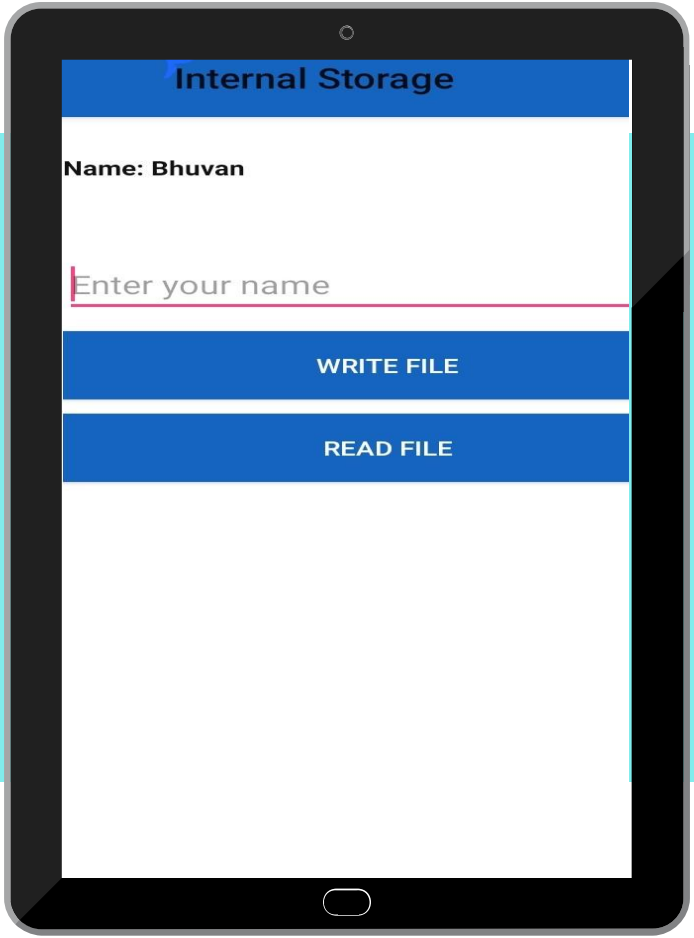
External storage

- Not always available, because the user can mount the external storage as USB storage and in some cases remove it from the device.
- World-readable. Any app can read.
- When the user uninstalls app, the system removes app's files from storage only if app is saved in the directory from getExternalFilesDir().
- External storage is the best place for files that don't require access restrictions and for files that want to share with other apps or allow the user to access with a computer.



WRITE FILE

When WRITE FILE Button is clicked it opens the file and writes Bhuvan into it and closes. Uses FileOutputStream.



READ FILE

When the Read FILE Button is clicked it opens and reads the Name Bhuvan that is stored in a file and closes the file. Uses FileInputStream

Databases

Store data in tables of rows and columns

SQL DATABASES

- The intersection of a row and column is called a *field*.
- Fields contain data, references to other fields, or references to other tables.
- Rows are identified by unique IDs.
- Columns are identified by names that are unique per table.

SQLite

characteristics:

- Self-contained (requires no other components)
- Serverless (requires no server backend)
- Zero-configuration (does not need to be configured for an app)
- Transactional (changes within a single transaction in SQLite either occur completely or not at all)
- most widely deployed database engine in the world. The source code in the public domain.

Student_table info

SQLite stores data in tables. Create a student_table with the following details.

- A database named **student.db**
- A table named **student_table**
- Columns for **ROLLNO**, **NAME**,**AGE** and **GENDER**

Query Language

A special SQL query language is used to interact with the database. Queries can be very complex, but there are four basic operations:

- Inserting rows
- Deleting rows
- Updating values in rows
- Retrieving rows that meet given criteria

Student_ table

ROLLNO	NAME	AGE	GENDER
1	Radhe	20	F
2	Varsha	21	F
3	Vini	21	F

SQLite in Android

A software library implemented an SQL database engine.

Constructor of SQLiteOpenHelper class

Constructor

```
public DatabaseHelper(Context
        context ,String name,
        SQLiteDatabase.CursorFactory
         factory, int version) {
    super(context,
            DATABASE_NAME, null, 1);
     this.getWritableDatabase();
  }
```

Description

creates an object for creating, opening and managing the database.

Methods of SQLiteOpenHelper class

Method	Description

`public void onCreate(SQLiteDatabase db) {`
`  db.execSQL("create table " +`
`  TABLE_NAME + "(ROLLNO`
`  INTEGER PRIMARY KEY`
`  AUTOINCREMENT ,NAME`
`  TEXT,AGE INTEGER,GENDER`
`  TEXT)");`
`}`
	called only once when database is created for the first time.

Methods of SQLiteOpenHelper class

Method	Description

public void onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) {
db.execSQL(**"DROP TABLE EXISTS"** + ***TABLE_NAME***);
onCreate(db); }
	called when database needs to be upgraded.

Methods of SQLiteDatabase class

Method

```
public boolean insertData(String name, String
                          age, String gender) {
SQLiteDatabase db =
                this.getWritableDatabase();
  ContentValues contentValues = new
                              ContentValues();
    contentValues.put(COL_2, name);
    contentValues.put(COL_3, age);
    contentValues.put(COL_4, gender);
    long result = db.insert(TABLE_NAME,
                null, contentValues);
    return result != -1;   }
```

Description

db.insert() 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.

Methods of SQLiteDatabase class

Method

```
public boolean updateData(String roll, String name, String
                                    age,  String gender) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put(COL_1, roll);
    contentValues.put(COL_2, name);
    contentValues.put(COL_3, age);
    contentValues.put(COL_4, gender);
    db.update(TABLE_NAME, contentValues, "ROLLNO =
                ?", new  String[]{roll});
    return true;  }
```

Description

db.update() updates the row based on the Rollno received and other details.

Methods of SQLiteDatabase class

Method	Description

`public Integer deleteData(String roll){`
`  SQLiteDatabase db =`
`  this.getWritableDatabase();`
`  return`
`  db.delete(TABLE_NAME,"ROLLNO =`
`  ?",new String[]{roll});  }`
	db.delete() deletes a row based on the ROLLNO received as input.

Methods of SQLiteDatabase class

Method

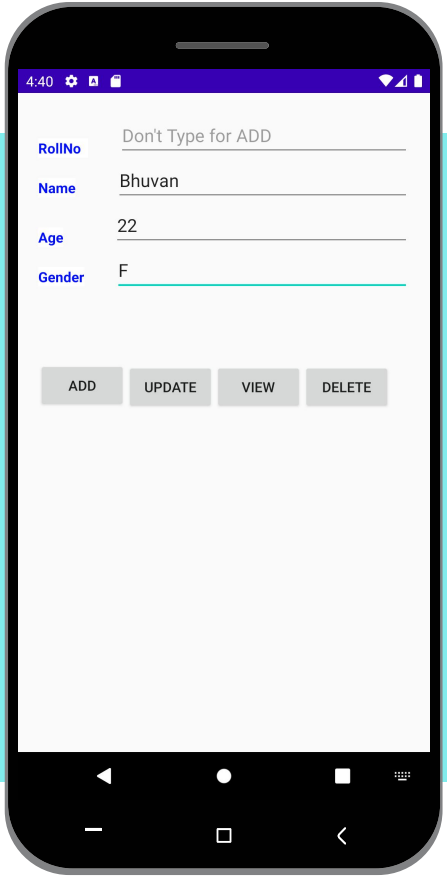
```
public Cursor getAllData() {
    SQLiteDatabase db =
this.getWritableDatabase();
      return db.rawQuery("select *
  from " +    TABLE_NAME, null);
  }
```

Description

db.rawQuery() returns a cursor over the result set

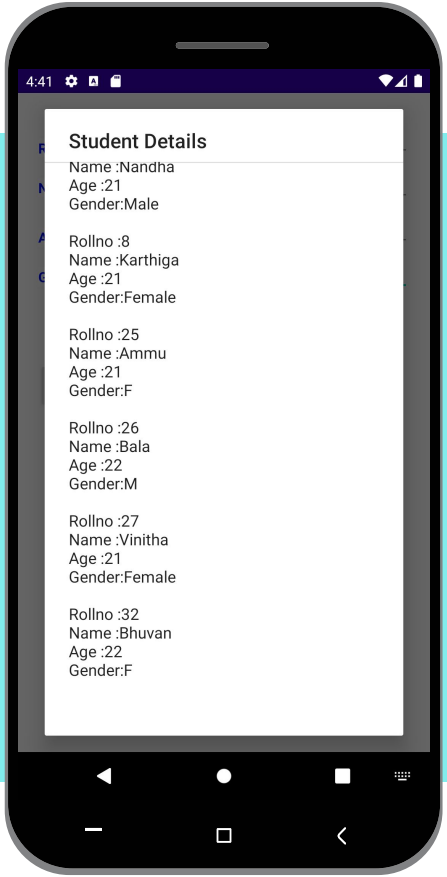
Entering Values to Insert

The User entered the Name, Age and Gender in the EditText. The RollNo need not be entered as it is auto generated.



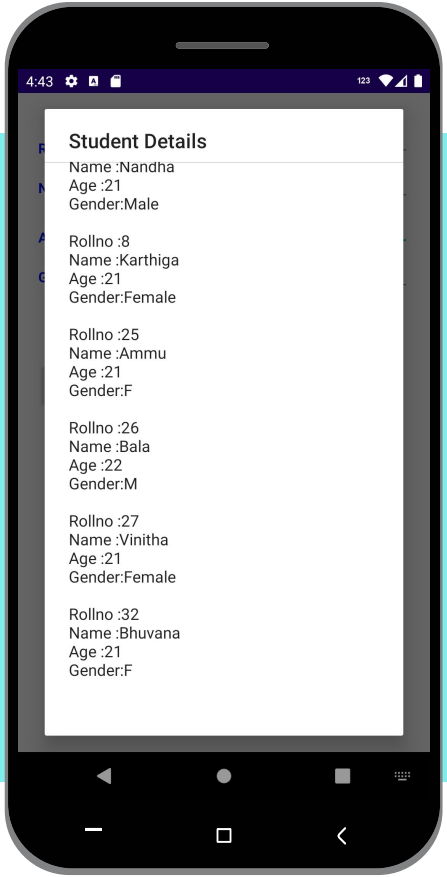
Values Displayed after Insertion

The Roll no 32 is autogenerated.The Name, Age and Gender are displayed as Bhuvan, 22, F respectively in an Alert Dialog.



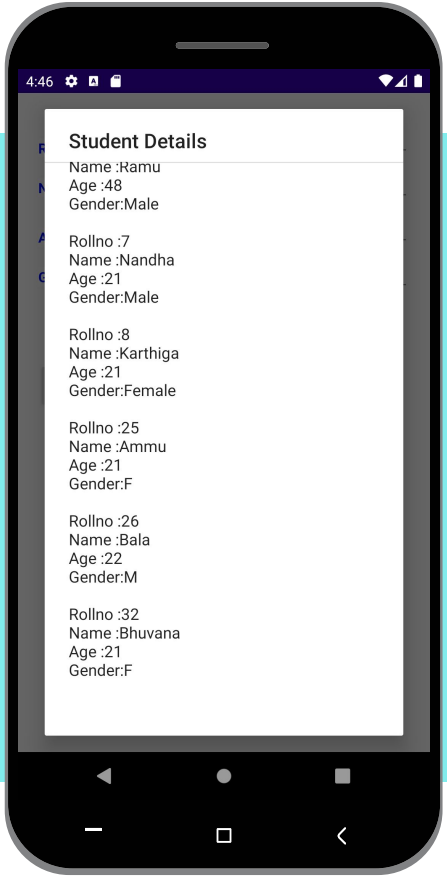
Updating Values

The Name and Age for the corresponding Rollno 32 is updated as Bhuvana and 21 respectively.



Deleting a Record

All the details related to Rollno 27 is deleted.



Loading Asynchronous Data

Loaders to request data from server asynchronously on background thread safely

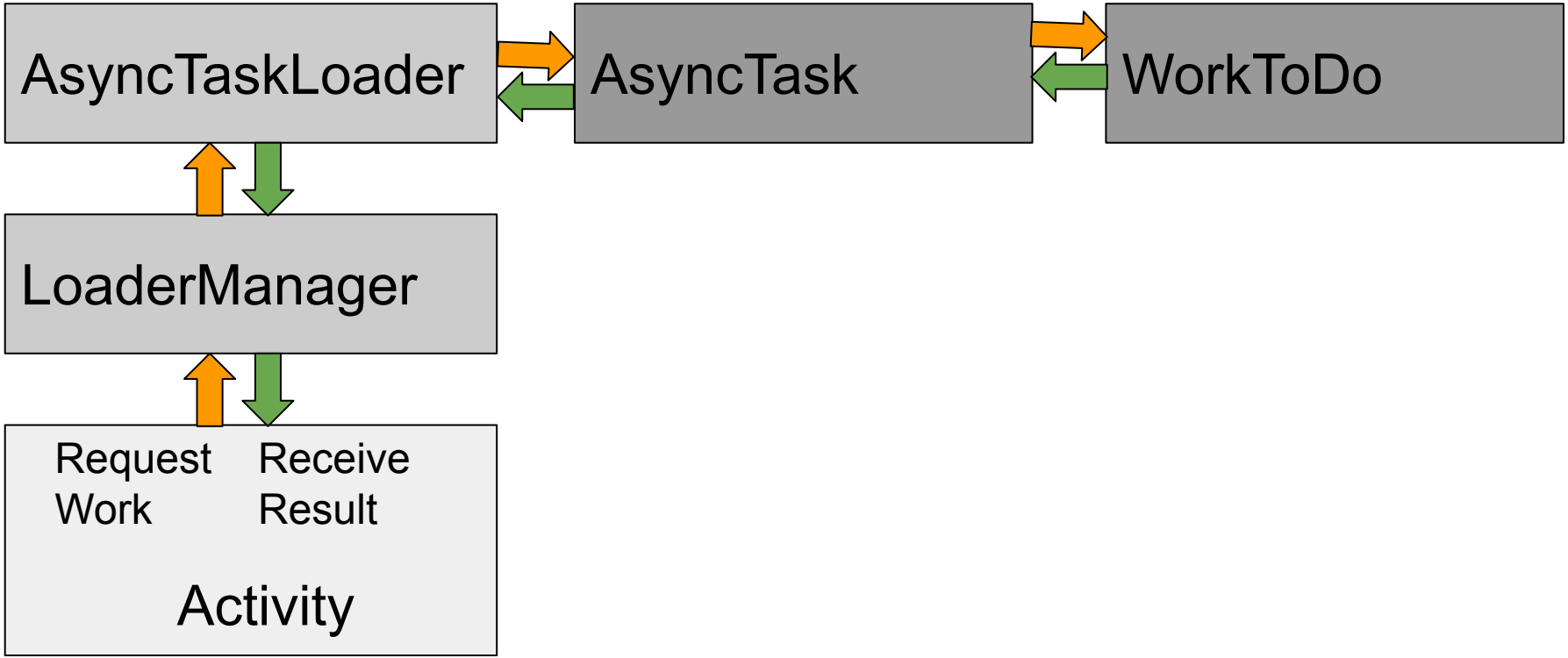
Loader

- Provides asynchronous loading of data
- Reconnects to Activity after configuration change
- Can monitor changes in data source and deliver new data
- Callbacks implemented in Activity
- Different types of loaders such as AsyncTaskLoader and CursorLoader are available.

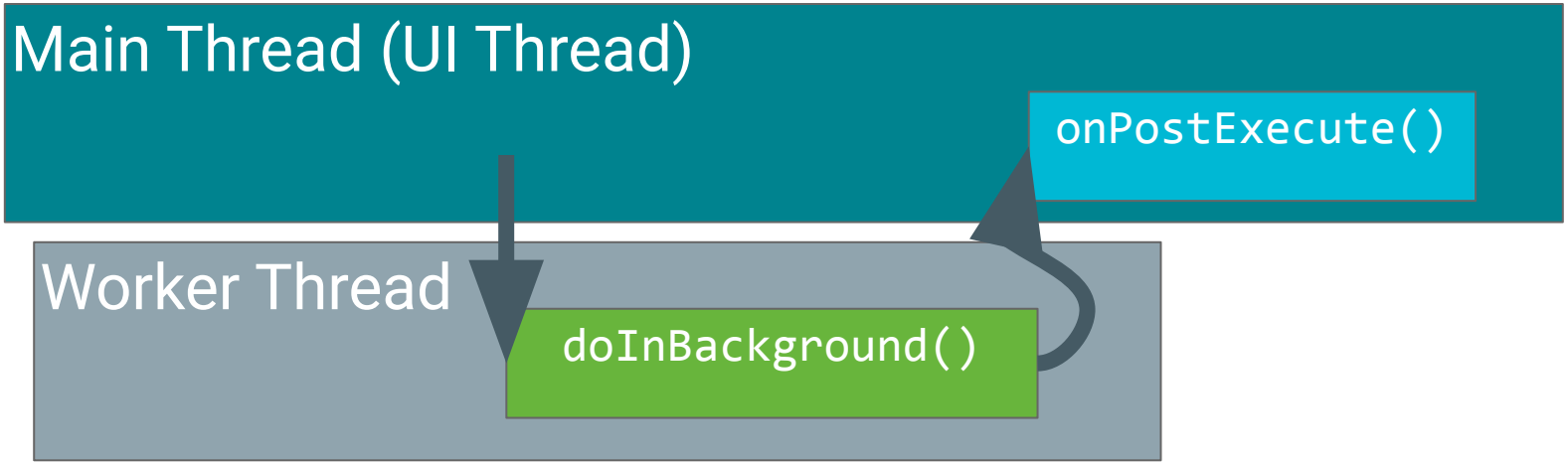
Need for Loaders

- Execute tasks OFF the UI thread
- LoaderManager handles configuration changes for user
- Efficiently implemented by the framework
- Users don't have to wait for data to load

AsyncTaskLoader



AsyncTask



AsyncTask is used to implement basic background tasks

Main Thread

- Independent path of execution in a running program
- Code is executed line by line
- App runs on Java thread called "main" or "UI thread"
- Draws UI on the screen
- Responds to user actions by handling UI events

Background Threads

Execute long running tasks such as Long calculations, Network operations,Downloading/uploading files,Processing images, Loading data on a background thread using

- AsyncTask
- The Loader Framework
- Services

AsyncTask Overriding Methods

- doInBackground() runs on a background thread when all the work is carried out in the background
- onPostExecute() runs on main thread when work is over. It process results and publishes results to the UI

LoaderManager

- Manages loader functions via callbacks
- Manages multiple loaders like Database data loader, AsyncTask data loader, Internet data etc.

AsyncTaskLoader Subclass

- Subclass AsyncTaskLoader
- Implement constructor
- loadInBackground()
- onStartLoading()

Subclass AsyncTaskLoader

```
public static class StringListLoader
        extends AsyncTaskLoader<List<String>> {

    public StringListLoader(Context context, String
    queryString) {
        super(context);
        mQueryString = queryString;
    }
}
```

loadInBackground()

```
public List<String> loadInBackground() {
    List<String> data = new ArrayList<String>;
    //TODO: Load the data from the network or from
a database
    return data;
}
```

onStartLoading()

When restartLoader() or initLoader() is called, the LoaderManager invokes the onStartLoading() callback

- Check for cached data
- Start observing the data source (if needed)
- Call forceLoad() to load the data if there are changes or no cached data

```
protected void onStartLoading() {  forceLoad();  }
```

Loader callbacks in Activity

- onCreateLoader() - Create and return a new Loader for the given ID
- onLoadFinished() - Called when a previously created loader has finished its load
- onLoaderReset() - Called when a previously created loader is being reset making its data unavailable

initLoader()

- Get a loader with initLoader() in Activity
- Use support library to be compatible with more devices

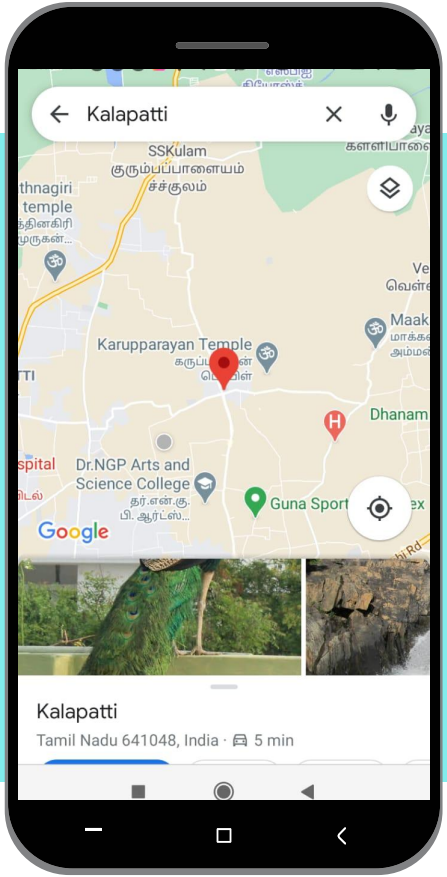
```
getSupportLoaderManager().initLoader(0, null, this);
```

Map API

Provides location intelligence for software developers creating location-based products and services.

Google Maps

Google Maps API allows user to include Google maps in app

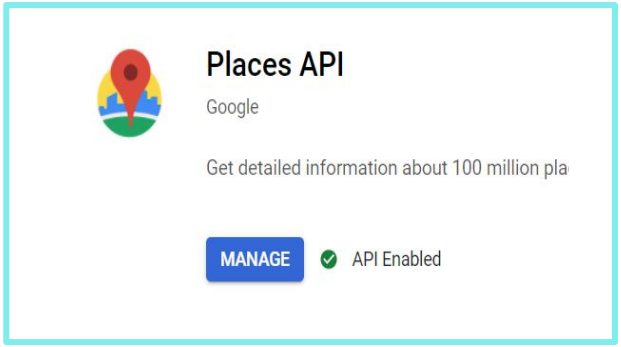


Steps to Add Map to an App

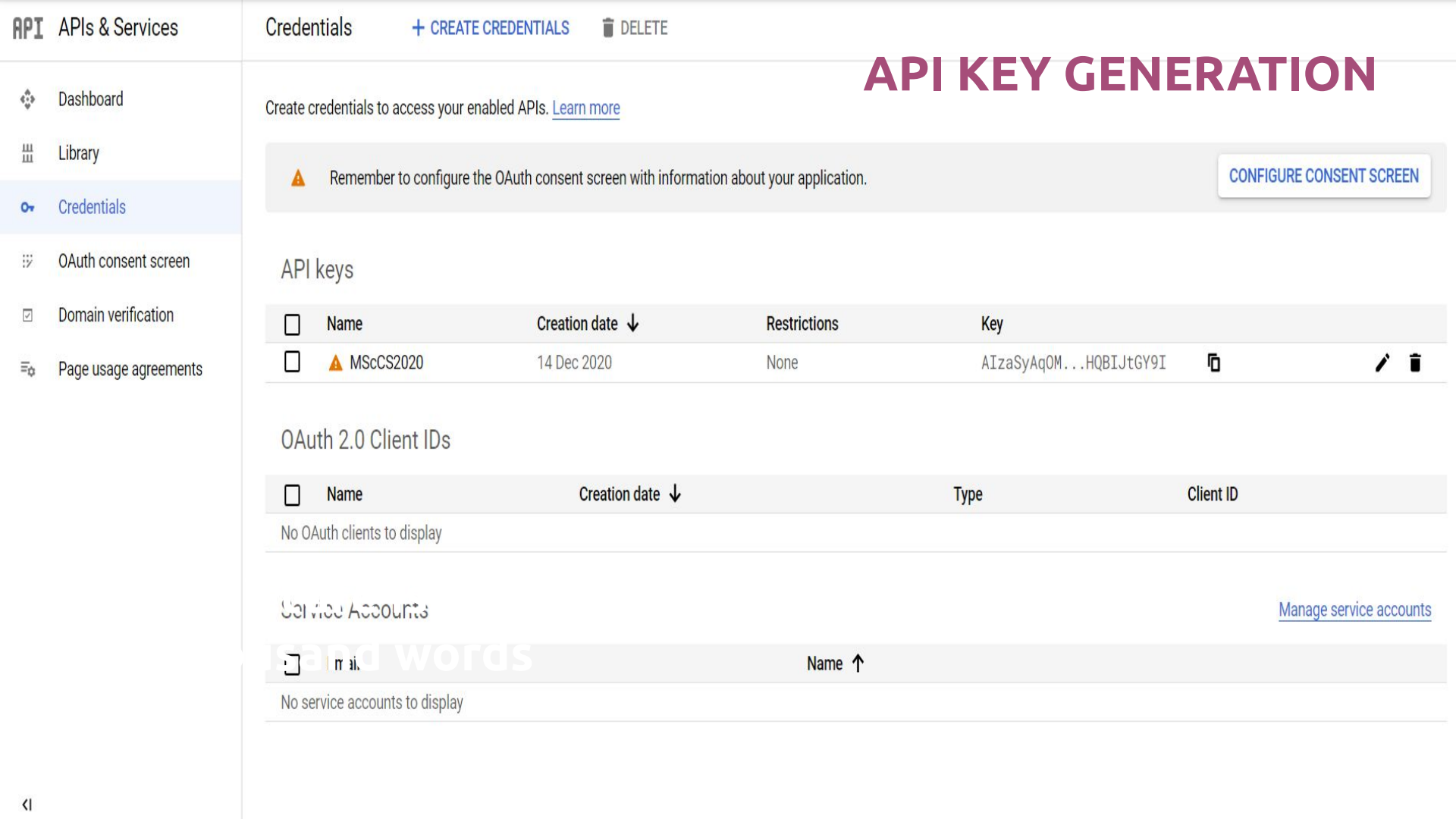
- Obtain API keys to use Google Map
- Include a Google Map in your app
- Change the look and feel of the map
- Change map behavior

Steps to Add API key to an App

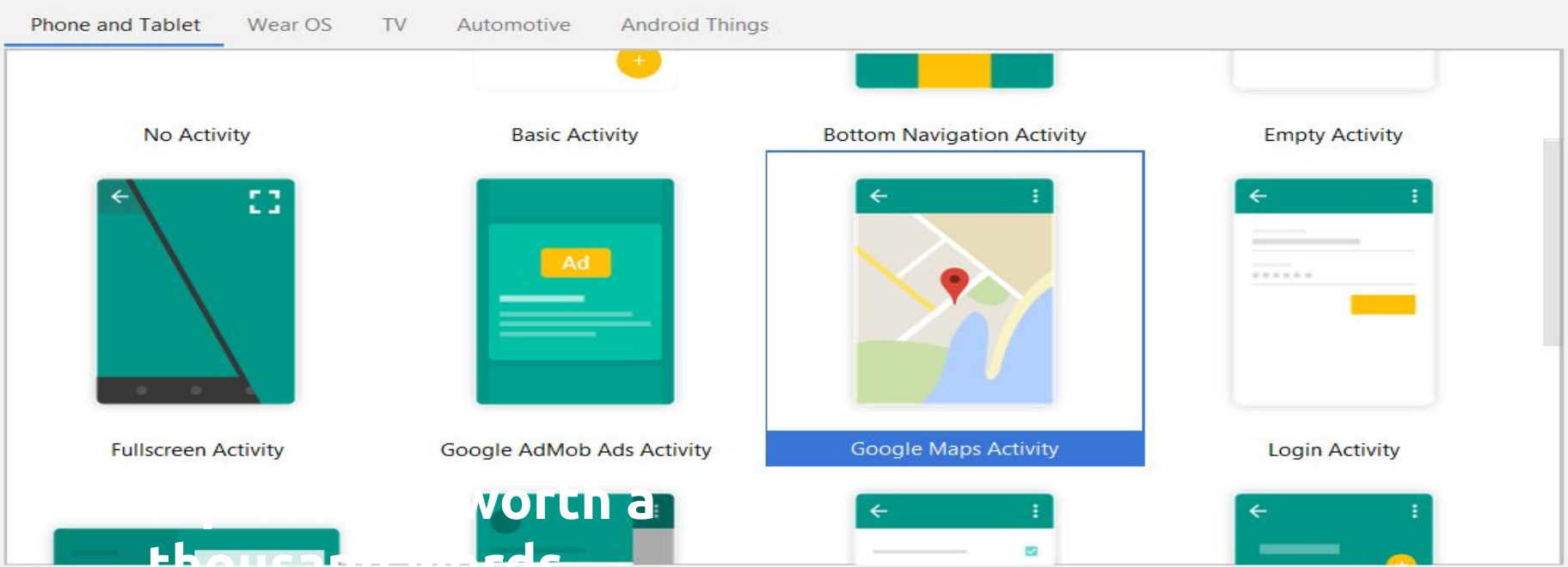
1. https://console.developers.google.com
2. Created new project named GAC-MSC-2020
3. Enabled APIs and Services
4. Created Credentials and API key is generated



API KEY GENERATION

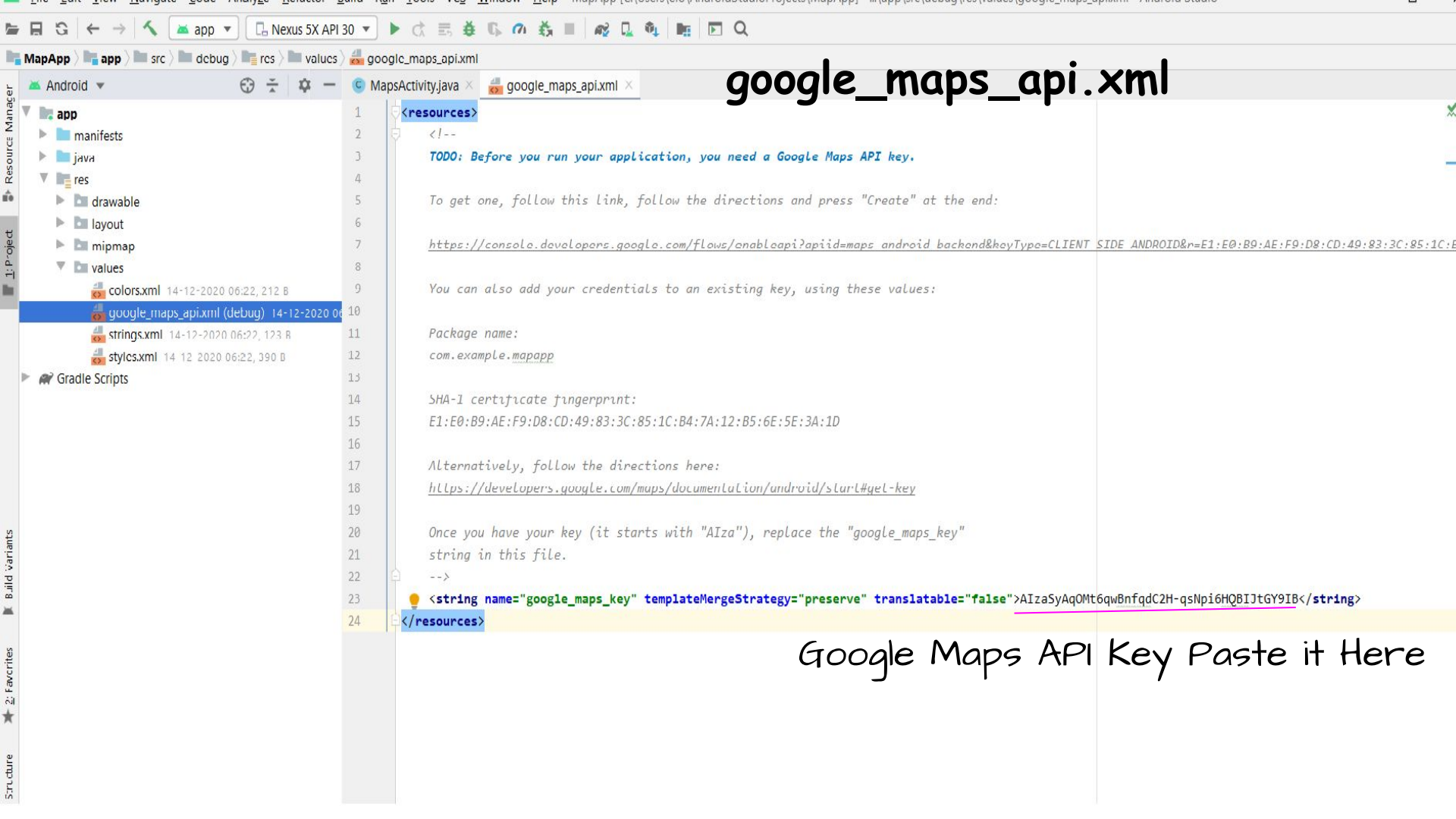


Google Maps Activity In Android Studio



worth a thousand words

google_maps_api.xml



Google Maps API Key Paste it Here

activity_maps.xml

```
<?xml version="1.0" encoding="utf-8"?>
<fragment
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:map="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:id="@+id/map"
  android:name="com.google.android.gms.maps.SupportMapFragment"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MapsActivity" />
```

MapsActivity.java

```
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {
   private GoogleMap mMap;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_maps);
       // 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); }
```

Activity containing the onMapReady() callback

onMapReady() in MapsActivity.java

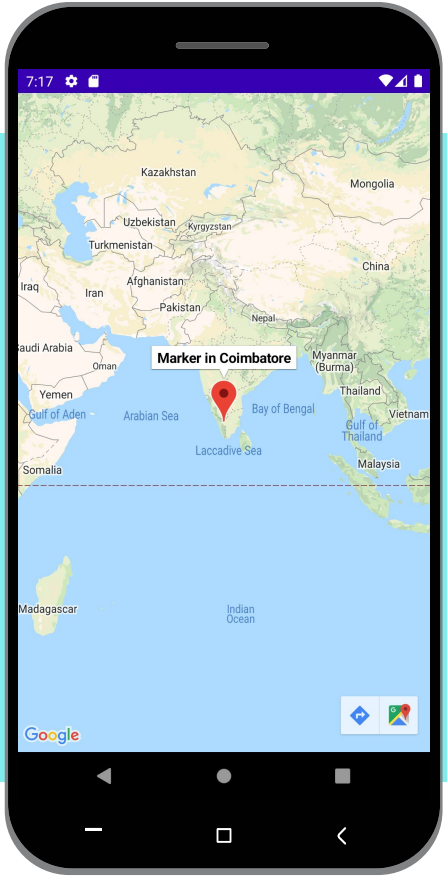
```
@Override
public void onMapReady(GoogleMap googleMap) {

    mMap = googleMap;

    // Add a marker in Coimbatore and move the camera
    LatLng coimbatore = new LatLng(11, 77);
    mMap.addMarker(new MarkerOptions().position(coimbatore).title("Marker
                    in Coimbatore"));
    mMap.moveCamera(CameraUpdateFactory.newLatLng(coimbatore));
}
```

Google Map displaying Coimbatore

OnMapReady(), the user sets the latitude as 11 and longitude as 77 for Coimbatore. A marker is added to the map with the title Marker in Coimbatore which will be displayed when clicked over the marker.



Thanks!

Does anyone have any queries?

Mail to
buvann@gmail.com

For More Info Visit
https://sudarmigumarivu.blogspot.com

Credits

- https://slidesgo.com/
- https://google-developer-training.github.io/
- https://developer.android.com