

Chapter 4

Advance App Development

4.1 SQLite Database

- SQLite is an open-source relational database i.e. used to perform database operations on android devices such as storing, manipulating or retrieving persistent data from the database.
- Android comes in with built in SQLite database implementation. SQLite is a lightweight and compact database that does not require any kind of server to run.
- It is embedded in android by default. So, there is no need to perform any database setup or administration task. In order to access this database, you don't need to establish any kind of connections for it like JDBC, ODBC, etc.
- SQLite databases are particularly useful for storing data locally on the device, caching data, and working with structured datasets in Android applications.
- Here are some key points about SQLite databases in Android:
 - **Local Data Storage:** SQLite databases are used in Android to store structured data on the device itself. This is particularly useful for scenarios where you need to store data that is specific to the app and doesn't need to be shared with other applications or synced with a remote server.
 - **Relational Database:** SQLite is a relational database management system, which means it supports tables with rows and columns and the ability to define relationships between tables. This makes it suitable for managing structured data.
 - **CRUD Operations:** SQLite supports standard database operations, often referred to as CRUD (Create, Read, Update, Delete) operations. You can create tables, insert data, query data, update data, and delete data in the database.
 - **Content Providers:** In Android, SQLite databases are commonly used with content providers, which allow data to be shared between different applications. Content providers provide a standard interface to access data stored in the SQLite database.
 - **OpenHelper Class:** To work with SQLite databases in Android, developers create a subclass of the SQLiteOpenHelper class. This class helps in managing database creation and version management.
 - **SQL Queries:** To interact with the database, developers use SQL (Structured Query Language) queries. These queries allow you to create, update, and retrieve data from the database.
 - **Security:** SQLite databases on Android are secure and private to the application that creates them. They are stored in the app's private directory and can't be accessed by other applications unless you explicitly share data through content providers or other mechanisms.
 - **Performance:** SQLite is known for its performance, especially when dealing with structured data. It's highly optimized and can efficiently handle database operations, even on resource-constrained mobile devices.

4.1.1 How Data is Being Stored in the SQLite Database?

- Data is stored in the SQLite database in the form of tables.
- When we stored this data in our SQLite database it is arranged in the form of tables that are similar to that of an excel sheet.
- Below is the representation of our SQLite database which we are storing in our SQLite database.

Data in our SQLite database is stored in the form of tables which is shown below.

This is the first column of our SQLite database which is of ID


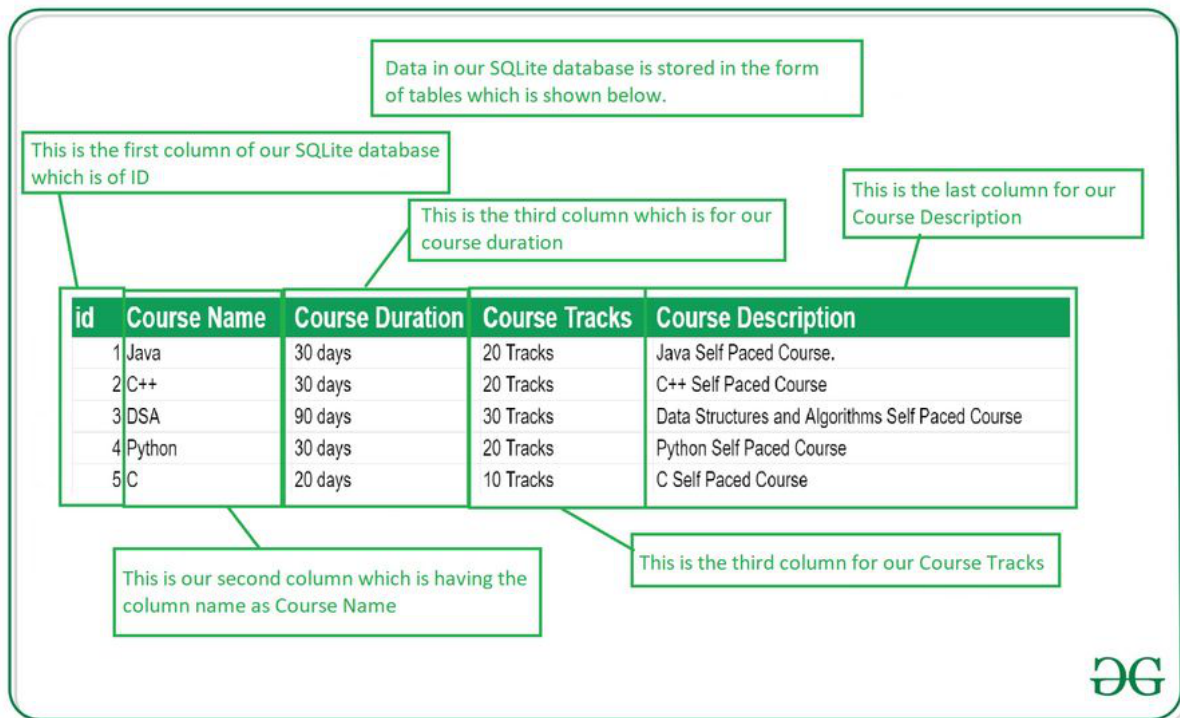
This is the third column which is for our course duration

This is the last column for our Course Description

id	Course Name	Course Duration	Course Tracks	Course Description
1	Java	30 days	20 Tracks	Java Self Paced Course.
2	C++	30 days	20 Tracks	C++ Self Paced Course
3	DSA	90 days	30 Tracks	Data Structures and Algorithms Self Paced Course
4	Python	30 days	20 Tracks	Python Self Paced Course
5	C	20 days	10 Tracks	C Self Paced Course

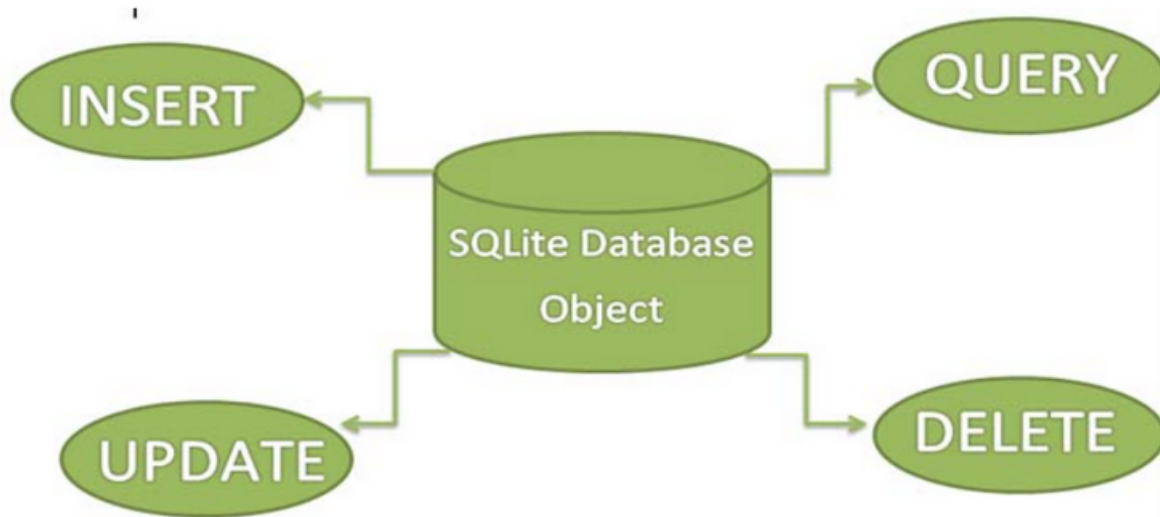
This is our second column which is having the column name as Course Name

This is the third column for our Course Tracks



4.1.2 SQLite architecture

- SQLite is, as previously mentioned, written in the C programming language while Android applications are primarily developed using Java. To bridge this “language gap”, the Android SDK includes a set of classes that provide a Java layer on top of the SQLite database management system. The remainder of this chapter will provide a basic overview of each of the major classes within this category.
- There are many libraries and classes available on Android to perform any kind of database queue on SQLite. It provides so many commands like add new data, update, read, and delete data.



4.1.3 Android SQLite Classes

4.1.3.1 SQLiteOpenHelper Class

- Android has features available to handle changing database schemas, which mostly depend on using the **SQLiteOpenHelper** class. **SQLiteOpenHelper** is designed to get rid of two very common problems.
 - When the application runs the first time - At this point, we do not yet have a database. So we will have to create the tables, indexes, starter data, and so on.
 - When the application is upgraded to a newer schema - Our database will still be on the old schema from the older edition of the app. We will have option to alter the database schema to match the needs of the rest of the app.
- For creating, updating and other operations you need to create a subclass or **SQLiteOpenHelper** class. **SQLiteOpenHelper** is a helper class to manage database creation and version management.
- The **SQLiteOpenHelper** is responsible for opening database if exist, creating database if it does not exists 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.
- This class provides two methods **onCreate(SQLiteDatabase db)**, **onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)**.
- **onCreate(SQLiteDatabase sqLiteDatabase)** method is called only once throughout the application life-cycle. 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(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.

Constructors of SQLiteOpenHelper Class:-

SQLiteOpenHelper class has two constructors.

1. **SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version):** This constructor creates an object for creating, opening, and managing the database.
2. **SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version, DatabaseErrorHandler errorHandler):** This constructor creates an object for creating, opening, and managing the database. It specifies the error handler.

Methods of SQLiteOpenHelper class:

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

1. **public abstract void onCreate(SQLiteDatabase db):** This method is called only when you create a database for the first time.
2. **public abstract void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion):** This method is called when the database needs to be upgraded.
3. **public synchronized void close():** This method closes the database object.

```
public class DbHandler extends SQLiteOpenHelper {
    private static final int DB_VERSION = 1;
    private static final String DB_NAME = "usersdb";
    private static final String TABLE_Users = "userdetails";
    private static final String KEY_ID = "id";
    private static final String KEY_NAME = "name";
    private static final String KEY_LOC = "location";
    private static final String KEY_DESG = "designation";
    public DbHandler(Context context){
        super(context,DB_NAME, null, DB_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db){
        String CREATE_TABLE = "CREATE TABLE " + TABLE_Users + "("
            + KEY_ID + " INTEGER PRIMARY KEY AUTOINCREMENT," + KEY_NAME + " TEXT,"
            + KEY_LOC + " TEXT,"
            + KEY_DESG + " TEXT"+ ")";
        db.execSQL(CREATE_TABLE);
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion){
        // Drop older table if exist
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_Users);
        // Create tables again
        onCreate(db);
    }
}
```

4.1.3.2 SQLiteDatabase Class

- The SQLiteDatabase class in Android is a fundamental part of working with SQLite databases. It provides methods and functionality for managing and interacting with an SQLite database.
- SQLiteDatabase provides methods for various database operations, including creating tables, inserting, updating, deleting, and querying data. Some key methods of this class are as follows:
 - **insert()** – Inserts a new row into a database table.
 - **delete()** – Deletes rows from a database table.
 - **query()** – Performs a specified database query and returns matching results via a Cursor object.
 - **execSQL()** – Executes a single SQL statement that does not return result data.
 - **rawQuery()** – Executes a SQL query statement and returns matching results in the form of a Cursor object.
- You should open the database using the **getReadableDatabase()** or **getWritableDatabase()** methods provided by your database helper. It's essential to close the database when you're done with it to release system resources. You can use the **close()** method for this purpose.

```
// Create or open a database
SQLiteDatabase db = dbHelper.getWritableDatabase();

// Insert data into a table
ContentValues values = new ContentValues();
values.put("name", "John");
values.put("age", 30);
long newRowId = db.insert("my_table", null, values);

// Query data
Cursor cursor = db.query(
    "my_table",
    new String[] { "name", "age" },
    null,
    null,
    null,
    null,
    null
);


// Close the database when you're done
db.close();
```

- In this example, we create or open a database, insert data, and query data from a table using the SQLiteDatabase class.

4.1.3.3 Cursor Class

- The Cursor class in Android is an integral part of working with SQLite databases.
 - A class provided specifically to provide access to the results of a database query. It provides a way to retrieve and manipulate data from a database, especially when you want to query data from tables.
 - A Cursor is used to represent the result set of a database query. It allows you to navigate through the rows and columns of the data retrieved from a database table.
 - You use a Cursor object to execute SQL queries that return data, like SELECT statements. The result of the query is stored in the Cursor, and you can move through the results row by row.
 - For example, a SQL SELECT operation performed on a database will potentially return multiple matching rows from the database. A Cursor instance can be used to step through these results, which may then be accessed from within the application code using a variety of methods.
 - Some key methods of this class are as follows:
- **close()** – Releases all resources used by the cursor and closes it.
 - **getCount()** – Returns the number of rows contained within the result set.
 - **moveToFirst()** – Moves to the first row within the result set.
 - **moveToLast()** – Moves to the last row in the result set.
 - **moveToNext()** – Moves to the next row in the result set.
 - **move()** – Moves by a specified offset from the current position in the result set.
 - **get<type>()** – Returns the value of the specified <type> contained at the specified column index of the row at the current cursor position (variations consist of *getString()*, *getInt()*, *getShort()*, *getFloat()* and *getDouble()*).

java

 Copy code

```
SQLiteDatabase db = dbHelper.getReadableDatabase();
Cursor cursor = db.query(
    "my_table",
    new String[] { "name", "age" },
    null,
    null,
    null,
    null,
    null
);

if (cursor != null) {
    if (cursor.moveToFirst()) {
        do {
            String name = cursor.getString(cursor.getColumnIndex("name"));
            int age = cursor.getInt(cursor.getColumnIndex("age"));
            // Process data
        } while (cursor.moveToNext());
    }
    cursor.close();
}
```

- In this example, we execute a query to retrieve "name" and "age" columns from a table. We use the Cursor to move through the result set and retrieve data from each row.

4.2 Sensors

- Most Android-powered devices have built-in sensors that measure motion, orientation, and various environmental conditions.
- Sensors are physical components built into a handset or tablet device. They derive their data by directly measuring specific environmental properties, such as acceleration, geomagnetic field strength, or angular change.
- These sensors are capable of providing raw data with high precision and accuracy, and are useful if you want to monitor three-dimensional device movement or positioning, or you want to monitor changes in the ambient environment near a device.
- For example, a game might track readings from a device's gravity sensor to infer complex user gestures and motions, such as tilt, shake, rotation, or swing.
- Likewise, a weather application might use a device's temperature sensor and humidity sensor to calculate and report the dewpoint, or a travel application might use the geomagnetic field sensor and accelerometer to report a compass bearing.
- The Android platform supports three broad categories of sensors:
 - **Motion sensors:** These sensors measure acceleration forces and rotational forces along three axes. This category includes accelerometers, gravity sensors, gyroscopes, and rotational vector sensors.
 - **Environmental sensors:** These sensors measure various environmental parameters, such as ambient air temperature and pressure, illumination, and humidity. This category includes barometers, photometers, and thermometers.
 - **Position sensors:** These sensors measure the physical position of a device. This category includes orientation sensors and magnetometers.
- Sensor types supported by the Android platform.

Sensor	Type	Description	Common Uses
TYPE_ACCELEROMETER	Hardware	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), including the force of gravity.	Motion detection (shake, tilt, etc.).
TYPE_AMBIENT_TEMPERATURE	Hardware	Measures the ambient room temperature in degrees Celsius ($^{\circ}\text{C}$). See note below.	Monitoring air temperatures.
TYPE_GRAVITY	Software or Hardware	Measures the force of gravity in m/s^2 that is applied to a device on all three physical axes (x, y, z).	Motion detection (shake, tilt, etc.).
TYPE_GYROSCOPE	Hardware	Measures a device's rate of rotation in rad/s around each of the three physical axes (x, y, and z).	Rotation detection (spin, turn, etc.).
TYPE_LIGHT	Hardware	Measures the ambient light level (illumination) in lx.	Controlling screen brightness.
TYPE_LINEAR_ACCELERATION	Software or Hardware	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), excluding the force of gravity.	Monitoring acceleration along a single axis.
TYPE_MAGNETIC_FIELD	Hardware	Measures the ambient geomagnetic field for all three physical axes (x, y, z) in μT .	Creating a compass.
TYPE_ORIENTATION	Software	Measures degrees of rotation that a device makes around all three physical axes (x, y, z). As of API level 3 you can obtain the inclination matrix and rotation matrix for a device by using the gravity sensor and the geomagnetic field sensor in conjunction with the getRotationMatrix() method.	Determining device position.
TYPE_PRESSURE	Hardware	Measures the ambient air pressure in hPa or mbar.	Monitoring air pressure changes.
TYPE_PROXIMITY	Hardware	Measures the proximity of an object in cm relative to the view screen of a device. This sensor is typically used to determine whether a handset is being held up to a person's ear.	Phone position during a call.
TYPE_RELATIVE_HUMIDITY	Hardware	Measures the relative ambient humidity in percent (%).	Monitoring dewpoint, absolute, and relative humidity.
TYPE_ROTATION_VECTOR	Software or Hardware	Measures the orientation of a device by providing the three elements of the device's rotation vector.	Motion detection and rotation detection.
TYPE_TEMPERATURE	Hardware	Measures the temperature of the device in degrees Celsius ($^{\circ}\text{C}$). This sensor implementation varies across devices and this sensor was replaced with the TYPE_AMBIENT_TEMPERATURE sensor in API Level 14	Monitoring temperatures.

4.2.1 Sensor Framework

You can access these sensors and acquire raw sensor data by using the Android sensor framework. The sensor framework is part of the `android.hardware` package and includes the following classes and interfaces:

4.2.1.1 SensorManager

You can use this class to create an instance of the sensor service. This class provides various methods for accessing and listing sensors, registering and unregistering sensor event listeners, and acquiring orientation information. This class also provides several sensor constants that are used to report sensor accuracy, set data acquisition rates, and calibrate sensors.

4.2.1.2 Sensor

You can use this class to create an instance of a specific sensor. This class provides various methods that let you determine a sensor's capabilities.

4.2.1.3 SensorEvent

The system uses this class to create a sensor event object, which provides information about a sensor event. A sensor event object includes the following information: the raw sensor data, the type of sensor that generated the event, the accuracy of the data, and the timestamp for the event.

4.2.1.4 SensorEventListener

You can use this interface to create two callback methods that receive notifications (sensor events) when sensor values change or when sensor accuracy changes.

4.3 Bluetooth

- Bluetooth is a wireless communication technology that enables data exchange over short distances between Android devices.
- In Android, Bluetooth is implemented through the Android Bluetooth API, which allows developers to create applications that can discover, pair with, and communicate with other Bluetooth devices, such as smartphones, tablets, headphones, speakers, and more.
- **Bluetooth Stack:** Android devices use the Bluetooth stack to enable Bluetooth communication. Android supports multiple Bluetooth profiles for various use cases, including A2DP (audio streaming), SPP (serial port profile), and many others.
- **Bluetooth Permissions:** To use Bluetooth in your Android app, you need to declare the necessary permissions in your **AndroidManifest.xml** file, such as **BLUETOOTH**, **BLUETOOTH_ADMIN**, and **BLUETOOTH_CONNECT** for basic Bluetooth operations.
- **Enabling Bluetooth:** You can programmatically enable or disable Bluetooth on the device using the **BluetoothAdapter** class. Users will also be prompted for consent when enabling Bluetooth via your app.
- **Device Discovery:** The Android Bluetooth API allows you to discover nearby Bluetooth devices. You can search for nearby devices and retrieve a list of available devices, including their names and addresses.
- **Pairing and Bonding:** Pairing is the process of establishing a connection between two Bluetooth devices. Bonding is an optional, more secure process that involves saving the pairing information for future connections. Android handles pairing and bonding processes automatically when connecting to devices.
- **Device Connections:** You can connect to a remote Bluetooth device using its MAC address and the **BluetoothSocket** class. Once a connection is established, you can send and receive data over the Bluetooth channel.
- **Bluetooth Profiles:** Android supports various Bluetooth profiles, which define how different types of devices can communicate with one another. Examples include:
 - A2DP (Advanced Audio Distribution Profile): For streaming audio to headphones or speakers.
 - HFP (Hands-Free Profile): For Bluetooth headsets and car kits.
 - SPP (Serial Port Profile): For serial communication, often used for IoT devices.
 - GATT (Generic Attribute Profile): For low-energy Bluetooth communication, used by Bluetooth Low Energy (BLE) devices.
- **Bluetooth Low Energy (BLE):** Android also supports Bluetooth Low Energy (BLE), which is a power-efficient version of Bluetooth primarily used for IoT devices, beacons, and wearable technology. The Android BLE API allows your app to communicate with BLE devices.
- **Bluetooth Permissions and Security:** Bluetooth communications should be secure. Android provides the means to secure Bluetooth communications, including encryption and authentication. Be aware of security best practices when using Bluetooth in your app.
- **Bluetooth in App Development:** You can develop various types of applications using Bluetooth, such as:
 - Bluetooth file transfer apps for sharing files between devices.
 - Bluetooth chat applications for text messaging.
 - Bluetooth control apps for remotely controlling other devices.
 - Bluetooth-based IoT applications for controlling smart devices.
 - Bluetooth audio streaming applications for wireless music playback.
- **Compatibility and Version:** Different Android devices may have different Bluetooth versions and capabilities. Ensure that your app accounts for variations in Bluetooth support.

4.3.1 BluetoothAdapter class

- The BluetoothAdapter class in Android is a key component of the Android Bluetooth API. It represents the device's own Bluetooth adapter and provides methods for various Bluetooth-related operations.
- This class allows you to control the device's Bluetooth functionality, including enabling or disabling Bluetooth, discovering nearby devices, and establishing connections.

Methods of BluetoothAdapter class

Commonly used methods of BluetoothAdapter class are as follows:

- **static synchronized BluetoothAdapter getDefaultAdapter()** returns the instance of BluetoothAdapter.
- **boolean enable()** enables the bluetooth adapter if it is disabled.
- **boolean isEnabled()** returns true if the bluetooth adapter is enabled.
- **boolean disable()** disables the bluetooth adapter if it is enabled.
- **String getName()** returns the name of the bluetooth adapter.
- **boolean setName(String name)** changes the bluetooth name.
- **int getState()** returns the current state of the local bluetooth adapter.
- **Set<BluetoothDevice> getBondedDevices()** returns a set of paired (bonded) BluetoothDevice objects.
- **boolean startDiscovery()** starts the discovery process.

4.4 Geo Location - Location Based Services

- You must have used apps like Google Maps, Waze, MapQuest, etc. These are the applications that help you track the location of the device. They also provide services like finding nearby restaurants, hospitals, petrol pumps, etc. Even the cab drivers now use maps to locate their route.
- Location-based services (LBS) in Android app development involve using the device's location (latitude and longitude coordinates) to provide users with location-specific information, features, or services.
- These services leverage various technologies, including GPS (Global Positioning System), network-based positioning, and sensors, to determine a device's geographic location accurately. Location-based services are widely used in applications such as maps and navigation, social networking, weather forecasting, geotagging, location-based advertising, and more.
- Location-Based Services(LBS) are present in Android to provide you with features like current location detection, display of nearby places, geofencing, etc. It fetches the location using your device's GPS, Wifi, or Cellular Networks.
- To build an app with location-based services, you need to access the Google Play Services Module. After that, you need to use a framework called Location Framework, which has many methods, classes, and interfaces to make your task easier.
- **Location Providers:** Android provides two primary location providers to determine a device's location:
 - **GPS Provider:** This provider uses signals from multiple satellites to calculate the device's precise location. It's accurate but may require an unobstructed view of the sky.
 - **Network Provider:** This provider uses cell tower and Wi-Fi signals to estimate the device's location. It works indoors and is quicker to provide a location fix but is less accurate than GPS.
- To access the device's location, your app needs the **ACCESS_FINE_LOCATION** or **ACCESS_COARSE_LOCATION** permission, depending on the desired level of accuracy. You must request these permissions in your app's manifest file and handle them at runtime on Android 6.0 (API level 23) and later.

4.4.1 Components of Location-Based Services in Android

The classes and the interfaces present in the Location Framework acts as the essential components for LBS. Those components are as follows:

- **LocationManager** Class – It is used to get Location Service access from the system.
- **LocationListener** Interface – It receives updates from the Location Manager class.
- **LocationProvider** – It is the class that provides us with the location for our devices.
- **Location** Class – Its objects carry information about the location. The information includes latitude, longitude, accuracy, altitude, and speed.

4.4.2 Location Object

The location objects carry the location of your device. The place is in the form of latitude and longitude.

On the location object, you can apply the below methods. The below methods help you to get location and other information regarding the location.

- **float distanceTo(Location destination):** It gives the approximate distance between our current location and the destination location.
- **float getAccuracy():** It gives us the accuracy of our location in metres.
- **double getAltitude():** It gives us the altitude of our place above sea level.
- **double getLatitude():** It gives the latitude coordinate of our place in degrees.
- **double getLongitude():** It gives the latitude coordinate of our place in degrees.

- **float getSpeed():** It gives the speed of our location change.
- **void setAccuracy(float accuracy):** Using setAccuracy(), you can set your custom accuracy in metres.
- **void setAltitude(double altitude):** Using setAltitude(), you can set the altitude of your place from sea level in metres.
- **void setBearing(float bearing):** Using the setBearing() method, you can set location bearing in degrees.
- **void setLongitude(double longitude):** You can even set your location to some other longitude using the setLongitude() method.
- **void setSpeed(float speed):** You can even set speed using the setSpeed() method.
- **void reset():** It is used to reset your set location.
- **boolean hasAccuracy():** It says whether or not the location is accurate.
- **boolean hasAltitude():** It says if the place has an altitude or not
- **boolean hasSpeed():** It is true if the place has a speed attached.
- **boolean hasBearing():** It returns true if the place has a bearing or not.

4.5 SMS and MMS

- We always send messages to others. In older days people have only one means of communication i.e, letters but nowadays there are many ways to send messages like email, SMS, MMS, and messages on the OTT (over-the-top) application like WhatsApp, telegram, and signal.
- The two main ways to send messages on the mobile phone using a cellular network are MMS and SMS. All the mobile operating systems support the MMS and SMS through their default messaging application.
- We need not download some additional applications for sending them. All devices do not support MMS it completely depends on the operator that is providing us the cellular network.
- Both the SMS and MMS fall into the same category of Text Messaging but there are a lot of differences between them.
- **SMS** stands for **Short Messaging Service**.
 - It was invented in 1980. It is the oldest and most widely used form of text messaging on mobile devices.
 - This form of messaging is fully operational and it is supported by all the mobile devices, and we need not download some other application to use it. We do not need to have a smartphone to use it.
 - There are some problems with the SMS like it allows message length up to 160 characters. If the length was above it then the message will be split into multiple messages.
 - Another problem with SMS is that it does not support multimedia.
- **MMS** stands for **Multimedia Messaging Service**.
 - MMS is just an extension of the features of the SMS. It is also sent over cellular networks.
 - The MMS is more appealing in comparison to SMS because of the images and videos we can use in them. It allows users to send Multimedia like images, GIF, videos with text which was not supported in the SMS.
 - The maximum length supported by it depends on the cellular operator not generally speaking it is 10 times more than SMS. The typical length is 1600.
 - It is not supported by every mobile device. It is more suitable for smartphones and not for analog cellphones. Sending MMS is more costly in comparison to sending SMS.

	SMS	MMS
Full-Form	Short Messaging Service	Multimedia Messaging Service
Definition	It is a method to send text messages over a cellular network.	It is a method to send text messages along with multimedia over a cellular network.
Multimedia Support	It does not support multimedia.	It supports multimedia.
Cost	SMS is cheaper in comparison to MMS.	MMS is three times costlier in comparison to SMS.
User-base	The user-base of SMS is large in comparison to MMS.	The user-base of MMS is small in comparison to SMS.
Length	160 characters	1600 characters
Links	We can send all the multimedia with the help of a link.	We can send links as well as multimedia directly with the in the MMS.
Compatibility	Compatible with all the devices.	Not compatible with analog mobile phones.
Invention	According to Wikipedia, the first SMS was sent in 1992.	MMS was just an extension of SMS technology. The first MMS capable phone introduced around 2002

4.5.1 SMSManager

- Android provides full SMS functionality from within your applications through the SMSManager.
- SMSManager class manages operations like sending a text message, data message, and multimedia messages (MMS). For sending a text message method `sendTextMessage()` is used likewise for multimedia message `sendMultimediaMessage()` and for data message `sendDataMessage()` method is used.

Function	Description
<code>sendTextMessage()</code>	<code>sendTextMessage(String destinationAddress, String scAddress, String text, PendingIntent sentIntent, PendingIntent deliveryIntent, long messageId)</code>
<code>sendDataMessage()</code>	<code>sendDataMessage(String destinationAddress, String scAddress, short destinationPort, byte[] data, PendingIntent sentIntent, PendingIntent deliveryIntent)</code>
<code>sendMultimediaMessage()</code>	<code>sendMultimediaMessage(Context context, Uri contentUri, String locationUri, Bundle configOverrides, PendingIntent sentIntent)</code>

4.6 Graphics and Animation

- Animation is the process of adding a motion effect to any view, image, or text. With the help of an animation, you can add motion or can change the shape of a specific view.
- Android provides a variety of powerful APIs for applying animation to UI elements and drawing custom 2D and 3D graphics.
- Animation are generally used to give your application UI a rich look and feel. Animation in android apps is the process of creating motion and shape change. The basic ways of animation are:
 - Fade In Animation
 - Fade Out Animation
 - Cross Fading Animation
 - Blink Animation
 - Zoom In Animation
 - Zoom Out Animation
 - Rotate Animation
 - Move Animation
 - Slide Up Animation
 - Slide Down Animation
 - Bounce Animation
 - Sequential Animation
 - Together Animation
- The Android framework provides two animation systems: **property animation (introduced in Android 3.0)** and **view animation**. Both animation systems are viable options, but the property animation system, in general, is the preferred method to use, because it is more flexible and offers more features. In addition to these two systems, you can utilize Drawable animation, which allows you to load drawable resources and display them one frame after another.
- The animations are basically of three types as follows:
 - **Property Animation:** Property Animation is one of the robust frameworks which allows animating almost everything. This is one of the powerful and flexible animations which was introduced in Android 3.0. Property animation can be used to add any animation in the CheckBox, RadioButtons, and widgets other than any view.
 - **View Animation:** View Animation can be used to add animation to a specific view to perform tweened animation on views. Tweened animation calculates animation information such as size, rotation, start point, and endpoint. These animations are slower and less flexible. An example of View animation can be used if we want to expand a specific layout in that place we can use View Animation. The example of View Animation can be seen in Expandable RecyclerView.
 - **Drawable Animation:** Drawable Animation is used if you want to animate one image over another. The simple way to understand is to animate drawable is to load the series of drawable one after another to create an animation. A simple example of drawable animation can be seen in many apps Splash screen on apps logo animation.
 - **Example:** <https://www.geeksforgeeks.org/animation-in-android-with-example/>

Important Methods of Animation

Methods	Description
<code>startAnimation()</code>	This method will start the animation.
<code>clearAnimation()</code>	This method will clear the animation running on a specific view.

4.7 Sample Questions

Marks	Sample Question
2	Does Android employ SQLite as a relational database management system? If affirmative, could you provide an illustrative example?
2	Is a cursor utilized for the management of query results in Android databases? In case of affirmation, kindly provide an example.
2	Can helper classes be utilized to initiate and conclude the operations of an Android database? If so, kindly furnish an example.
4	Please elucidate the purpose and function of the component designated for the storage and retrieval of extensive or structured data sets shared among Android applications. How does this component facilitate the exchange of data?
4	Describe the specific permission necessary to access the Bluetooth functionality on Android devices. Expound on its significance and potential use scenarios for Bluetooth-related tasks.
6	Examine the involvement of sensors in Android devices and furnish examples of applications that harness various types of sensors.
6	Delve into the benefits and hurdles associated with the integration of Bluetooth technology in the development of Android applications.
6	Contrast and equate GPS-based and network-based location services in the Android ecosystem.
3	Define the concept of "Geo Location" within the Android context and elaborate on the methodology for ascertaining a device's location through GPS or other location techniques.
3	Clarify the abbreviations "SMS" and "MMS" as they pertain to Android. Disclose the full meanings of these terms and outline their distinctions.
3	Android provides a comprehensive framework for 2D drawing. Examine the pivotal classes involved, such as Canvas, Paint, and others, and their contributions to the drawing process.
4	Describe the realm of graphics and animation within Android app development.
4	Provide instances of scenarios in which you would employ Frame Animation and Tween Animation in an Android application.
18	Formulate a conceptual design for an Android application aimed at assisting users in tracking their daily water consumption. Include the primary interface and essential functionalities.
18	Craft a prototype for an Android application that harnesses sensors to monitor and enhance sleep quality. Elaborate on the app's capabilities and the integration of sensors.

4.8 Related Videos

- **SQLite Database Tutorial - 1**

<https://www.youtube.com/watch?v=DVWGY-4Cc3Y>
<https://www.youtube.com/watch?v=C2uiPWYwMY>
<https://www.youtube.com/watch?v=IQAXYZrOrMg>

- **SQLite Database Tutorial - 2**

<https://www.youtube.com/watch?v=Ip0LaPUcxiU>
<https://www.youtube.com/watch?v=Hqo4Nr2d3DA>
<https://www.youtube.com/watch?v=7NJavvSLYx8>
<https://www.youtube.com/watch?v=SoqujjHp08Q>
<https://www.youtube.com/watch?v=rK4walNCMzI>
<https://www.youtube.com/watch?v=lqoVrGWFrJI>

- **Sensors**

<https://www.youtube.com/watch?v=H0dxrwRT1aE>
<https://www.youtube.com/watch?v=gVszXHio7hU>
<https://www.youtube.com/watch?v=Sv0KrWf7cmc>
<https://www.youtube.com/watch?v=R-8k2GYK1G4>

- **Bluetooth**

<https://www.youtube.com/watch?v=KfM5N6m10kY>

- **Geo Location - Location Based Services**

<https://www.youtube.com/watch?v=B01uthYhsms>
<https://www.youtube.com/watch?v=xg6r4AnPZzY>
<https://www.youtube.com/watch?v=InTLAZ99Y40>

- **SMS and MMS**

<https://www.youtube.com/watch?v=ksD2ItUkfLc>

- **Graphics and Animation**

<https://www.youtube.com/watch?v=LoluU2Jg0Es>
<https://www.youtube.com/watch?v=XHgqo-PgjGA>