Unit 10:
Major Elements for Building Apps

Mobile Application Development (MAD)
CCIT4059, 2025-2026

Outline

- Major Elements for Building Apps
  - Broadcast Intents, Broadcast Receivers
  - Content Providers
  - Application Manifest, Resources, Context
  - App Widgets
  - Android Notification
  - Process and Thread Issues
  - Data Storage in Android
- Other Issues in Android Apps
  - Other Features

Four Essential Components for App Development

- An android application comprises four essential components and some additional components.
  - The four components serve a specific purpose and have a specific lifecycle. They define a path for both user and system to how they can start or leave an application

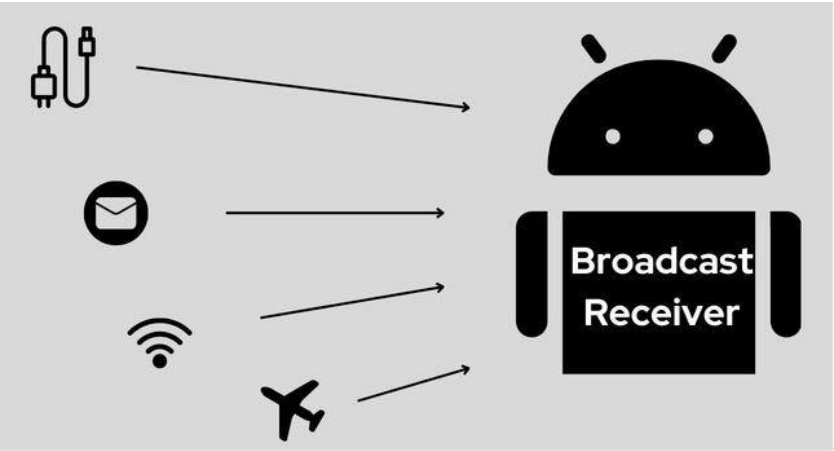
- Four components are essential for any Android-based application. Namely,

1. Activities
2. Services
3. Broadcast Receivers
4. Content Providers

Broadcast Receivers

- A Broadcast Receiver in Android is a component that lets your app listen for system-wide or app-specific messages.
  - These messages, called "intents," indicate that an event has happened.

- Many broadcasts originate from the system.
  - E.g. screen has turned off, low battery status, a picture was captured, or SMS is received.
  - E.g, the app can listen for system messages like the device booting up or a low battery warning, as well as custom messages from other apps.



Broadcast Receivers

- These messages, called **"intents,"** indicate that an event has happened.

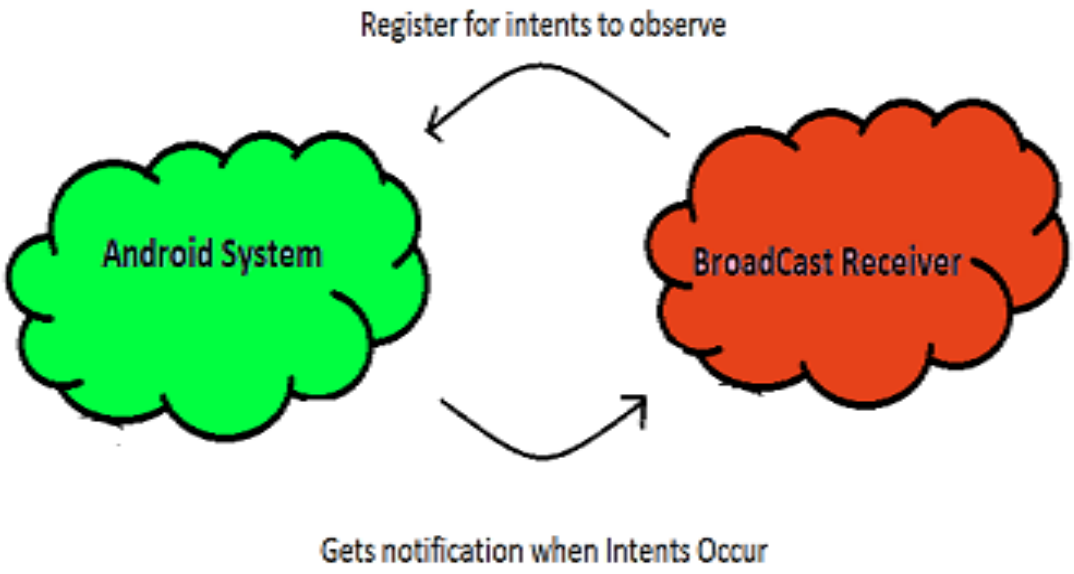
- Applications can also create broadcasts receivers to listen for specific broadcast intents.
  - E.g. to let other applications know that some data has been downloaded to the device and is ready to use.

- A Broadcast Receiver must be registered by an app and configured with an Intent Filter to indicate the types of broadcast in which it is interested.

Broadcast Receivers

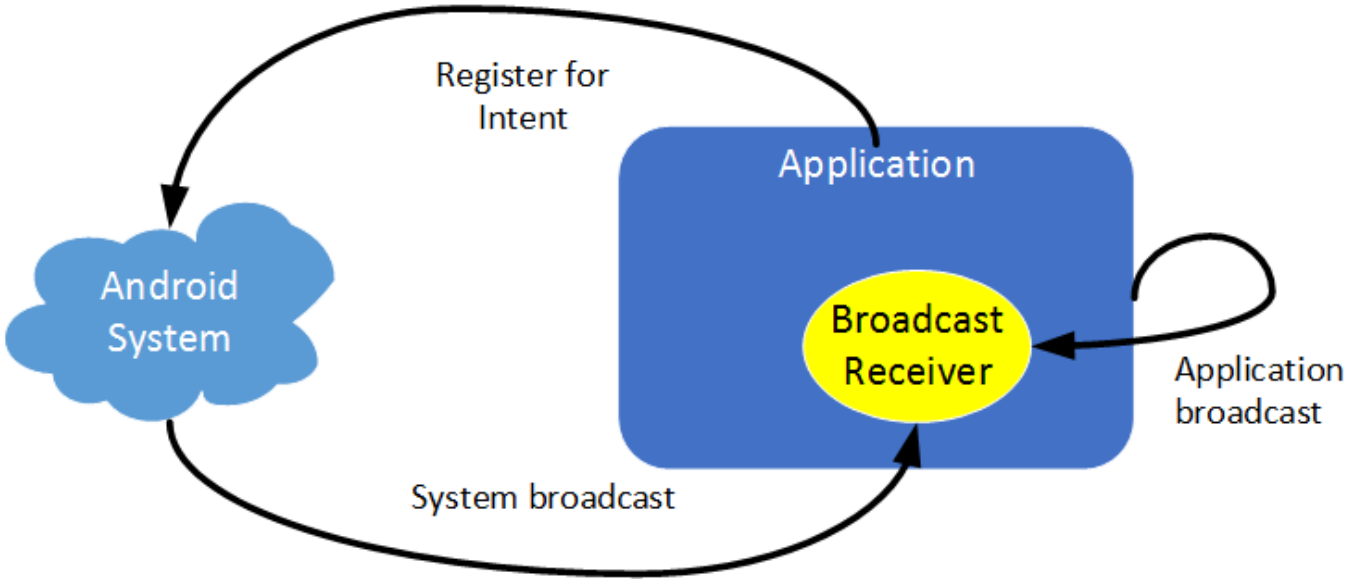
- When an event happens, the Android system creates a **broadcast intent** that multiple apps can receive.

- The Broadcast Receiver acts as a link between the system or other apps and your app, letting it respond to events without always running in the background.



Broadcast Receivers

- A broadcast receiver is a component that responds to broadcast messages (announcements).
  - Developers subclass `BroadcastReceiver` and implement the `onReceive(Context, Intent)` method.



Broadcast Intents

- Apart from Android classes `Activity` and `Intent`, there are other major elements for building Android Apps

- Broadcast Intents
  - Broadcast Intents serve as a crucial mechanism for communication within the Android system.
  - Allow apps to send or receive broadcast messages

- E.g. Android system sends out Broadcast Intents to indicate device status change such as end of system start up, or screen being turned on or off, on or off of airplane mode)

Broadcast Intents

- Broadcast Intents can be classified into two major types:
  - System Broadcast Intents
  - Custom Broadcast Intents

1. System Broadcast Intents
   - Android system automatically sends broadcasts when various system events occur.
   - Sent by the Android system (e.g., ACTION_BOOT_COMPLETED, ACTION_BATTERY_LOW)
   - E.g. When the system switches in and out of airplane mode, it sends a broadcast with the action string android.intent.action.AIRPLANE_MODE_CHANGED

Broadcast Intents

2. Custom Broadcast Intents
   - Sent by your app or other apps.
   - Apps can also send custom broadcasts to notify other apps of specific events.
   - For example, an app might broadcast that new data has been downloaded.
   - Custom broadcasts allow inter-application communication and coordination.

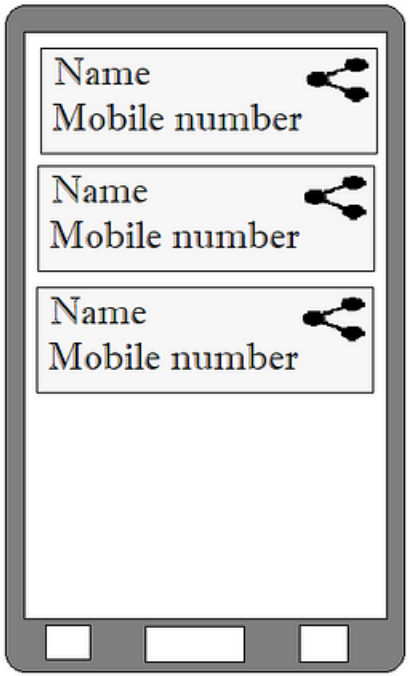
Broadcast Intents

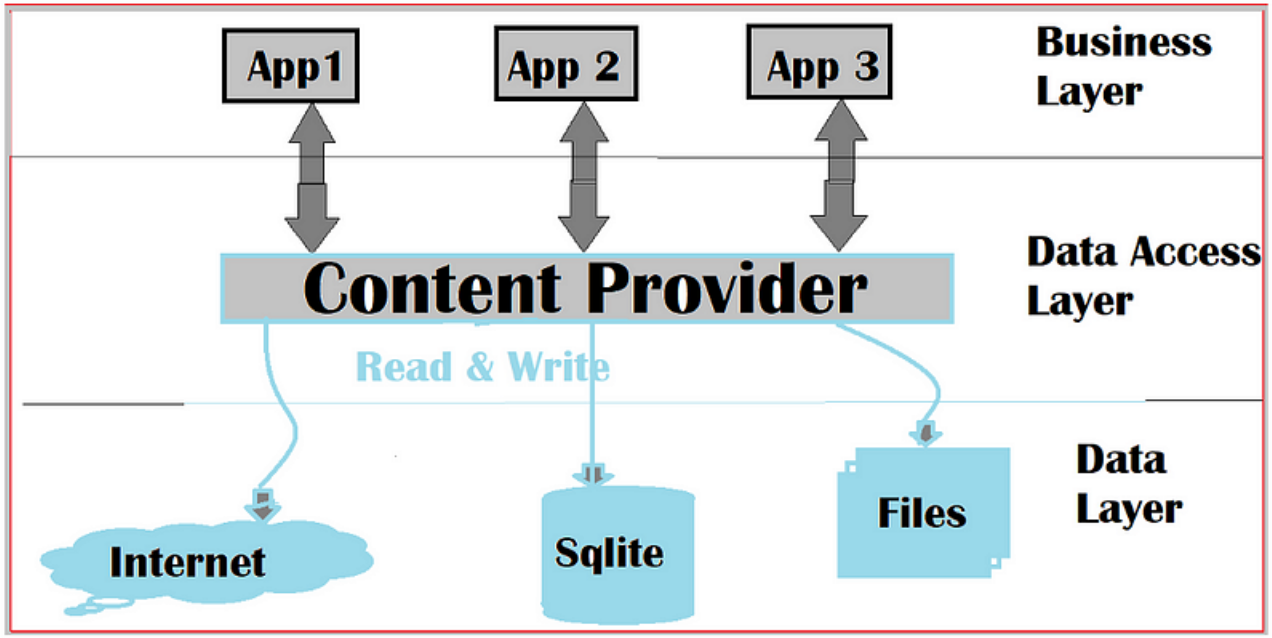
- Broadcast intent is a messaging object,
- Broadcast receiver is an app component.
- An intent is used to request some action from some app component (for a broadcast receiver, an activity or a service)

- Broadcasts can be used as a messaging system across apps and outside of the normal user flow.
  - However, developers must be careful not to abuse the opportunity to respond to broadcasts and run background jobs that could impact system performance.

Content Providers

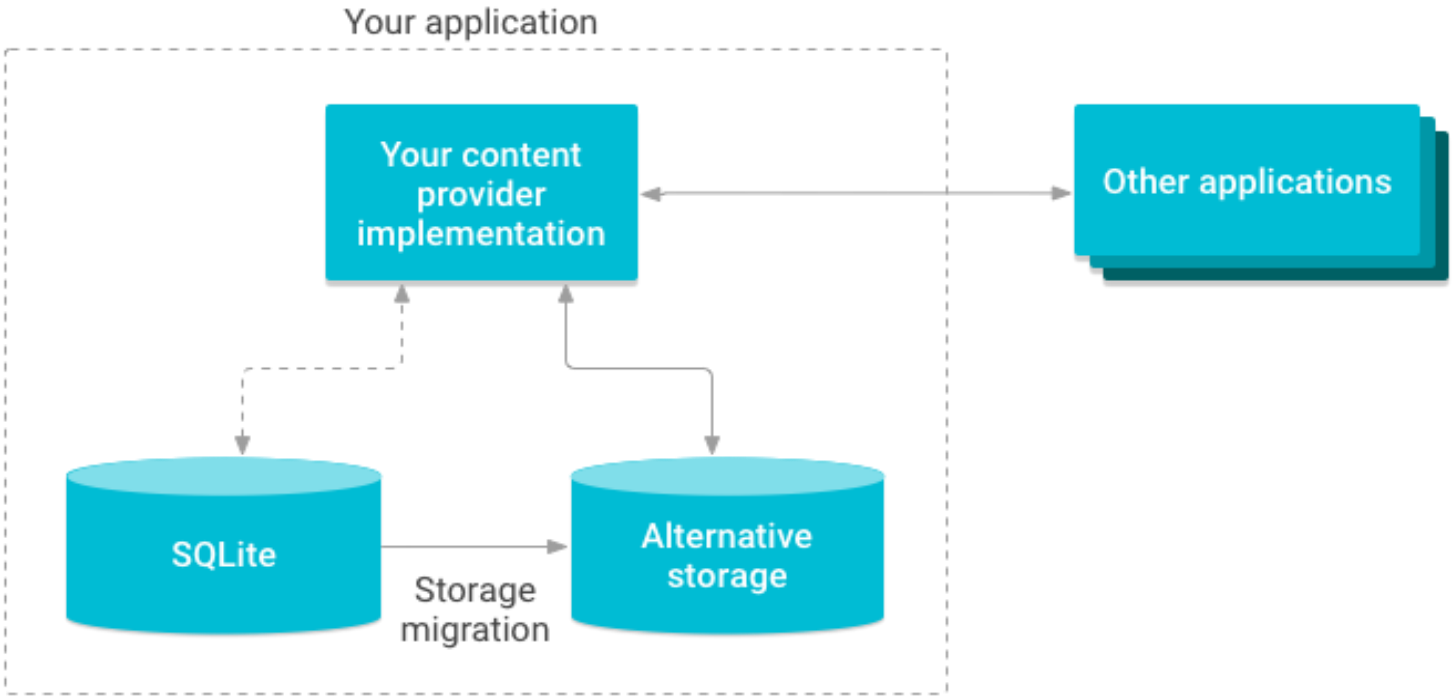
- Overview
  - Content Providers implement a mechanism to manage access to data stored by the application itself, stored by other apps, and provide a way to share data with other apps.





Content Providers

– Any app can provide other apps with access to its underlying data through the implementation of a Content Provider including the ability to add, remove and query *(subject to permissions)*
  - Access to the data is provided via a Universal Resource Identifier (URI) defined by the Content Provider



Summary for Four Major Components

Activities

1. Provides User Interface
2. Usually represents a Single Screen
3. Can contain one/more Views

Services

1. No User Interface
2. Runs in Background

Intent/Broadcast Receiver

1. Receives and Reacts to broadcast Intents
2. No UI but can start an Activity

Content Provider

1. Makes application data available to other apps
2. Data stored in SQLite database

Major Elements for Building Apps (Manifest)

- Application Manifest (`AndroidManifest.xml`)
  - An XML-based file that app outlines activities, services, broadcast receivers, data providers and permissions that make up a complete app
  - Before the Android system can start an app component, the system must know that the component exists by reading the app's AndroidManifest.xml file (the "manifest" file)
  - An app must declare all its components in this file, which must be at the root of the app project directory

Major Elements for Building Apps (App Permissions)

- App permissions help support user privacy by protecting access to the following:
  - Restricted data, such as system state and users' contact information
  - Restricted actions, such as connecting to a paired device and recording audio
- Android App Permissions are divided into several protection levels
  - The protection level affects whether runtime permission requests are required.
- Three protection levels affect third-party apps:
  - *normal*, *signature*, and *dangerous* permissions.

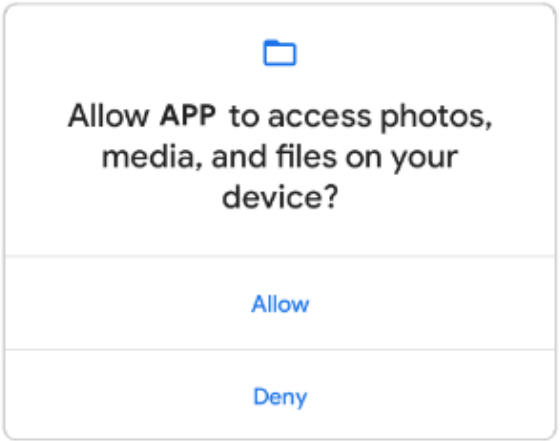


Figure 3. *The system permission prompt that appears when your app requests a runtime permission.*

Major Elements for Building Apps (App Permissions)

- If the app lists *normal* permissions in its manifest (that is, permissions that don't pose much risk to the user's privacy or the device's operation), the system automatically grants those permissions to your app.
- For *signature* permissions, the system grants these app permissions at install time, but only when the app that attempts to use a permission is signed by the same certificate as the app that defines the permission

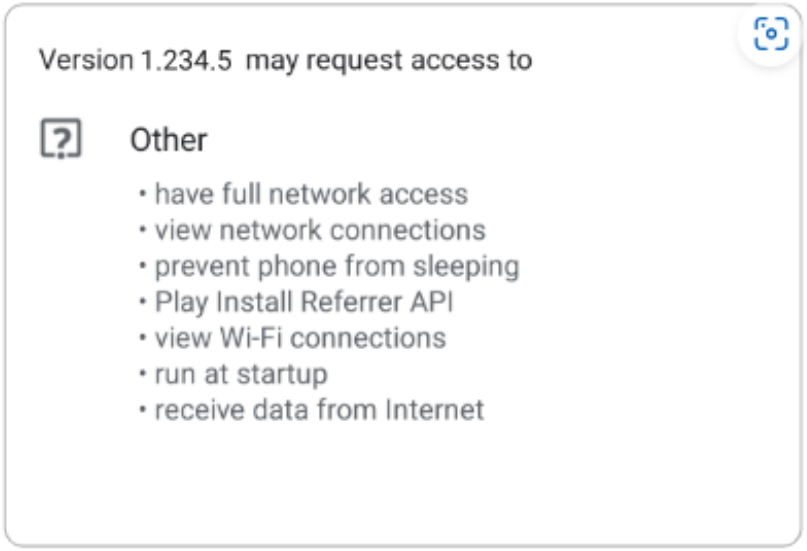


Figure 2. *The list of an app's install-time permissions, which appears in an app store.*

Major Elements for Building Apps (App Permissions)

Reference Only

- If the app lists *dangerous* permissions in its manifest (that is, permissions that could potentially affect the user's privacy or the device's normal operation), such as the `SEND_SMS` permission above, the user must explicitly agree to grant those permissions.
  - If the device is running Android 6.0 (API level 23) or higher, user isn't notified of any app permissions at install time, but at runtime.
  - For Android 5.1.1 (API level 22) or lower, system automatically asks user to grant all dangerous permissions for your app at install-time

Major Elements for Building Apps (List of Dangerous Permissions)

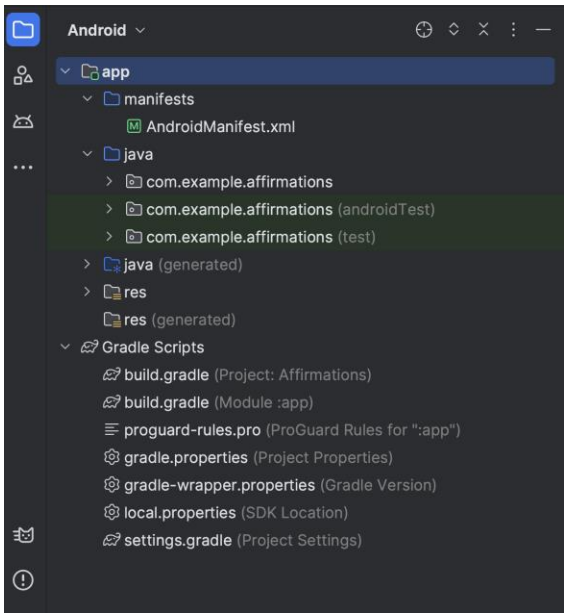
Reference Only

Table 1. Dangerous permissions and permission groups.

Permission Group	Permissions
CALENDAR	• READ_CALENDAR • WRITE_CALENDAR
CALL_LOG	• READ_CALL_LOG • WRITE_CALL_LOG • PROCESS_OUTGOING_CALLS
CAMERA	• CAMERA
CONTACTS	• READ_CONTACTS • WRITE_CONTACTS • GET_ACCOUNTS
LOCATION	• ACCESS_FINE_LOCATION • ACCESS_COARSE_LOCATION
MICROPHONE	• RECORD_AUDIO
PHONE	• READ_PHONE_STATE • READ_PHONE_NUMBERS • CALL_PHONE • ANSWER_PHONE_CALLS • ADD_VOICEMAIL • USE_SIP
SENSORS	• BODY_SENSORS
SMS	• SEND_SMS • RECEIVE_SMS • READ_SMS • RECEIVE_WAP_PUSH • RECEIVE_MMS
STORAGE	• READ_EXTERNAL_STORAGE • WRITE_EXTERNAL_STORAGE

Major Elements for Building Apps (Resources)

- Application Resources
  - In addition to manifest file and Dex files that contain the byte code, an Android application package will also typically contain a collection of resource files
  - These files contain resources such as the ***strings***, ***images***, ***fonts*** **and** ***colors*** that appear in the user interface together with the XML representation of the user interface *layouts*
  - By default, these files are stored in the **`/res`** sub-directory of the application project's hierarchy



Major Elements for Building Apps (Context)

For reference

- Application Context
  - When an application is compiled, a class named **`R`** is created that contains references to the application resources
  - The manifest file and resources combine to create what is known as the *Application Context*
    - This context, represented by the Android **`Context`** class, may be used in the application code to gain access to the application resources at runtime
  - A wide range of methods may be called on an app's context to gather information and make changes to the app's environment at runtime
  - Example codes of using the class **`R`**:

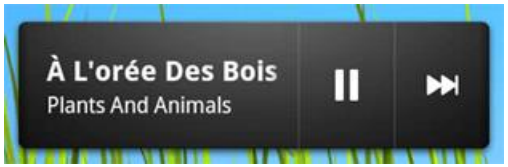
```
setContentView(R.layout.activity_main)
(Button)findViewById(R.id.button)
```

Major Elements for Building Apps (Android App Widgets)

- App Widgets are miniature application views that can be embedded in other applications (such as the Home screen) and receive periodic updates

- These views are referred to as Widgets in the user interface, and you can publish one with an App Widget provider

- An application component that is able to hold other App Widgets is called an App Widget host

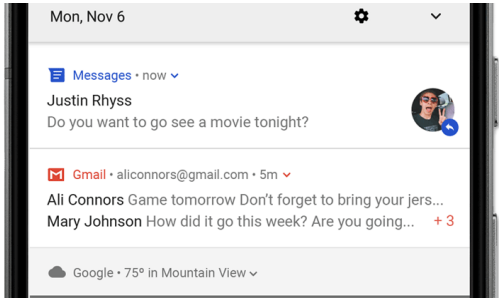


Major Elements for Building Apps (Android Notification)

- A notification is a message displayed to the user outside of application's normal UI

- When the system issues a notification, it first appears as an icon in the notification area
  - To see the details of the notification, the user opens the notification drawer

- Both the notification area and the notification drawer are system-controlled areas that the user can view at any time



Other Issues in Android Apps

- Apart from the major Android elements, Android developers should also know other features and issues related to developing Android apps.
- When an app component starts and the app does not have any other components running, Android starts a new process for the app with a single thread of execution
  - By default, all components of the same app run in the same process and thread (called the "main" thread)
  - If an app component starts and there already exists a process for that app (because another component from the app exists), then the component is started within that process and uses the same thread of execution

Other Issues in Android Apps (Process and Thread in Android)

- A thread is a thread of execution in a program. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently.

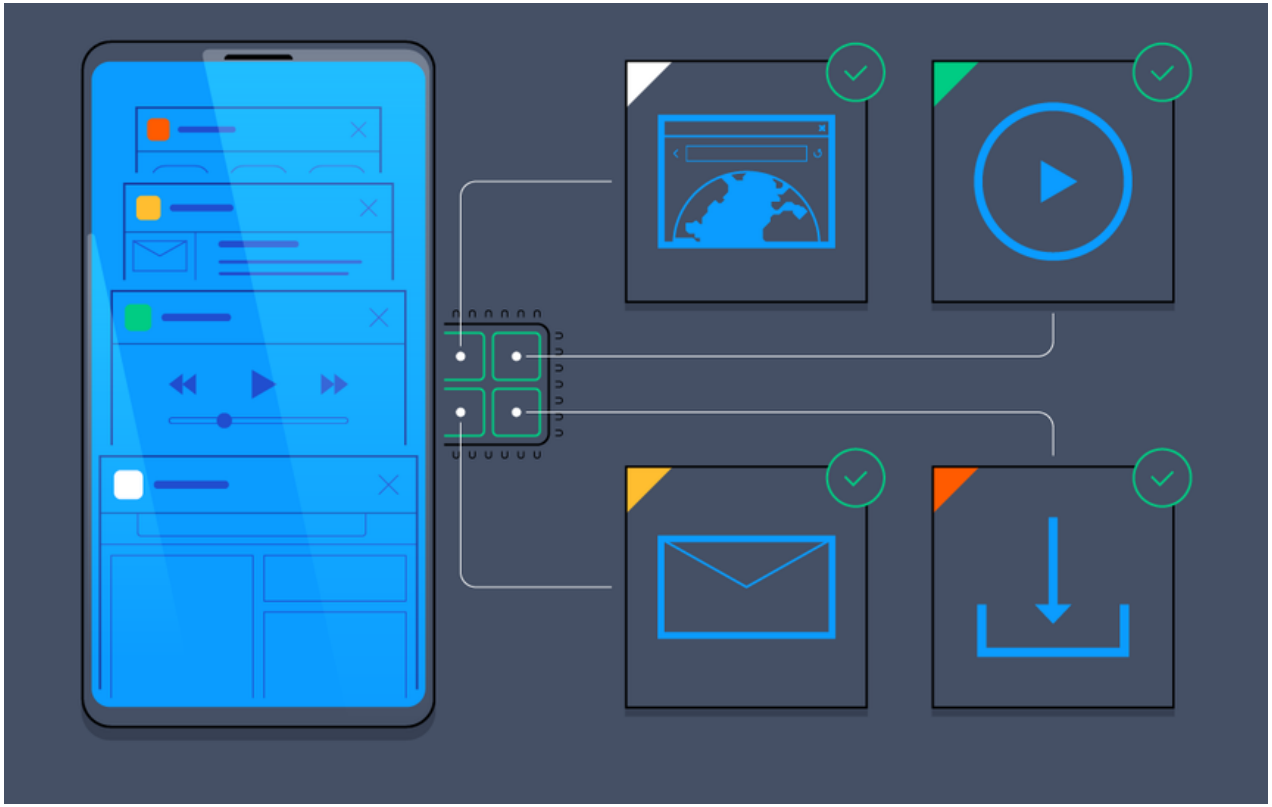
- When an application is launched in Android, it creates the first thread of execution, known as the “main” thread.

- The main thread is responsible for dispatching events to the appropriate user interface widgets as well as communicating with components from the Android UI toolkit.

- Every thread has a priority. Threads with higher priority are executed in preference to threads with lower priority.

Other Issues in Android Apps (Process and Thread in Android)

- We may also arrange for different components in our application to run in separate processes, and we can create additional threads for any process as in Java
  - E.g. There are various ways to create threads in Java, including `implements Runnable`



Other Issues in Android Apps (Main Thread and Worker Thread)

- Main Thread (UI Thread):
  - The main thread, also known as the UI thread, is responsible for handling all UI updates and user interactions in an Android app.
  - Any code that modifies the UI or interacts with the user should run on this main thread.
  - It ensures a smooth user experience by preventing UI-related tasks from blocking each other.

- Worker Thread:
  - Android apps often need to perform background tasks (such as fetching data from a server, processing images, or performing computations) without freezing the UI.
  - Worker threads come to the rescue! They allow you to offload time-consuming operations from the main thread.

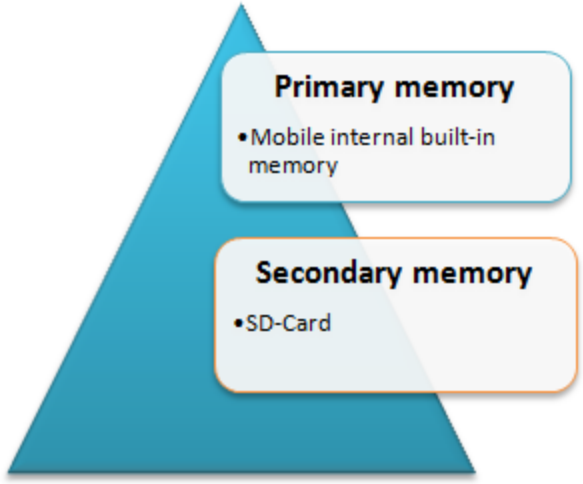
Data Storage in Android Platform

- In the realm of Android development, data storage plays a crucial role in retaining information for future reference.
- Considerations:
  - Space Requirements: Internal storage has limited space for app-specific data. If you need to save a substantial amount of data, explore other storage options.
  - Reliability: If your app's basic functionality relies on certain data (e.g., during startup), place it within internal storage or a database.
  - Data Type: Choose the appropriate storage based on the kind of data you need to store.

Data Storage in Android Platform (Primary Storage)

- In Android we can use the device storage space to store the data in it for the applications. The type of data involves things such as a text file, image file, video file, audio file etc.

- One way is to write the raw files into primary /secondary storage. Another way is to write the cache files into the primary/secondary storage.



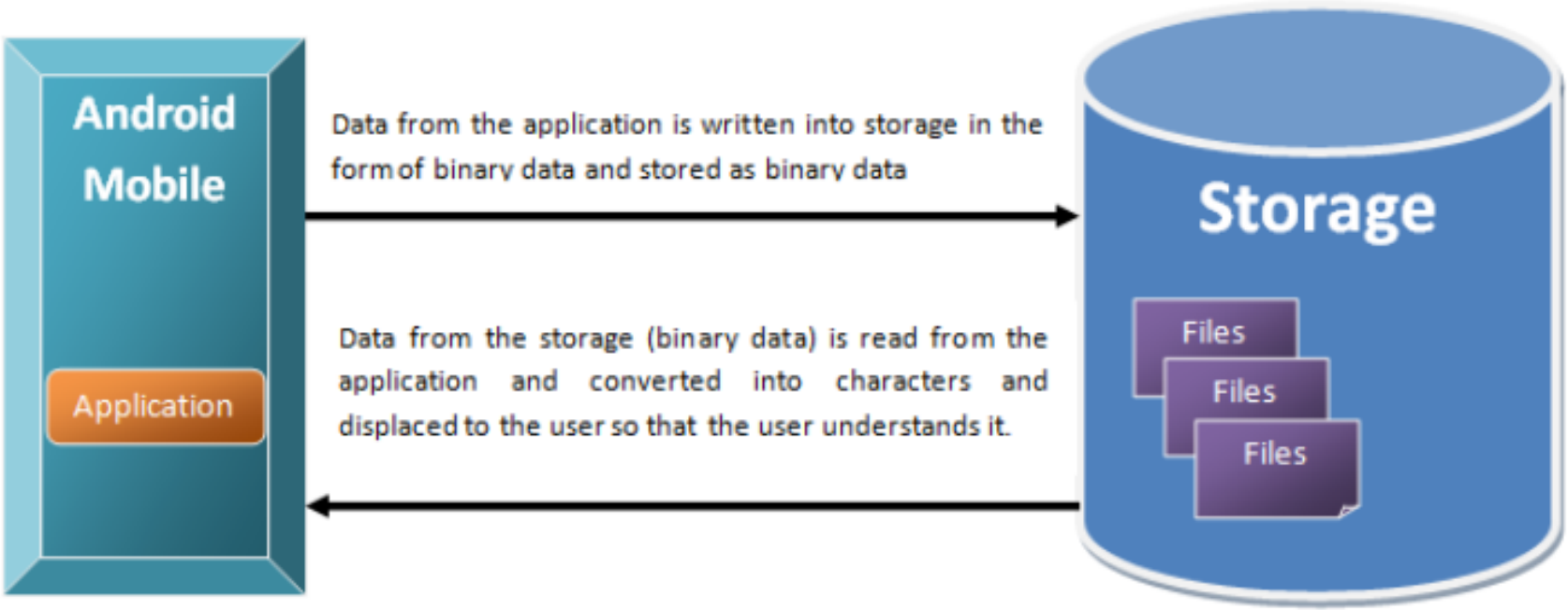
Data Storage in Android Platform (Internal Storage)

- Android provides several types of storage options for managing data within apps.

- There are four major types of storage medium in Android

1. Internal Storage:
    - •Purpose: Dedicated to the app, internal storage is available on all devices.
    - •Usage:
        - Store files meant exclusively for your app's use.
        - Ideal for sensitive information that other apps shouldn't access.
    - •Directories:
        - getFilesDir(): For app-specific files.
        - getCacheDir(): For temporary cache files.
        - getExternalFilesDir(): For app-specific files on external storage.
        - getExternalCacheDir(): For temporary cache files on external storage.

Data Storage in Android Platform (Internal Storage)



Data Storage in Android Platform (Shared Storage)

2. Shared Storage:

- **Purpose**: Used for files your app intends to share with other apps.
- **Content Types**:
  - **Media Files (Images, Audio, Videos)**:
    - Accessible via the **MediaStore API**.
    - Permissions required for all files on Android 9 (API level 28) or lower.
  - **Documents and Other Files**:
    - Shareable content accessed through the **Storage Access Framework**.

Data Storage in Android Platform (Preferences, Databases)

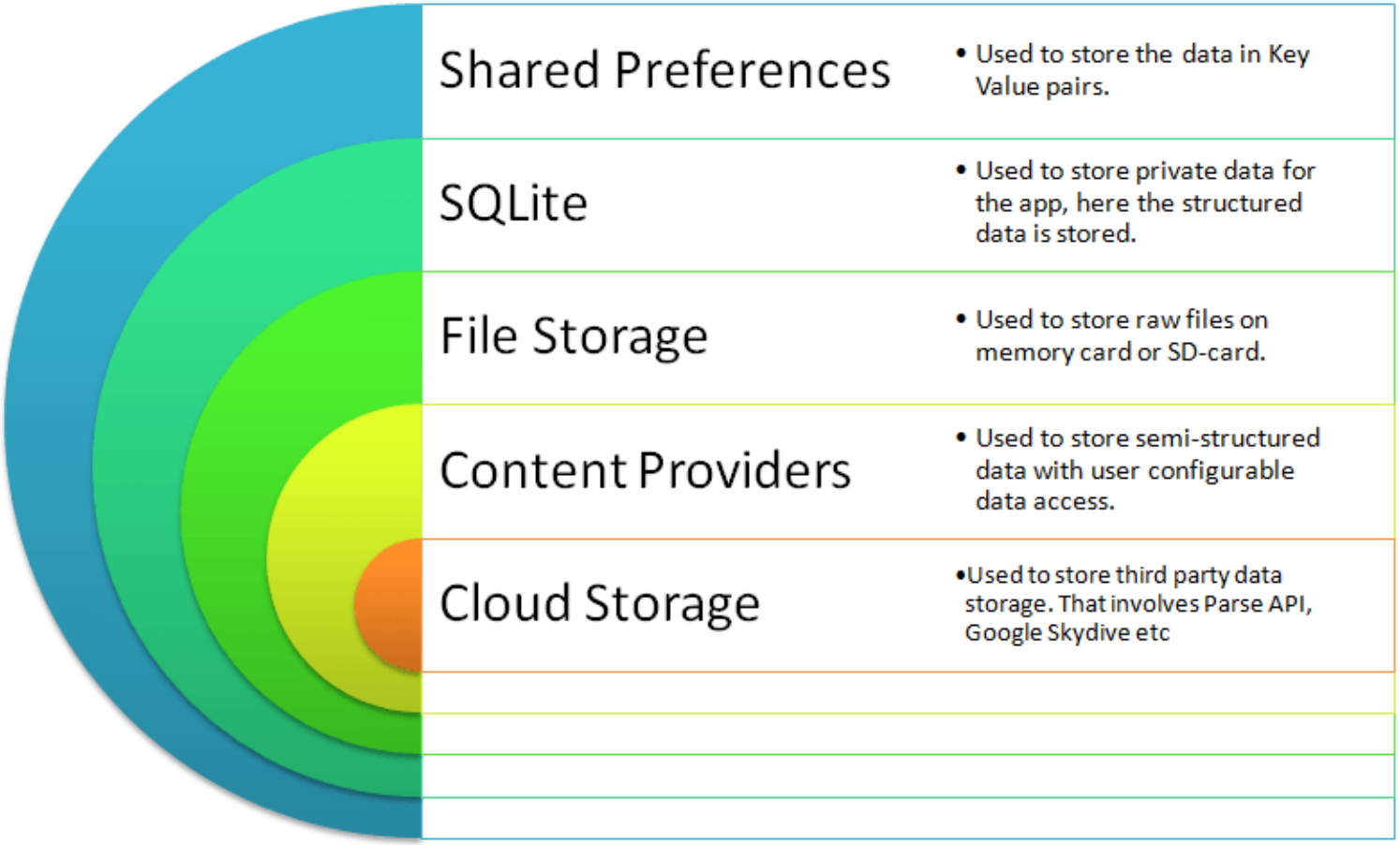
3. Preferences:

1. **Purpose**: Store private, primitive data in key-value pairs.
2. **Library**: Use the **Jetpack Preferences library**.

4. Databases:

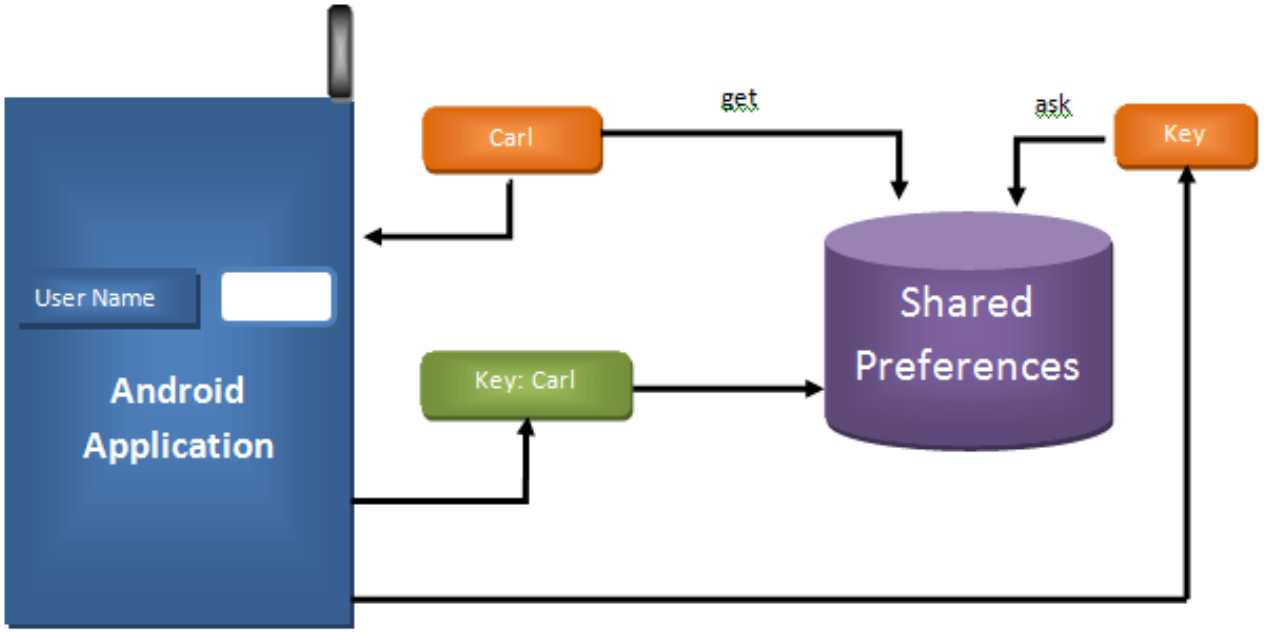
1. **Purpose**: For structured data storage.
2. **Library**: Utilize the **Room persistence library**.

Data Storage in Android Platform (Storage Types)



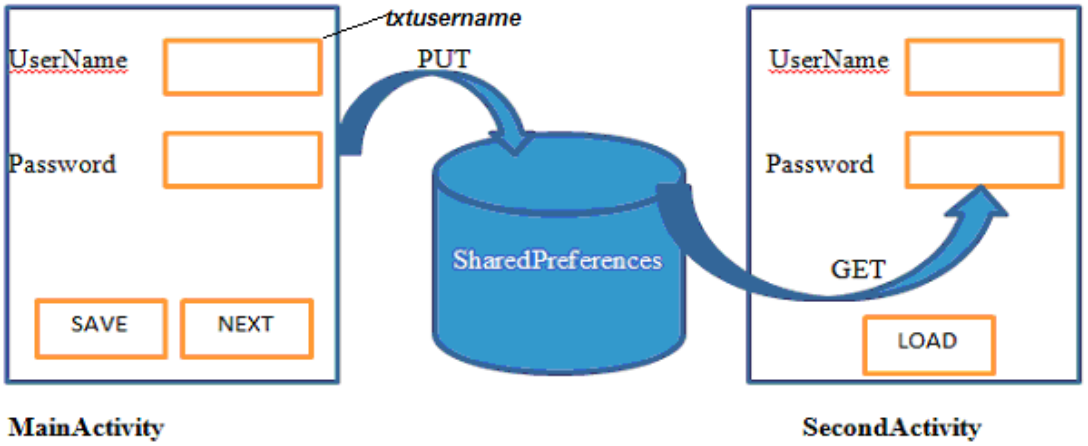
Data Storage in Android Platform (Shared Preferences)

- Shared Preferences is the better way to store and retrieve small amounts of primitive data
- Data stored in the Shared Preferences is stored in key-value pairs.
- This makes retrieving the data simpler, but there is not a really efficient way to query/search for a specific piece of data.



Data Storage in Android Platform (Shared Preferences)

- Shared Preference uses key/value pairs (like a dictionary in Python)
  - String, int, float, Boolean that make up the preferences in an XML file for the layout inside the app
- A key with name “username” and for the value, you might store the user’s username. And then you could retrieve that by its key (here username).
- Simple shared preference API that you can use to store preferences and pull them back as and when needed.
  - Shared Preferences class provides APIs for reading, writing, and managing the data.



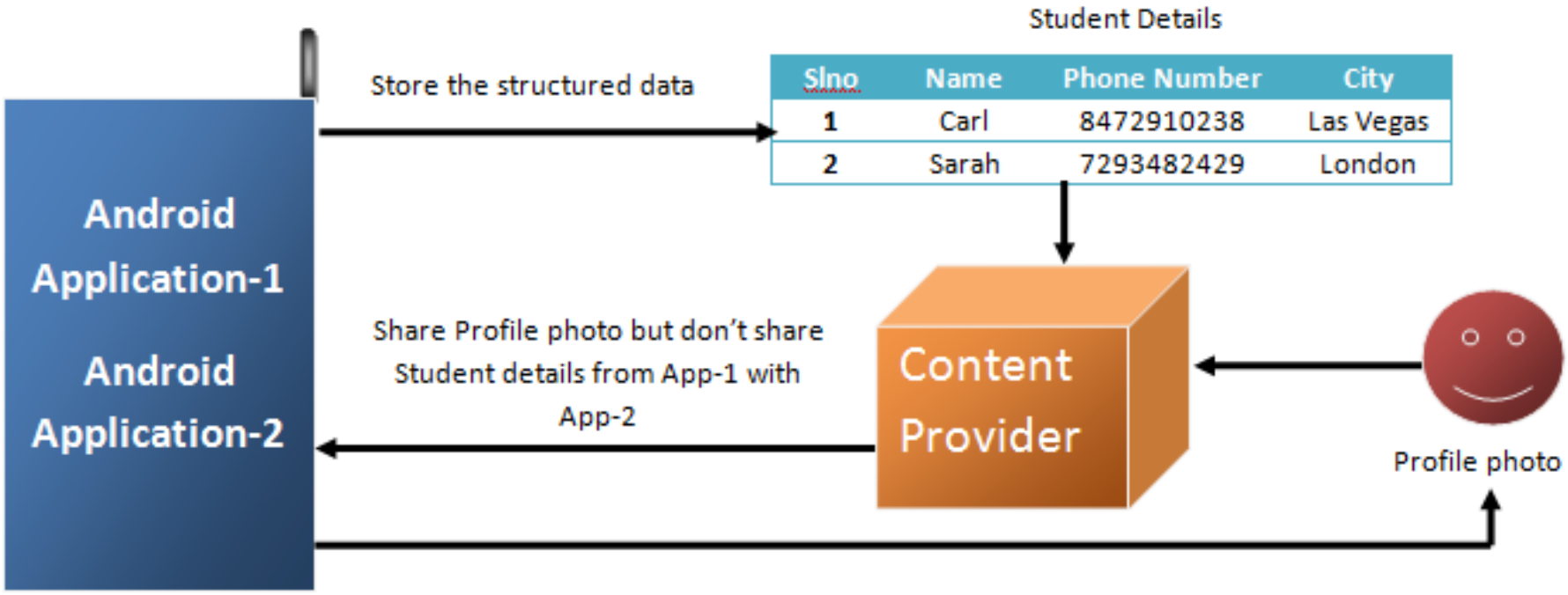
Data Storage in Android Platform (SQLite Database)

- SQLite is used to store more structured data locally in a mobile where the android app is running.
  - Example: Structured data of student information in the form of rows and columns.
- SQLite offers similar functionality like MySQL and Oracle but with limited functional features.
  - Some of the things involve performing query operations on tables but also some features are not available like stored procedure.
- SQLite is very helpful in storing complex and large data, which can be downloaded once and can be used again and again until the application is running.
  - When the application is uninstalled, the SQLite database is also destroyed.

Data Storage in Android Platform (Content Provider)

For reference

- Content Provider is an interface for providing data to other apps;
  - It is used to store semi-structured data with user configurable data access
  - It doesn't directly involve the actual storage of the data itself, although generally they will be backed by a database.



Other Issues in Android Apps (Other Features)

- Other Features Android supports include:
  - **Touch and Gesture**
  - Media, Camera and Microphone
  - **Sensors: Accelerometer, Motion**, Location
  - **Data Storage Access and Database (SQLite)**
  - Connectivity / Networking
  - Other Device Types: Wearable, TV, Auto, etc.
  - And more …

References

- Part of this slide set is prepared and extracted from the followings:
  - Gok, N. and Khanna, N. (2013) *'Building Hybrid Android Apps with Java and JavaScript'*, O'Reilly Media, Inc.
  - Gargenta, M. and Nakamura, M. (2014) *'Learning Android'*, 2ed, O'Reilly Media, Inc.
  - Google Inc. 'Website for Android Developers', http://developer.android.com/
  - Google Inc. 'Website for Android Sources', https://source.android.com/
  - Apple Inc. 'Website for Apple Developers', Website: https://developer.apple.com/
  - Microsoft Corp. 'Website for Windows Developers', https://developer.microsoft.com/en-us/windows
  - Wikipedia Website: http://en.wikipedia.org
- This set of slides is for teaching purpose only