Android Mobile Application Development
Week 7: Android and Firebase

Data Persistence

Most apps require data to be stored persistently, meaning across app launches. There are two approaches to data persistence. Data can be persisted locally on the device for data only accessed by the one user in the Android app, or saved in the cloud if the data will be accessed or aggregated across users, or accessed through different platforms.

Local Data Storage

https://developer.android.com/training/data-storage

There are multiple local data storage approaches on Android. The approach you pick should be based on

- What kind of data you need to store
- How much space your data requires
- How reliable does data access needs to be
- Whether the data should be private to your app

App-specific storage

https://developer.android.com/training/data-storage/app-specific

All Android devices have two file storage areas for files in your app that other apps don't, or shouldn't, access: "internal" and "external" storage. These names come from the early days of Android, when most devices offered built-in non-volatile memory (internal storage), plus a removable storage medium such as a micro SD card (external storage). Many devices now divide the permanent storage space into separate "internal" and "external" partitions.

Internal file storage

Internal storage is best when you want to be sure that neither the user nor other apps can access the data. No permission is needed to read and write files to internal storage.

Pros

- Always available
- By default files saved are private to your app
- Other apps and the user can't access the files
- Starting in Android 10(API 29) internal storage directories will be encrypted

Cons

- Should check the amount of free space before saving files
- Hard to share data
- Internal storage might have limited capacity

External file storage

External storage often provides more space than internal storage for app-specific files. Other apps can access these files if they have the proper permissions.

Pros

- Often provides increased capacity

Cons

- Might not always be available as it might resides on a physical volume that the user might be able to remove
- Must verify that the volume is accessible before using

Both internal and external storage provide directories for an app's persistent files and another for an app's cached files.
In both internal and external storage files are removed when the user uninstalls your app

For files that you want to persist past the lifetime of an app you should use shared storage instead.

Shared storage
https://developer.android.com/training/data-storage/shared
Use shared storage for user data that should be accessible to other apps and saved even if the user uninstalls your app.

- Use the MediaStore API to store media content in a common location on the device
- The Storage Access Framework uses a document provider to store documents and other files in a specific directory

Key-Value Data
https://developer.android.com/training/data-storage/shared-preferences.html
Shared Preferences (different than preferences through Settings)
For small amounts of data that doesn't require structure saving the data as key-value pairs is a good choice.
The Shared Preferences API has been around since API 1and stores data as key-value pairs in an unencrypted XML file in internal storage.

- Keys are always of type String
- Values must be primitive data types: boolean, float, int, long, String, and stringset
- You can use a single file or multiple files
- You can use the default or a named preference
- Preference data can be deleted from the device by the user
- Data is deleted when the app is uninstalled

Some downsides of SharedPreferences include:

- Lack of safety from runtime exceptions
- Lack of a fully asynchronous API
- Lack of main thread safety
- No type safety

DataStore
https://developer.android.com/topic/libraries/architecture/datastore
The new DataStore library has been added to Android Jetpack and provides an improved local data solution using key-value pairs. It is now suggested over Shared Preferences.
DataStore provides two different implementations.

1. Preferences DataStore to store data using key-value pairs
2. Proto DataStore to store data as instances of a custom data type using protocol buffers for structured data

Both DataStore implementations store data asynchronously, consistently, and transactionally, overcoming most of the drawbacks of SharedPreferences. DataStore uses coroutines and Flow to store data asynchronously.

Databases

https://developer.android.com/training/data-storage/room
Relational databases (RDBMS) are a good choice to store structured data. They handle complex data with relationships and enforce data integrity. They are also a good fit for larger amounts of data that will be accessed and manipulated often.
In a RDBMS database data is organized in a collection of tables with defined relationships. This structure and organization is referred to as the database schema.
A table is organized into columns that represent what is stored in the table. Each row in a table contains one record of data.
Android's SQLite database is an RDBMS well suited for persisting large amounts of structured data locally. Similar to internal storage, Android stores your database in your app's private folder and therefore is not accessible to other apps or the user.
The Room library provides an abstraction layer over SQLite that makes it much easier to work with SQLite. It is highly recommended instead of using SQLite APIs directly. It is one of the architecture components that is included in Jetpack.

Local Data Storage Advantages:
- No internet access needed
- Local control
- No fees
- Speed not network dependent

Local Data Storage Disadvantages:
- Subject to device failure/loss
- No backup/recovery ability

Cloud Data Storage
Setting up your own database in the cloud, and handling data syncing, takes a lot of work. Services that handle this for you such as Heroku and Firebase have become very popular as they provide platforms in the cloud for app data management.

Cloud Data Storage Advantages:
- Cross-platform access
- Ability to aggregate data
- Can handle large amounts of data
- Often supports local offline access as well
- Recovery/backup ability

Cloud Data Storage Disadvantages:
- Internet access needed
- Fees involved
- Downtime
- Performance network dependent