>>> UNIT-1

- A brief history of Mobile

→ Mobile technology has seen significant advancements over the years, shaping how we communicate and interact with the world.

1) Fvist mobile call - 1973

→ The fvist mobile phone call was made by Dr Martin Cooper using a Motorola DynaTAC. It was a large and heavy device. (1.1 Kg).

2) Motorola's DynaTAC 8000X - 1983

→ The first commercially available mobile phone, known as the "brick" phone, was released.

3) SMS Introduced - 1992

→ Short Message Service (SMS) was introduced, allowing text messages to be sent between mobile phones.

4) Late 1990s - Mobile revolution

→ Mobile phones became smaller and more affordable, leading to widespread adoption.

5) Rise of smartphones - 2000s

→ The 2000s saw the emergence of smartphones like the BlackBerry and Nokia 3310.

6) iPhone debut - 2007

→ Apple's iPhone revolutionized the industry with a touch screen and the App Store.

7) Android and iOS dominance - 2010s

→ Android and iOS became the leading mobile operating systems, with a wide range of smartphones and apps.

8) 5G technology - 2019

→ The rollout / Launch of the 5G network. Mobile phones like Xiaomi Mi Note 10 are launched, which was the first smartphone to have a 108 MP Camera.

9) Present - Smartphones are essential

→ Mobile phones are now integral to modern life, offering a wide range of features beyond just calling and texting.

10) Foldable phones and beyond - The future

→ Mobile technology continues to evolve with the development of foldable phones, AR/VR capabilities and more.

- Mobile phone generations

* 1G (First Generation) - Early 1980s

⇒ Use analog signals for communication.

⇒ Primarily focused on voice communication with basic sound quality.

* 2G (Second Generation) - Early 1990s

⇒ Transition from analog to digital

⇒ Improved sound quality and introduction to data services like SMS.

* 3G (Third Generation) - Early 2000s

⇒ Faster data transfer.

⇒ Introduction to video calling and mobile internet.

⇒ Video calling capabilities
⇒ Improved multimedia support.

* 4G (Fourth Generation) - Around 2010.

⇒ LTE (Long term evaluation) was mainly for internet.
⇒ VO-LTE (Voice over LTE) is for both voice and the internet.
⇒ Introduction of mobile broadband.

* 5G (Fifth Generation) - Late 2010s.

⇒ Ultra-fast data speeds
⇒ Lower latency.
⇒ Connectivity improvements for IOT, VR and AR

• Mobile Ecosystem

→ The mobile ecosystem refers to the interconnected network of hardware (mobile phones, Tablets etc), software (OS) and services (sms, transaction etc) that enable the functioning of mobile technology.

* It includes :

1. Hardware :- Mobile devices like smartphones, tablets & wearables.

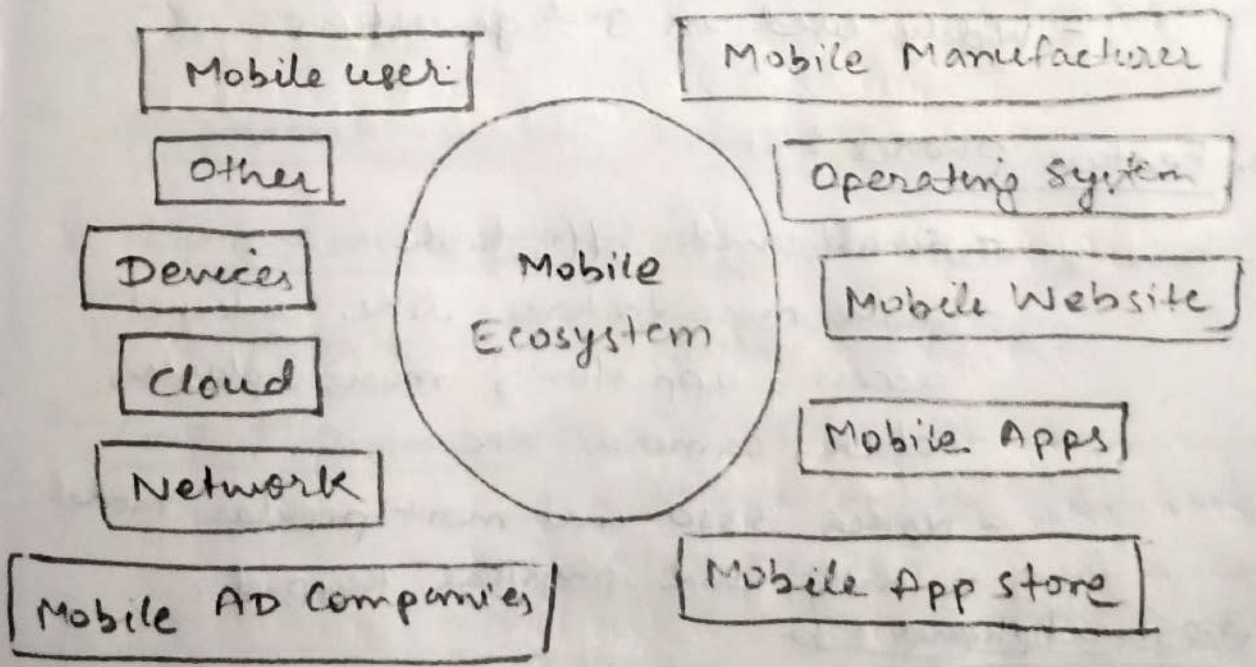
2. Software :- Operating systems (eg. Android, iOS)

3. Network :- Telecommunication networks like 2G, 3G, 4G, 5G and local wireless communication like Wifi, & and bluetooth.

4. Services :- Play store or App Store.

5. Cloud :- Google Drive, iCloud.

6. Security :- Encription, Biometric Authentication.



- **Types of Mobile**

→ There are different types of mobile phones like Smartphones, Feature phone, Foldable phones, Gaming phone, Camera phone, ~~Etc~~ etc. but

→ the mainly three important types are :-

1. Cell phone
2. Feature phone
3. Smartphone

1. **Cell phones :-**
   - Portable telephone uses cellular networking.
   - Generally used for calling, message sending and do calculations.
   - Widely used in 3rd generation.

2. **Feature phones :-**
   - Small and affordable.
   - Have more features like internet access, app store, music player, clock, camera etc.
   - Nokia 3310 was most popular model.
   - may have physical keypad.

3. ~~Smartphones~~

3. Smartphone :-

- High hardware specification and act like a miniature computer.
- They have touchscreen.
- Improved camera, high computation power, app ecosystem, internet connectivity etc.
- Google Android phones, iphone etc.

- Mobile Information Architecture

→ Mobile Information architecture involves organizing and structuring information on mobile devices to enhance user experience.

→ It's like the blueprint for a mobile app or website.

* Important aspects include :

1. Navigation :- Navigation menu, and search functionality.

2. Content Organization :- Hierarchy organization of information. Ensuring important content is easily accessible.

3. Responsive design :- Adaptable layouts to accomodate various screen sizes.

4. Performance Optimization :- loading speed of app should be faster.
   - Load important or critical content first.

5. Security & Privacy :- Robust security measures to protect user data.
   - Privacy policies and settings for user control.

6. Feedback Mechanisms :- Error messages and guidance for users in case of issues.

- Android Versions

1. Android 1.0 (2008)
   - Basic features calling, messaging, web browser
   - Google apps : YouTube, Gmail etc.

2. Android 1.5 [Cupcake - 2009]
   - Added widgets for home screen.
   - On Screen Keyboard.
   - Video recording.

3. Android 1.6 [Donut - 2003]
   - Quick search box
   - Support different screen size and resolution.

4. Android 2.0/2.1 [Eclair - 2009]
   - Enchanced UI.
   - Introduced Google Maps Navigation.

5. Android 2.2 [Froyo - 2010]
   - Wifi, hotspot.

6. Android 2.3 [Gingerbread - 2010]
   - Copy - paste functionality
   - Enchance gaming

7. Android 3.0 / 3.1 / 3.2 [Honeycomb - 2011]

8. Android 4.0 [Ice Cream Sandwich - 2011]

9. Android 4.1 / 4.2 / 4.3 [Jelly Bean - 2012]

10. Android 4.4 [KitKat - 2013]
    - Personalized Experience

11. Android 5.0 / 5.1 [Lollipop - 2014]
    - Material Design

~~12 Android 7.0/7.1 [Nougat - 2016]~~

12. Android 6.0 [Marshmallow - 2015]
    - Battery saver mode.

13. Android 7.0/7.1 [Nougat - 2016]
    - Multitasking
    - Splitscreen mode

14. Android 8.0/8.1 [Oreo - 2017]
- Autofill

15. Android 9 [Pie - 2018]
- Adaptive Battery & Brightness

16. Android 10 [2019]
- System dark mode

17. Android 11 [2020]
- Conversational Notification (Message through notification) or reply

18. Android 12 [2021]

19. Android 13 [2022]

- **Features of Android**

1. **Open Source :-**
- Encourage customization by developers.

2. **Google Play Store :-**
- Central hub for apps, games etc.

3. **Customization :-**
- High level of personalization with widgets, themes etc

4. Multitasking :-

- Run multiple apps simultaneously.

5. Notification system :-

- Quick access to messages and alerts.

6. Google Assistant :-

- Voice-activated assistance for hand free-tasks.

7. Security :-

- ~~Apps~~ Google play protect ~~clan~~ enhance security, regular updates.

8. Wireless Connectivity :-

- wifi, bluetooth.

- Android Architecture

→ It comprises several layers that work together to provide a robust OS.

→ The main components are :-

1. Applications
2. Application Framework
3. Android Runtime
4. Libraries
5. Linux Kernel

1. Application :-

- Top layer of android architecture.
- These includes pre-installed system apps like camera or apps downloaded from the play Store.
- Apps run within Android Runtime (ART) with the help of classes and services provided by Application Framework.

2. Application Framework :-

- It provides several important classes and APIs which are used to create an android application.
- These can be activities, Content providers and UI.

3. Android Runtime (ART) :-

- Responsible for running and managing applications.
- It is one of the most important part of Android.
- It contains components like Core libraries and Dalvik virtual

Machine (DVM).

- It is responsible for running the byte code, which is the compiled form of the source code (Android app code).

4. Libraries :- The platform libraries includes various C/C++ core libraries and java based libraries such as :

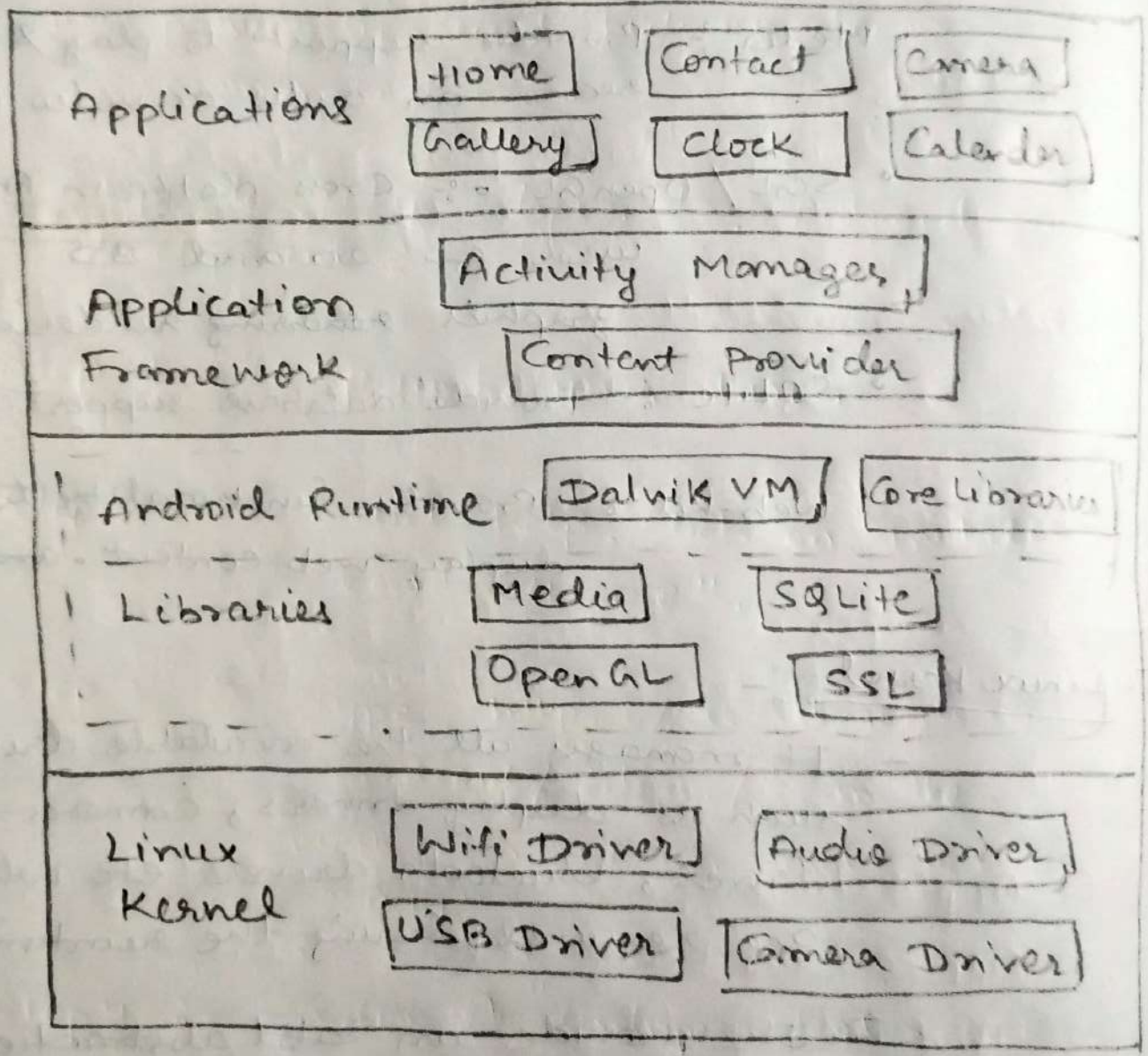
- Media :- Provides support to play & record an audio or video format.
- SGL / OpenGL :- Cross platform API used for 2D and 3D graphics ~~rendering~~ rendering.
- SQLite :- Provide database support.
- Web-kit :- Provides functionality to display web content. ~~and~~

5. Linux Kernel :-

- It manages all the available drivers such as display drivers, camera drives, bluetooth drivers etc which are required during the runtime.
- It provide an ~~add~~ abstraction layer between the device hardware and android app.

- Features of Linux Kernel :
  - Security
  - Memory Management
  - Process Management
  - Network Stack
  - Drive Model.

→



- **Installing Android SDK Tools**

→ An android SDK tool is a software component included in a software development kit (SDK) that provides resources and functionalities like debugging to develop an app.

→ An SDK is a set of tools, libraries and documentations that enables developers to create any applications.

→ Steps to install Android SDK tool :-

Step 1 :- Click Tools > SDK Manager.

Step 2 :- In the SDK Manager, a list of SDK tools, SDK platform and component will show, choose the SDK component you want to install.

Step 3 :- After selecting click the "Apply" or "OK" button to start the installation.

Step 4 :- Accept the licenses for some components. (if ask)

Step 5 :- The SDK Manager will download and install the selected SDK tool.

Step 6 :- Once installed, you'll see a Confirmation message.

- Configuring Android in Eclipse IDE.

→ Android studio is the official IDE for Android development, the eclipse was previously used but no longer recommended.

→ Steps to setup eclipse for Android development :

Step 1 :- Open Eclipse IDE.

Step 2 :- Select help, Click on "Install new Software".

Step 3 :- Type "https://dl-ssl.google.com/android/eclipse/" in the work with section and click on "Add".

Step 4 :- "developer tool" - check box will show, tick the box and click "next".

After that, installation will begin

Step 5 :- A' Restart Eclipse after installation.

Step 6 :- Go go Eclipse > Window > Preferences

~~(if dialog box doesn't appear automatically)~~

Step 7 :- In Android Preferences,
Browse the SDK file location of Android.
and Click "Apply".

- Generally :- (C:\Program Files\Android\android-sdk)

Step 8 :- After installation, go to
Eclipse > Window > Android Virtual
Device Manager

~~- fill the details as the form.~~

- Edit and fill the details and
Click "OK".

Step 9 :- To create an android application,
~~Select~~
File > New > Other > Android Application
project,
then Click on next.

Step 10 :- Eclipse IDE ~~s~~ setup is completed
for Android Application Development

- **Android Development Tools (ADT)**

→ ADT refers to the set of plugins that were available for the Eclipse IDE to facilitate Android app development.

→ ADT provide tools and features within Eclipse like visual layout editor, XML editors and tools for debugging and packaging Android apps.

→ Some of the ~~ADT~~ Android development Tools are:-

1. Android Studio
2. Android SDK
3. Gradle
4. Android Emulator
5. SDK Manager
6. SQLite Database Browser
7. AVD Manager.

• Creating Android Virtual devices (AVD)

→ AVD is an emulator that mimics the functionality of a physical android device, allowing developers to test and debug their applications without the need for actual hardware.

→ Steps to create an AVD :-

Step 1 :- Launch Android Studio

Step 2 :- Open AVD Manager.

- Go to Tools > AVD Manager.

Step 3 :- Create a New Virtual Device

- In the AVD Manager, click on "Create Virtual Device". Choose a hardware profile (eg Pixel 4) from the list and click "Next".

Step 4 :- Select a System Image.

- You can download also, then click "next".

Step 5 :- Customize the AVD settings, like RAM, orientation and name or more. then, click "Finish".

Step 6 :- Back in the AVD Manager; select the newly created AVD and click on the green "Play" button to launch it.

Step 7 :- The emulator will take some time to start.

Now, ~~You~~ we have successfully created and launched an Android virtual Device using Android Studio.

——X——

Unit - 02

- Creating First android application.

→

Brale

• Android application components

→ ~~These components of an android~~

~~application~~.

→ They are the essential building block of an android application that work together.

→ These main components are :-

1. Activities
2. Services
3. Content Provider
4. Brodcast Receivers

→ These components are loosely coupled by the application manifest file "AndroidManifest.xml" that describes each component of the application and how they interact.

1. Activities :-

- An activity represents a single screen with a user Interface (UI).
- It is the presentation layer of the app.
- It use fragments and views to set the proper layout.

public class MainActivity extends Activity {

}

2. Services :-

- Services runs in the background to perform long-running operations like play music or downloading files.
- It is used to perform long-running tasks or handling operations that don't require user interaction.
- It can continue running even if the user switches to another app or the device is locked.

public class MyService extends Service {

}

3. Content Provider :-

- It is used to supply data from one application to other on request.
- The requests are handled by the methods of the ContentResolver class.
- The data may be stored in the file system or in the SQL database.

- It provides set of methods i.e CRUD (Create, Read, Update, Delete) operations to manipulate or access the data.

public class MyContentProvider extends ContentProvider
{ - - - .

4. Broadcast Receivers :-

- It simply respond to broadcast messages from other applications or from the system.
- It doesn't have any UI it run in background.
- Allow components to react to events such as incoming SMS, Low battery etc.

# public class MyReceiver extends BroadcastReceiver{
public void onReceive(context, intent) {
- - -
}
}

- Example, Telegram use it to ~~get~~ handle the incoming call.

- **Understanding Activities**

→ Activities represent screens or user interfaces in an Android app.

→ Each activity handles a specific interaction or task within the app.

→ They are declared in the AndroidManifest.xml file.

→ It contains UI elements like buttons, images, videos or text fields.

→ One Activities can communicate with other activity by using Intents.

→ Activities have a lifecycle with methods like onCreate, onResume, onPause and onDestroy.

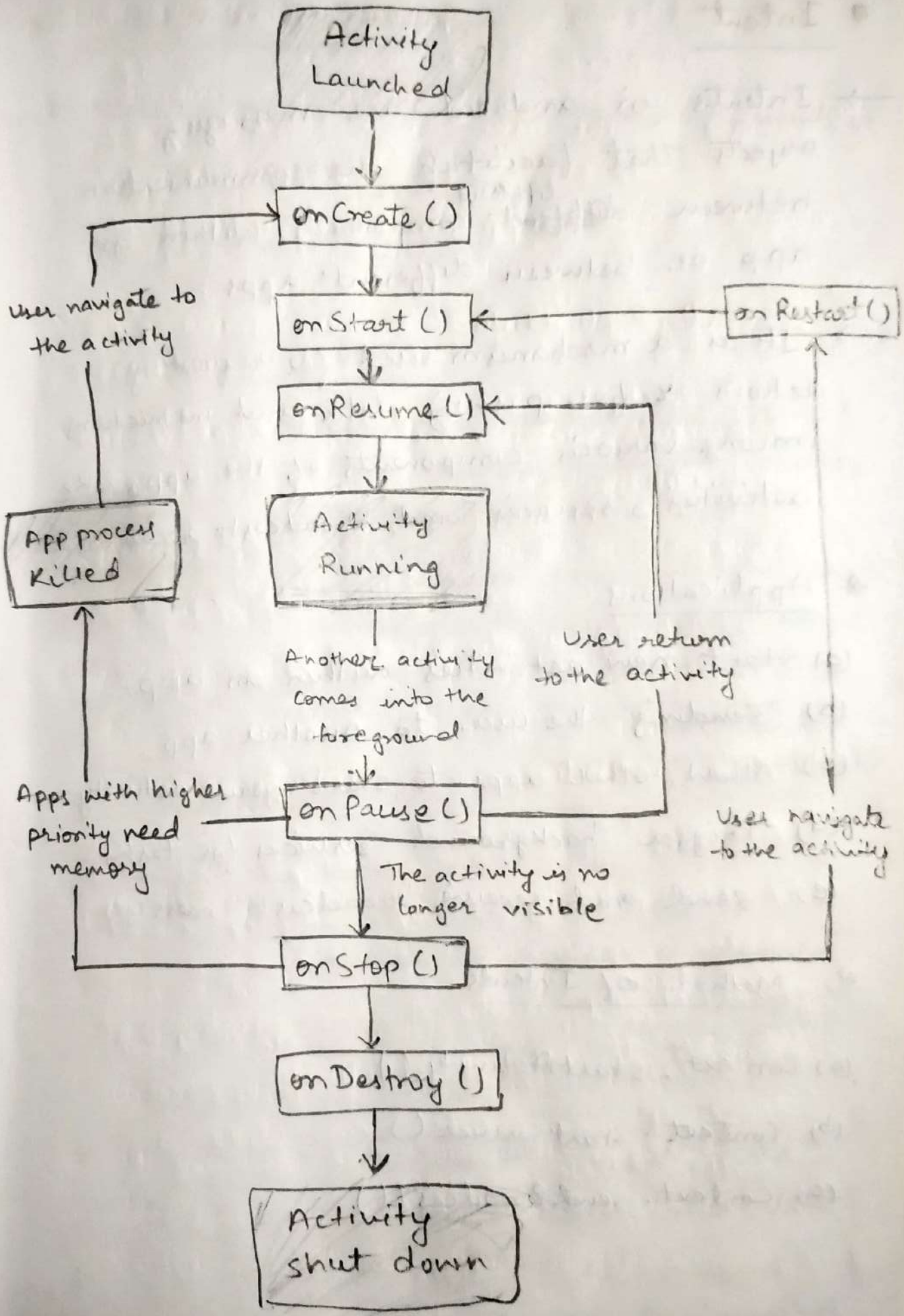
- **Activity life cycle.**

→ It refers to the sequence of states an activity goes through, from its creation to its destruction.

→ It consists of key methods that allow developers to manage the behavior of an activity at different stages.

→ The main stages includes :-

1. onCreate() :- Called when the activity is first created.
2. onStart() :- Called when the activity becomes visible to the user.
3. onResume() :- Called when the activity starts interacting with the user.
4. onPause() :- Called when the activity is ~~starts partially~~ going into the background but has not yet been killed.
5. onStop() :- Called when the activity is no longer visible to the user.
6. onDestroy() :- The final call received before the activity is destroyed.
7. onRestart() :- Called when the activity has been stopped and is about to be started again.


Activity Launched
onCreate()
onStart()
onRestart()
onResume()
Activity Running
User navigate to the activity
App process Killed
Another activity comes into the foreground
User return to the activity
Apps with higher priority need memory
onPause()
User navigate to the activity
The activity is no longer visible
onStop()
onDestroy()
Activity shut down

• Intent

→ Intents in android are messaging objects that facilitate the communication between ~~different~~ different components within an app or between different apps.

→ It is a mechanism used for requesting actions or for passing data and instructions among various components of the app, like activities, services and broadcast receivers.

* Applications

(a) Start new activities within an app.
(b) Sending the user to Another app.
(c) Allow other apps to start your activity.
(d) Trigger background services for tasks.
(e) Send and receive broadcast messages.

* Method of Intents

(a) Context.startActivity()
(b) Context.startService()
(c) Context.sendBroadcast()

- Types of Intent

→ There are mainly two types of Intents :-

1. Explicit Intent
2. Implicit Intent.

1). Explicit Intent :-

⇒ Used to navigate from one specific screen to another within the same app.

⇒ It require specifing the target components by its class name.

```
Intent ExplicitIntent = new Intent(Context,
                                   TargetActivity.class);
startActivity(ExplicitIntent);
```

2. Implicit Intent :-

⇒ Used to declare a desired action without specifying the target component

⇒ It identify the component based on the intent's action and data.

⇒ Instead of specifying a particular app to handled

the sharing action, we use an implicit intent to choose any app.

Example :- The user can choose to share data on different apps like whatsapp, facebook, message etc.

~~Intent explicitIntent = new Intent(Intent.~~
~~ACTION_VIEW),~~

Uri webpage = Uri.parse("https://www.google.com");

Intent implicitIntent = new Intent(Intent.ACTION_VIEW, webpage);

startActivity(implicitIntent);

- Linking activities using Intents

→ Assume you have multiple activities as MainActivity, FirstActivity or SecondActivity.

→ MainActivity is the starting point, that navigate to the FirstActivity when some specific event occur like a button click by user.

You can also transition from FirstActivity to SecondActivity.

1) Create intent in MainActivity :-

```
Intent intent = new Intent (MainActivity.this,
                            FirstActivity.class);
// To send data
intent.putExtra ("Key", "value");
startActivity (intent);
```

2) Retrieve data in the FirstActivity :-

```
Intent intent = getIntent();
String data = intent.getStringExtra ("Key");
```

3.) Declare Activity in the Manifest file

→ Ensure both activity are declared in the "AndroidManifest.xml" file.

```
<application>
    <activity android:name = ".MainActivity">
    ....
    </activity>
    <activity android:name = ".FirstActivity">
    </activity>
</application>
```

—X—

Unit - 03

- **Fragments** (sub-activity)

→ Fragments are the modular components of an activity that represents a portion of the user interface (UI).

→ Fragments can exist only inside an activity.

→ They are used to create more flexible and reusable UI design, especially for large screen or to create multi-panel layout.

→ Fragments have their own lifecycle similar to activity.

→ Inside activity.xml file, Fragment can be added by using <fragment> tag.

→ Used to design layout for both portrait and landscape orientation.

→ The FragmentManger class is used to make interaction between fragment objects.

→ Like Activity, they have both java and xml file.

→ Example, whatsapp has ~~three~~ four fragments for community, chat, Updates, and Calls.

* Types of Android fragments

1. Single fragment :- Display only one single view on the device screen, common for mobile phones

2. List fragment :- Used to display a list-view from which the user can select the desired sub-activity or fragment.

3. Fragment transcaction :- Support transition from one fragment to another at run time.

• Fragment life cycle

→ Each fragment has it's own lifecycle but sometimes due to the connection with the Activity, its lifecycle is influenced by the activity lifecycle.

1. onAttach () :- Called when the fragment is attached to an activity.

2. onCreate () :- It initializes the fragment by adding all the required attributes and components.

3. onCreate View () :- System calls this method to create the User Interface (UI) of the fragment

4. onView Created () :- Indicates that the activity has been created in which the fragments exists.

5. onStart () :- It make the fragment visible on the user's device

6. onResume () :- Called when the fragment is interacting with the user.

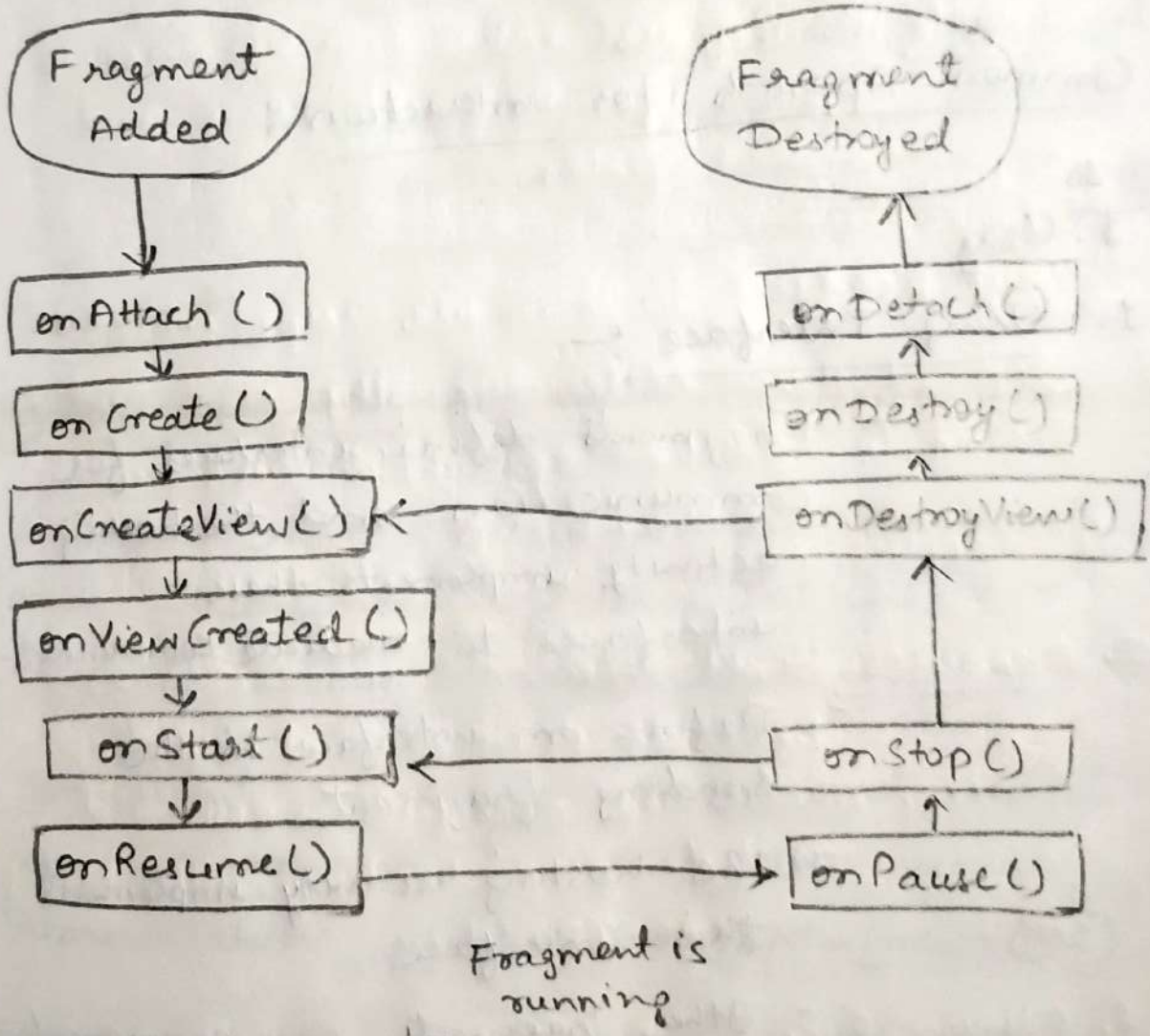
7. onPause () :- It indicates that the user left the fragment.

8. onStop () :- Method to terminate the functioning and visibility of fragment from the user's screen.

9. onDestroyView() :- Called to clean up all kinds of resources and UI of the fragment.

10. onDestroy() :- Final clean up of fragment's state and its lifecycle

11. onDetach() :- Indicates that the fragment is being detached from its hosting activity.



- **Interaction between fragments**

→ The hosting activity facilitates the interaction or communication between the fragments indirectly.

→ Fragments can send data to their hosting activity and then the hosting activity send that data to other fragments.

* **Common Approch for interaction :**

1. **Using Interface :-**
   - Fragments define interfaces for communication and the hosting activity implements these interfaces to mediate communi^.
   - Define an interface in the sending fragment.
   - The hosting activity implements this interface.
   - Other fragments can communicate indirectly by calling methods in the hosting activity.

2. Using ViewModel :-

- Use a shared viewModel between fragments.
- Fragments can observe changes in the shared 'View Model' and react according to it.

3. Using EventBus :-

- Use EventBus libaries (like, Green Robot's, EventBus) that provides a publish-subscribe pattern for communication between fragments.

- Layout
- ~~Understanding the components of a screen (Layout)~~.

→ A layout in Android represents the structure and arrangement of user Interface (UI) elements within an application screen or activity.

→ Layout defines how views and widgets are organized to create the visual representation of the user Interface (UI).

→ Helps in organizing UI Elements such as buttons, textviews, images etc.

- **Components of layout**

1. **Views** :- They are the basic building blocks of a layout, used to draw elements such as buttons, text fields and images.

2. **ViewGroups** :- Containers that hold and organize multiple views or other viewGroups.

3. **Widgets** :- They are interactive UI elements such as buttons, checkboxes and text fields.

4. **Layout Managers** :- They are responsible for determing the size and position of child views within a View Group.

5. **Attributes** :- It define the appearance and behavior of views and layouts. These are specified in XML and include properties like width, height margin & padding.

6. **Resource Files** :- Define the properties of layout stored in 'res/layout' directory as layout.xml.

- Types of Layout

1) Linear Layout :-

- It arrange the views or child layout in a single line, either horizontally or vertically.
- android : orientation = "horizontal / vertical"
  - In xml -

2. Relative Layout :-

- Position child view relative to each other or to the parent layout.
- android : layout_width = "match_parent"

3. Constraint Layout :-

- Allow positioning and sizing of childviews using constraints.

4. Grid Layout :-

- Organizes views in a grid, with rows and column.

5. Scroll view :-

- Allow for scrollable view, typically user to fit too large content.

6. List View :-

- Displays a scrollable list of items.
- Efficient for handling large datasets.

- **Adapting to display orientation.**

→ To handle layout for different orientation Create different layout resources for landscape and portrait mode and implement them according to the orientation.

→ Create two xml. files.

res / layout / activity_main.xml
res / layout - Landscape / activity_main.xml

- **Action Bar**

→ The Action Bar is a UI element at the top of the screen, typically used to navigate ~~and~~ in app.

code: <application

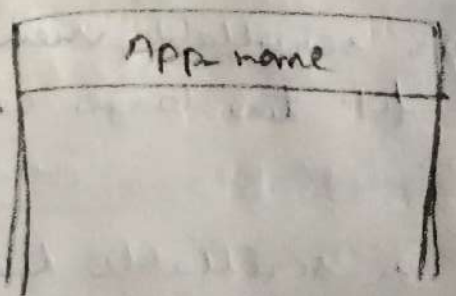
. . . .

android:label = "@string/app_name"

. . . .

</application>

o/p :-



• Views (UI Widgets)

→ UI widgets in android are reusable components that allow users to interact with the application.

→ These widgets are defined in XML layout files and can be customized with attributes to control apperance and behavior.

→ Some of the UI widgets are:-

1) **Button** :- A clickable UI element that triggers an action when pressed.

In xml file :-

```
<Button
    android:id="@+id/myButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click me!" />
```

In Java :-

Event Listener handles the button click event.

```
myButton.setOnClickListener();
```

2) **Toast** :- It is a small pop-up message that appears temporarily, usually to display a brief notification.

In Java :-

```
Toast.makeText(context, "Hello, Toast!", Toast
        Toast.LENGTH_SHORT).show();
```

3) Toggle Button :- It is a two-state button that can be toggled on or off.

In xml :-

```
<ToggleButton
    android:id="@+id/toggleButton"
android:layout_width="wrap-content"
android:layout_height="wrap-content"
android:text="Toggle me!" />
```

4) Check Box :- It allows users to select or deselect options independently.

In xml :

```
<CheckBox
    android:id="@+id/checkBox"
    android:text="Check me" />
```

5) Radio Button :- They are used when users need to select only one option from a set of choices.

6) **Spinner** :- A spinner displays a drop-down list from which users can select an item.

In xml :-

```
<Spinner
    android:id = "@+id/spinner"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content" />
```

7) **Web View** :- It is a UI component that displays web content, allowing us to embed web pages within our android app.

In xml :-

```
<WebView
    android:id = "+@+id/WebView"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent" />
```

8) **Edit Text** :- It allows users to input text, commonly used for forms and text entry by users.

9) **Date Picker** :- It allows users to select a data from the calendar.

10) **Time Picker** :- It allows users to select a specific time.

11) List View :- It displays a scrollable list of items, commonly used to present data in a vertical list.

12. Progress Bar :- It shows the progress of an ongoing operation. It can be determinate or ~~indeterminate~~ indeterminate.

13. Analog and Digital Clock :- Analog clock shows time with rotating hand and digital clock for precise time.

- **Handling UI events**

→ Event handling refers to the process of managing and responding to events that are triggered by user interactions or system actions.

→ In the context of Android development, event handling involves responding to events generated by user interactions with the help of user interface (UI) elements. like

⇒ When a user click on UI element (button), then the event is handle by onclick event Listener.

* Some event Listeners and their respective event handlers are :-

1) **OnClickListener ()** :-

- method called when the user click, touches or focuses on any view lik Button, Image etc.
- Handled by eventhandler onclick ().

2) OnLongClickListener () :-

- Responds to long-click event, on UI components, such as holding down a button
- Event handler onLongClick().

3) OnMenuItemClickListener () :-

- Called when menu item is selected,
- Event handler onMenuItemClick().

4) OnTouch () :-

- Called For movement gesture on screen, press, & release or moving.
- Event handler onTouch

Example :

```
Button myButton = findViewById(R.id.myButton);
myButton.setOnClickListener(new View.OnClickListener()
    {
        @Override
        public void onClick(View v) {
            Toast.makeText(getApplicationContext(),
                # Handle the click"
        }
});
```

- **List fragment**

→ A list fragment is a specialized fragment that displays a list of items.

→ It is useful when we want to present a scrollable list of items without the need to create a custom layout.

→ It is a subclass of the Fragment class.

public class MyListfragment extents listfragment{

}

→ Provides a built-In user interface (UI) for displaying lists.

→ Requires an adapter to bind data to the list, commonly used adapters include ArrayAdapter, CursorAdapter or Custom Adapter

→ Comes with a default layout that includes a ListView for displaying the list.

→ Supports item click callbacks through 'onListItemClick()' method

Dialog Fragment

→ It is a specialized fragment used to display a dialog window on top of an existing activity like a popup.

→ It helps to manage the Creation and presentation of dialogs.

→ It is a subclass of the Fragment class.

public class MyDialogFragment extends DialogFragment {

}

→ It also supports user interaction with the dialog, such as button clicks or choosing an option.

→ It can be used to create various types of dialogs, including alerts, pickers and custom dialogs with the help of 'onCreateDialog' method.

→ DialogFragment inherits the theme from its parent activity but can be customized with specific themes.

— X —

Unit - 05

- **Data storage Options**

→ Storing data in Android application is essential for preserving user preferences (setting or choice made by user), caching data or saving user-generated content.

→ Here are the main data storage options available in Android :

1. Preferences
2. Internal storage
3. External storage
4. Content Provider
5. SQLite Database

1.) **Shared Preference** :-

⇒ It allows us to store primitive data types (such as strings, integers, booleans) in key-value pairs.

⇒ Ideal for storing small amounts of simple data like user settings or preferences.

⇒ Private access, only accessible by app that created it.

→ Stored in app's xml file directory.

Example : Storing user preferences like theme settings, language preference or user-specific settings.

2) Internal Storage :-

→ It stores private data on the device's internal memory.

→ Suitable for storing sensitive data that should not be accessible by other apps or users.

→ But there is limited space, not recommended for large files.

→ Store in app's private Internal Directory.

Example :- Saving user-generated content, application database or cached data. like app is downloading a pdf or storing an image for video

3). External Storage :-

⇒ Stores data on the device's external storage, such as an SD card.

⇒ Used for storing large files or data that can be shared with other apps or accessed by the user.

⇒ Accessed using file I/O operations similar to the internal storage.

⇒ Readable / Writable by other apps

⇒ Requires appropriate permissions in the app manifest to access the data.

** They are mainly of two types :-

(a) Primary ES :- Inbuilt shared storage.
(b) Secondary ES :- Removable storage device

⇒ The user can keep the data even after uninstalling the app.

Example :- Store media files, documents or files.

4) Content Provider :-

→ It provides a structured interface for accessing and sharing data from one app to other on request.

→ Such request are handled by the methods of the Content Resolver class.

→ It defines a unique Content URL (Uniform Resource Identifier) to identify the requested data.

→ It behave like databases, supporting operations like query, insert, update and ~~dat~~ delete.

→ Commonly Content Providers use SQLite database for data storage.

5) SQLite Database :-

→ SQLite database is an open-source database provided in Android that can be used to store data inside the user's device in the form of a Text file.

→ SQLite is an offline database, that is it doesn't require a separate server.

→ The Android SDK includes a built-in SQLite database library that simplifies the database operations in the Android app.

→ No need to establish connection to JDBC drivers or server

- Connecting with SQLite database

→ Connecting with SQLite database involves creating a database helper class, managing database operations and utilizing 'SQLiteOpenHelper' class.

→ SQLiteOpenHelper class helps in creating, upgrading and managing the database. It defines the database name, version, and schema (structure) in the 'onCreate()' method.

→ Obtain a writable database instance using "getWritableDatabase()", that provides a connection to the database that allows both reading and writing operations.

→ SQLite is an offline database, that is it doesn't require a separate server.

→ The Android SDK includes a built-in SQLite database library that simplifies the database operations in the Android app.

→ No need to establish connection to JDBC drivers or server

- Connecting with SQLite database

→ Connecting with SQLite database involves creating a database helper class, managing database operations and utilizing 'SQLiteOpenHelper' class.

→ SQLiteOpenHelper class helps in creating, upgrading and managing the database. It defines the database name, version, and schema (structure) in the 'onCreate()' method.

→ Obtain a writable database instance using "getWritableDatabase()", that provides a connection to the database that allows both reading and writing operations.

Code :

```
// Step 1 : Create a database Helper class

public class MyDatabaseHelper extends SQLiteOpenHelper {

    Private static final String Database_name
                    = " my-database";
    private static final int Database_version = 1;
    // Constructor
    public MyDatabaseHelper (Context context) {
        super(context, Database_name, null, Database_
                                            version);
    }

    @ Override
    public void onCreate (SQLiteDatabase db) {
        // Create tables and define schema
        String createTableQuery
                    = " CREATE TABLE my_table
                    (id INTEGER Primary KEY,
                    name TEXT,
                    age INTEGER); ";

        db.execSQL (createTableQuery);
    }
```

```
@Override
public void onUpgrade(SQLiteDatabase db,
                      int oldVersion, int newVersion){
    // Handle database upgrades if needed
    }
}
```

// Step 2 : Get Writable Database

```
MyDatabaseHelper dbHelper = new MyDatabase
                            MyDatabaseHelper(context);
SQLiteDatabase db = dbHelper.getWritableDatabase();
```

- Operations

1) Insert :-

```
ContentValues values = new ContentValues();

values.put("name","Harsh")
values.put("age", 25);

Long newRowId
long newRowId = db.insert("my_table", null,
                          values);
```

2.) Delete :-

```
//Delete rows where the "name" is "Harsh"
int rowsDeleted = db.delete("my_table",
        "name = ?", new String[]{"Harsh"});
```

3) Update :-

```
ContentValues updatedValues = new ContentValues();
updatedValues.put("age", 26);
// Update row with name "Harsh"
int rowsUpdated = db.Update("my_table",
        updatedValues, "name = ?",
        new String[]{"Harsh"});
```

4) Fetch :-

```
Cursor cursor = db.query("my_table", null,
        null, null, null, null, null);
while (cursor.moveToNext()) {
    int id = cursor.getInt(cursor.getColumnIndex
            ("_id"));
    String name = cursor.getString(Cursor.
            getColumnIndex("name"));
```

~~int age = cursor.getInt(~~

}

Cursor.close();

- **Publishing Android application**

1. Prepare Your App.
2. Create a Developer Account. ($25 required)
3. Set Up Your Google Play Console.
4. Upload Your App Bundle or APK.
5. Configure Store Listing.
6. Set Up Monetization,
7. Define Content Ratings.
8. Define Testing and Release Tracks.
9. Publish Your APP.
10. Promote Your App.
11. Monitor and Update.

- **Apk file**

→ APK stands for Android Package.

→ It is a file format used by the Android operating system for the distribution and installation of mobile apps.

→ It is the standard format with .apk extension.

* The main types of APK files are:

1. **Debug APK** :- Used during the development and debugging phase.
2. **Release APK** :- Used for distribution to the end-users.
3. **Signed APK** :- Required for installation on Android devices.
4. **UnSigned APK** :- Typically used for testing purpose.

——X——