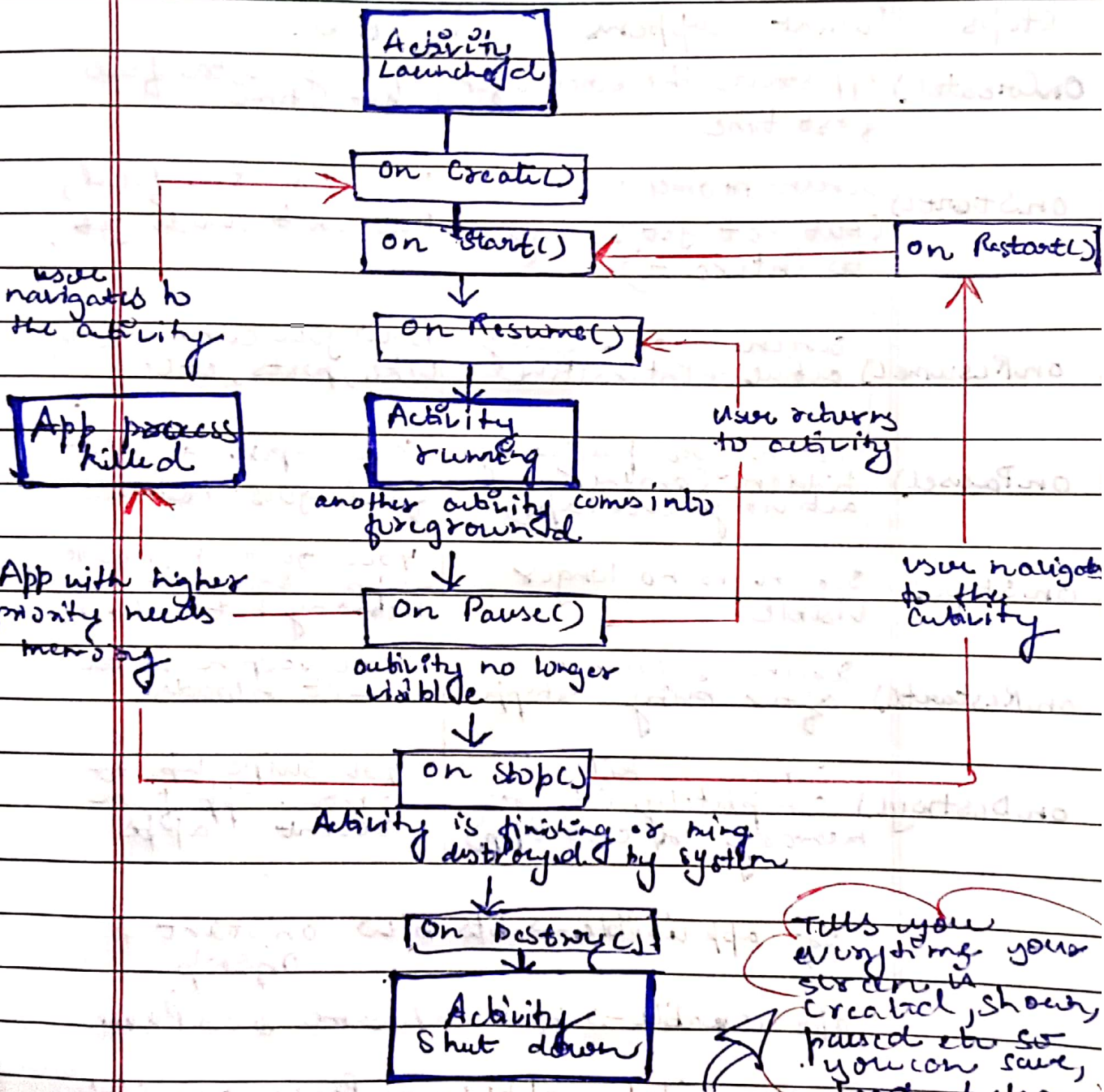
Date / /
Page No

App Dev.



Android Activity Life Cycle ?

means :- Every screen in an Android app is called 'an Activity'

Life Cycle :- how that screen is created, shown, paused, stopped or destroyed as you use app.

Lifecycle steps.

Steps	what happens	Ex.
onCreate()	App starts the screen for first time	opening insta feed 1st time
onStart()	screen becomes visible (but not yet ready to interact)	you can see feed, but can't scroll yet.
onResume()	screen becomes fully active + interactive	now you can scroll, like, posts, etc.
onPause()	screen goes partially hidden (another activity overlaps)	You open a DM - feed goes behind
onStop()	screen is no longer visible	You go to homescreen - app stays in memory but hidden
onRestart()	screen is coming back after being stopped	You reopen insta - it reloads
onDestroy()	screen is killed completely (to free memory or exit app)	You swipe up to remove app from recents apps

If app visible → b/w onStart & onStop

app usable → b/w onResume & onPause

user switch apps :- onPause → onStop

app crashes or closed → onDestroy

cases

Save data in onPause()

Release heavy resources in onStop()

Initialising things onCreate()

Date / /
Page No

What is an Intent?

⇒ A message you send in Android to do something like "Hey system, open this page", or "start this activity".

2 types:-

1) **Explicit**:- "You know exactly which component (Activity/service) to start.

ex Intent i = new Intent(this, SecondActivity.class);

⇒ used within your own app.

★ "call my friend directly - I know their no.".

2) **Implicit**:- You don't know who will handle it - system decides which app can do it.

ex Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse("https://google.com"));

⇒ used to open camera, gallery, browser etc.

★ "shout out - anyone who can open this link, please do".

Extras (Data u send):-

• putExtra("key", value) ⇒ Passes small data (like name, no.).

ex i.putExtra("username", "Ayushi");
startActivity(i);

retrieve with • getStringExtra("username");

(I used think "Attach a sticky note to delivery")

Date / /
Page No.

Android debugging Tool

LOGCAT ⇒ when something breaks in app Logcat tells u why?

use:- Dev use to understand the error instead of giving them

ex Val name = null
println(name.length) // we will be then app crash

Logcat will show:-

NullPointerException: Attempt to read length of a null object

∴ We know issue ⇒ Name was null

Errors: for crashes

Debug: messages printed with Log.d()

Info: general info

* If u want to print own message for debugging:-

Log.d("AyushiApp", "Button clicked!")

Logcat will show:- syntax
Log.d("Tag", "message")

D/AyushiApp: Button clicked!

Date / /
Page No

DATABASE

Steps

(1) Add Dependency → Initialise DB

Val database = FirebaseDatabase.getInstance()
Val ref = database.getReference("users")

think: "Pointing to my storage box for users".

(2) Declare Variables (lateinit)

↳ For things initialised later (EditText, DB reference)

lateinit var userName: EditText
lateinit var userAge: EditText
Lateinit var database: DatabaseReference

userName = findViewById(R.id.editUserName)
userAge = findViewById(R.id.editUserAge)
database = FirebaseDatabase.getInstance().getReference("users")

☆ lateinit = promise to initialize before use
Initialize later (not while writing code)

(3) Data class / object — ≈ DB str.

↳ template for your data

data class User(var Name: String = "", var age: Int = 0)
Val user = User(userName.text.toString(), userAge.text.toString().toInt())

④ CRUD Ops :-

Ops	Meaning	Example.
Create	Add data	ref.child("userId1").setValue(user)
Read	Get data.	ref.child("userId1").get(). addOnSuccessListener{...}
Update	Edit data	.child("age").setValue(25)
Delete	Remove data	.removeValue().

Tip: Push unique keys → No overwriting

Flowchart

[App started]
↓
[connect to DB]
↓
[Declare variables (lateinit)]
↓
[create Data object]
↓
[choose Op: CRUD]
↓
[DB executes operation]
↓
[Success / Fail: callback]
↓
[update UI / Notify user]

TRICKS :-

- CRUD = Add, View, edit, Remove
- Data class = blueprint of DB obj
- Firebase is async → always use callbacks
- lateinit avoids null errors
- Push unique IDs for each entry.

Date / /
Page No

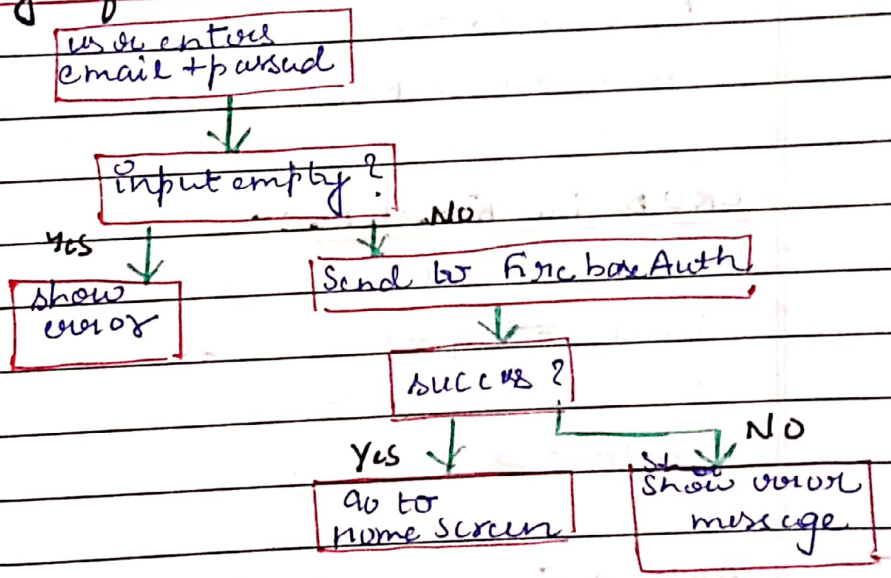
App structure Diagram:-

UI (screens) ⇄ UI shows → Logic decides
↑↓ ↓
Logic DB stores.
↑↓
DB (Firebase)

Firebase str.:-

[users] ← collection
└ [user ID 123] ← Document
name: Ayushi
age: 18
city: Lucknow

Logic flow (Firebase Authentication):-



Reading Data from Firebase:-

App Opens → call Firestore → Firestore sends data
↓
Logic updates variables
↓
UI shows List

Date / /
Page No.

Logic :— (brain of App)

→ connects UI ↔ database

Uses if/else, functions, checks

Steps :- Validate → Process → Store/Fetch → Show UI

Keywords: lateinit, val/var, functions, state

Firebase :—

Login flow = input → validate → Firebase
↓
success / Fail

Analogy { Firebase Firestore }

collection = folder

Document = file

Field = data

users → user ID → name, age, mail

CRUD in Firestore :-

create → add()

Read → get()

Update → update()

Delete → delete()

App → Firebase Flow :-

1. User clicks
2. Logic checks
3. Firebase called
4. DB updated/fetched
5. UI updates

Date / /
Page No

ListView ⇒ A UI component that shows a vertical list of items (like contact list).

How it works?
-) We give data (array / list)
-) It shows one item per row
-) Needs Adapter to connect data → UI

Tips :- -) Don't put images, heavy layout inside list view (it lags)
-) Use for very simple lists. Otherwise → RecyclerView

ADAPTERS :- (The 'Bridge')

What it is : Middle man between data & UI

Ex : You have a list of fruits → Adapter → ListView/RecyclerView displays them

Simple Meanings :-

Without Adapter → the list will never show

Tips :- -) Adapter decides HOW each list item looks.
-) Keep item layout simple for smooth scroll

RECYCLER VIEW — (Advanced ListView)

What it is : improved, faster, more powerful version of ListView.

Used for insta feed, YT list, charts etc.

Why is it better : -) Recycle old views = super fast
-) support grids, horizontal list etc.
-) works with ViewHolder, so no lag.

Tips :- -) Make ViewHolder light weight

Date / /
Page No

API (Application Programming Interface)
↳ Getting Data from Internet.

What it is?
A way for your app to talk to a server and get data.

Ex:- App → says "Give me weather details"
API → sends data back in JSON form
{} → obj [] list

Tips:- •) Always check "internet permission"
•) Use libraries like "Retrofit" or "Volley" for easy handling.
•) Handle no-internet errors gracefully.

NAVIGATION

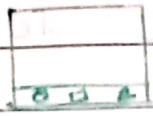
What it is?
Handles moving between screens (Fragments)

Why it's useful:-
•) No need to write manually fragment transactions.
•) Built-in animations
•) Safe to handle back stack

Tips:- •) use 'NavGraph' to visualize screens
•) Use 'safe args' to pass data b/w fragments

Date / /
Page No

BOTTOM NAVIGATION VIEW :-



What it is ?

The bottom bar with 3-5 icons (Home, search etc)

Uses :-

Quick navigation b/w major app sections

Tips :-

- Ideal for apps with 3-5 main features
- Keep icons simple + meaningful
- Do NOT overcrowd it

FRAGMENTS – (Screens Inside an Activity)

What it is ?

A mini screen inside an activity.

Helps create multiple screens without multiple activities

Ex :- Home fragment, Profile fragment
All inside 1 activity.

Tips :

- Use ~~not~~ fragments with Navigation component.
- Avoid too much logic inside fragment ↓
  Keep it in ViewModel.

Date / /

Page No.

NAVIGATION DRAWER
(side Menus)

what it is ?

The left-side sliding panel

Use case!

When app has many categories or settings

Tips :-

- Don't put more than 7-8 items
- Keep imp items at top