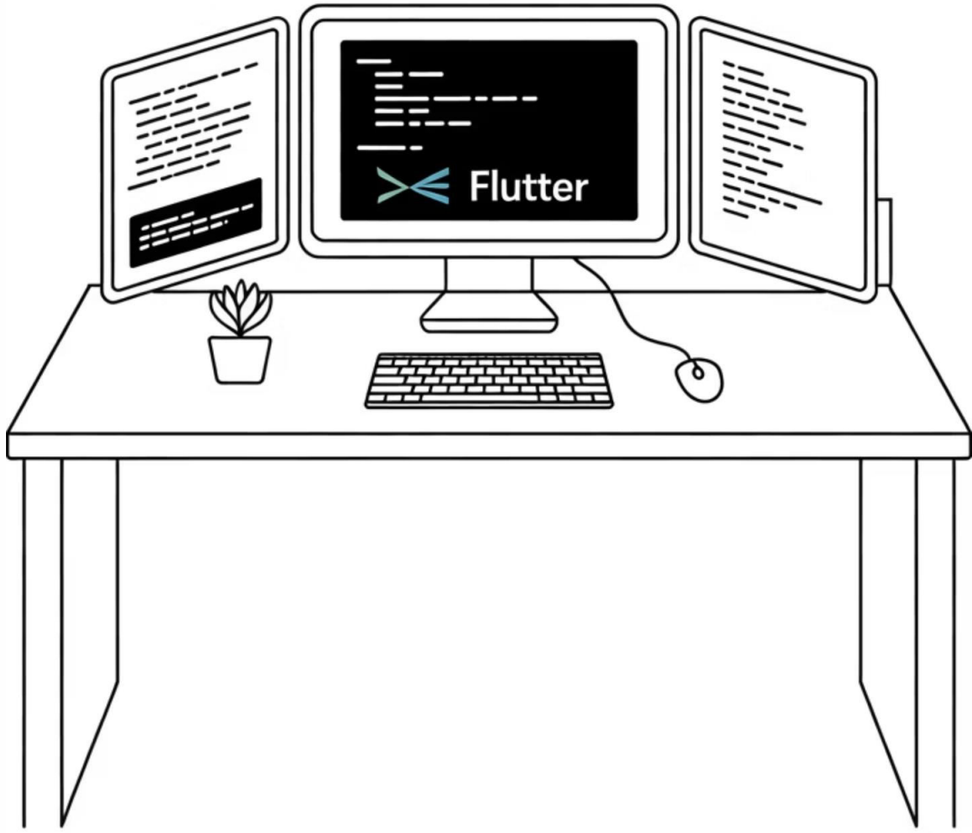
Welcome Everyone!

Class Topics:

- **What is Flutter?**
- **Why do companies choose Flutter?**
- **What is Computer**
- **History of Computer**
- **How a computer works**
- **What is RAM?**
- **What is ROM?**
- এগুলা জেনে কী হবে?

Why Start With Computers?

Before diving into Flutter development, we need to understand the foundation - how computers actually work. This knowledge will make you a better developer and help you avoid common pitfalls.

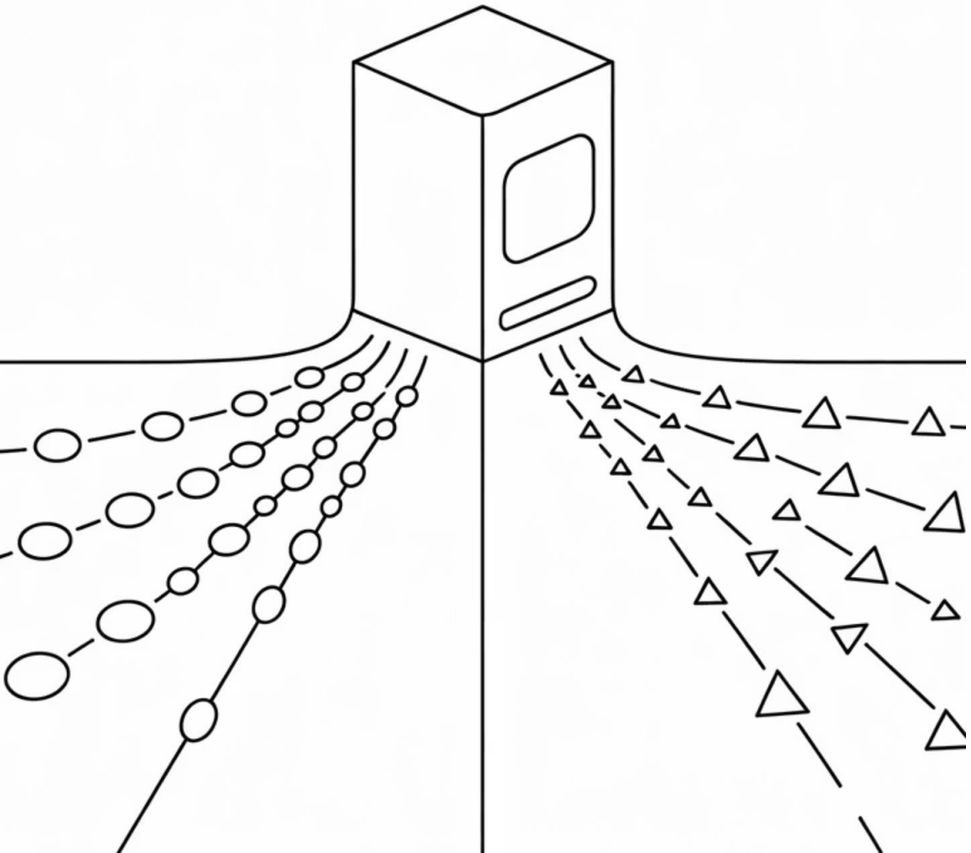


Imagine trying to build a skyscraper without knowing what bricks are. 💡

Same with Flutter – before coding apps, you must know how computers think.

Flutter runs on computers, mobiles, devices – let's peek inside their brains first.

Story: "When I first learned Flutter, I ignored basics. Later, one bug took me 3 days to fix because I didn't understand memory properly. Don't repeat my mistake!"



What is a Computer? (Simple Definition)

1. Accepts input
2. Processes
3. Produces output
4. Stores it

A computer = *A smart servant* (takes orders, gives results).

Not "magic", just fast calculations + storage.

Interactive: Ask students: "If your brain is a computer, what's your RAM? What's your ROM?"

A Quick History of Computers

1st Generation: Big, heavy, vacuum tubes. — 1

2 — 2nd Gen: Transistors made computers smaller.

3rd Gen: Integrated circuits (ICs) – faster, cheaper. — 3

4 — 4th Gen: Microprocessors – personal computers.

5th Gen: Today – AI, smartphones, Flutter apps! — 5

Story: "My dad had a PC with floppy disks. That thing took 5 minutes to start. Now, I open Flutter and my emulator runs in seconds. That's progress!"

Why History Matters for Flutter Devs?

Shows how hardware limitations shaped software.

Explains why we care about performance in apps.

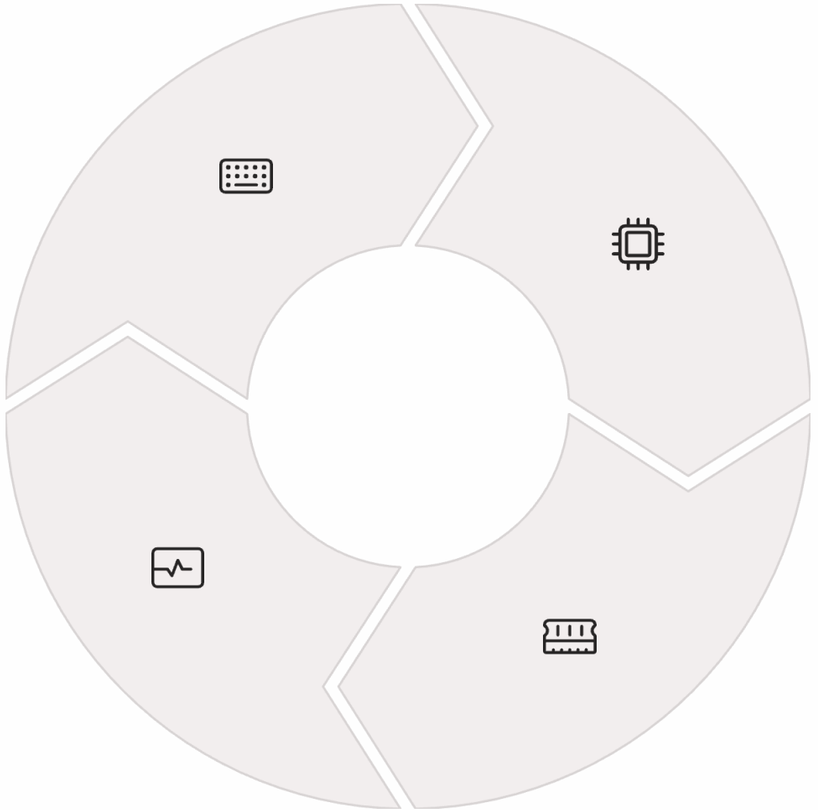
Flutter apps run on small devices → optimization is key.

Example: Old computers had 64KB memory. Today's phones have 8GB RAM. That's why Flutter apps can look smooth like native apps.

How a Computer Works (The Cycle)

Input: Keyboard, touch, voice.

Process: CPU (the brain).



Output: Screen, speakers.

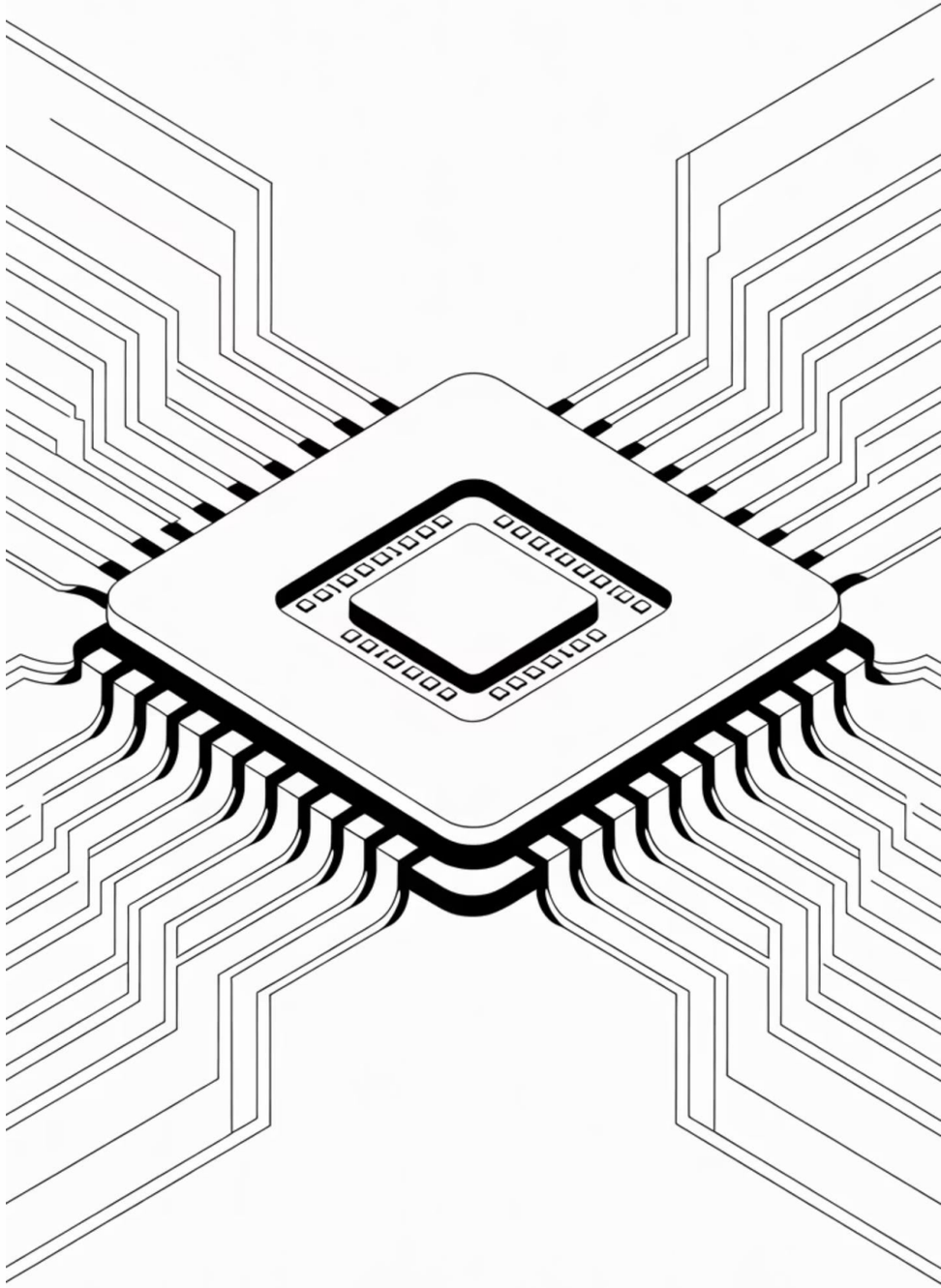
Memory: RAM/ROM.

Interactive: Draw the cycle on board. Ask students: "When you tap a button in Flutter, which part of the cycle is triggered?"

CPU – The Brain of Computer

- Executes instructions line by line.
- Works with RAM closely.
- Speed measured in GHz (billions of steps per second).

Story: "One of my first apps ran super slow. I blamed Flutter. Later I realized, my old laptop's CPU was just too weak. Hardware matters!"



RAM – The Short-Term Memory

Temporary memory, clears after shutdown.

Stores data while app is running.

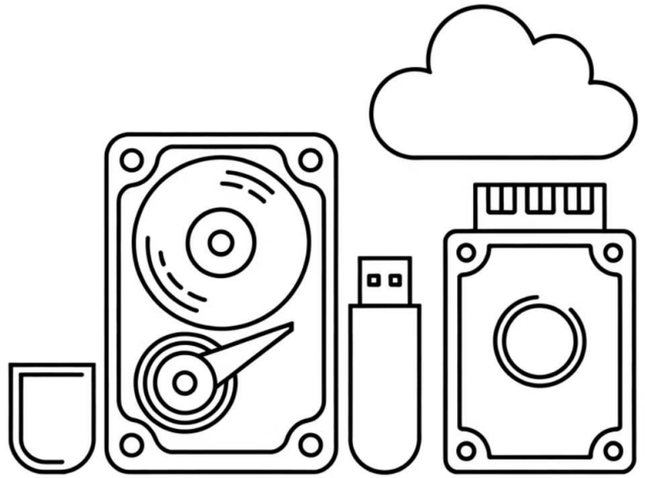
More RAM = more smooth multitasking.

Flutter Relevance: When you open Android Emulator + VS Code + Flutter project → RAM usage spikes. That's why low-RAM PCs lag badly.

ROM – The Long-Term Memory

- Permanent storage (doesn't vanish after restart).
- Stores OS, system files.
- Without ROM → no booting.

Flutter Relevance: When you install Flutter SDK, Android Studio, Emulator → All stored in ROM (your hard drive).



Why RAM & ROM Matter in Flutter

Flutter apps use RAM during hot reloads.

Flutter SDK + Emulator live in ROM.

Knowing this saves you from silly issues.

Example: Students often say "My emulator is not running!" → 90% of time, low RAM or storage issue.

Wrap-Up & Takeaway

- Computer = smart servant (input → process → output).
- History shows why performance matters.
- CPU, RAM, ROM = the real stage where Flutter performs.
- Learn basics → Build apps without fear.

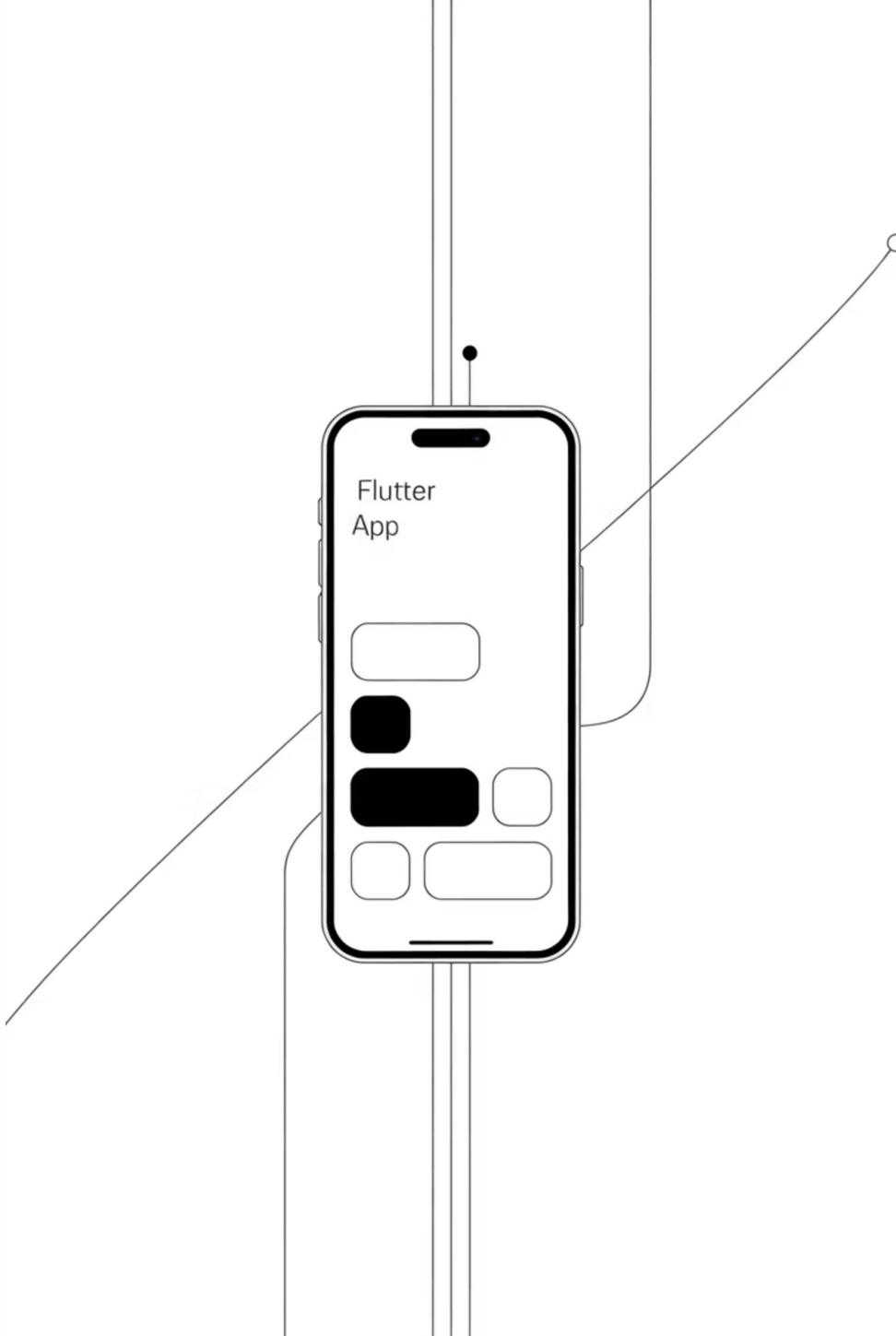
Interactive Ending: Ask: "Next time your emulator crashes, what will you check first: RAM or ROM?"

Class Topics:

- **Basic Networking**
- **Basic Operating System**
- **How a mobile device works**
- এগুলো জেনে কী হবে? কেনো শিখছি?

Why Networking, OS & Mobiles?

Understanding the foundation that powers your Flutter applications



Why Networking, OS & Mobiles?

- You want to build Flutter apps → Where will they run? On mobiles!
- Who controls mobiles? Operating Systems!
- How do apps talk to each other? Networking!

> Story: "When I made my first Flutter chat app, it worked perfectly… on my phone only. Didn't send messages to others. Why? I didn't understand networking basics."

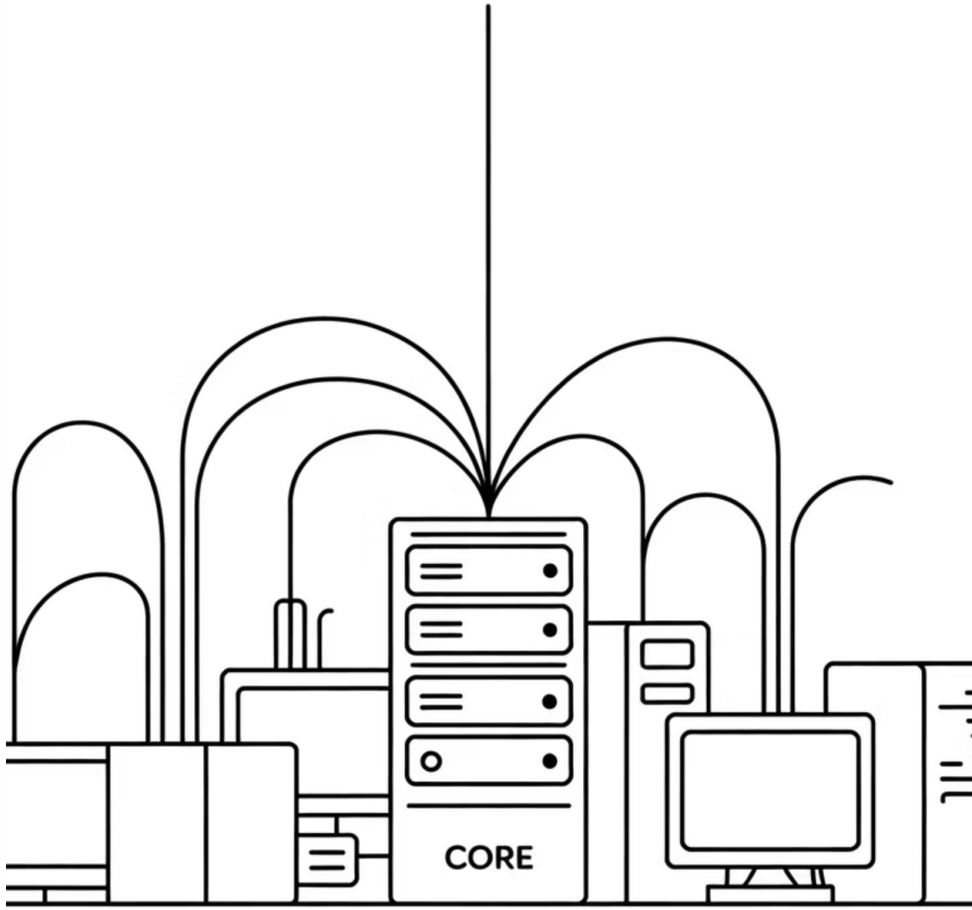
What is Networking (Simple)

Networking = devices talking to each other

Like humans need language, computers need protocols

Example: WhatsApp → You send text, it travels through network → Shows up on friend's phone

Interactive: Ask: "When you press LIKE on Facebook, where does that info go first?"



Networking Basics for Flutter Devs

IP Address

IP address = unique identity (like your house number)

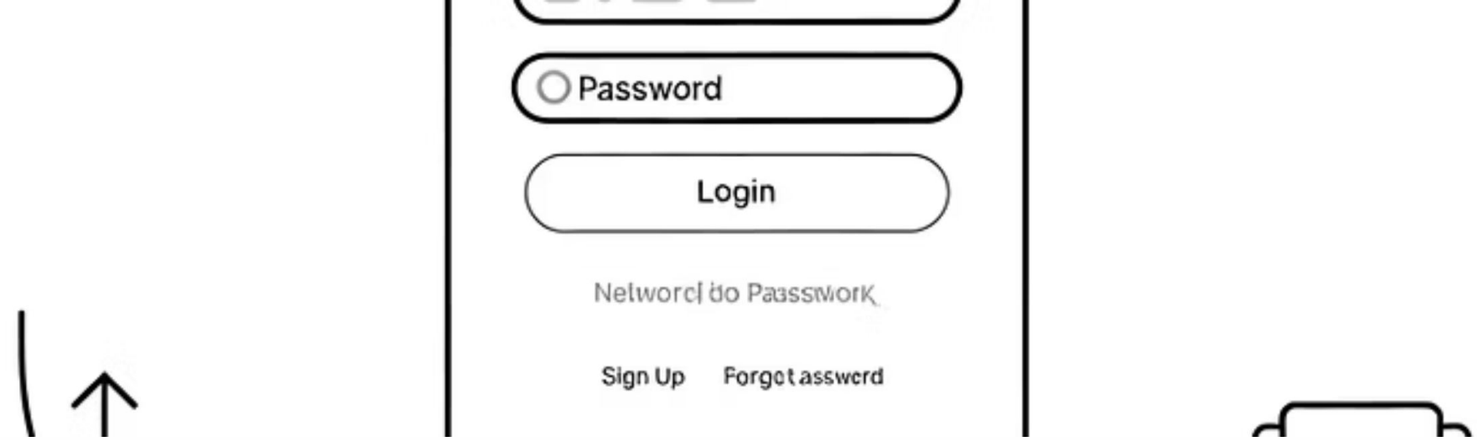
Protocols

Protocols (HTTP, HTTPS) = rules of talking

API

API = the waiter who brings food from kitchen to your table

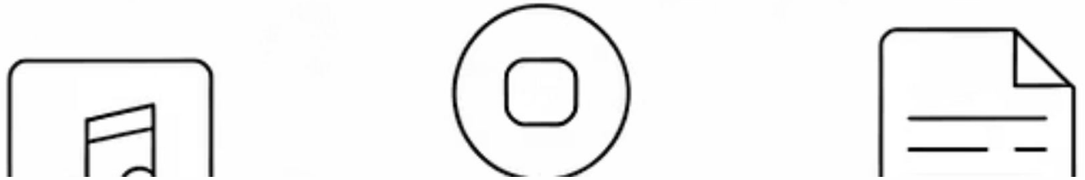
Flutter Relevance: When you call an API in Flutter (http.get), you're literally doing networking.



Real Example – Flutter & Networking

1	2	3
Login Screen	**Server Processing**	**Without Networking**
Flutter sends data to server	Server checks, replies with success/failure	app = offline calculator

> Story: "I once built a weather app in Flutter. It showed only *Dhaka weather*. Why? I hardcoded it. Didn't fetch data from network API. Students laughed at me."



What is an Operating System (OS)?

OS = the manager of your device

Controls hardware, apps, memory

Examples: Android, iOS, Windows

Interactive: Ask: "What's running your Flutter emulator – hardware or OS?"

OS for Flutter Developers

How Flutter Apps Work

- Flutter apps don't run directly on hardware
- They talk with OS → OS controls device

Platform Differences

- iOS has strict rules (App Store)
- Android = more flexible (Google Play)

Flutter Relevance: You must test your Flutter app on both OS because same code may behave differently.

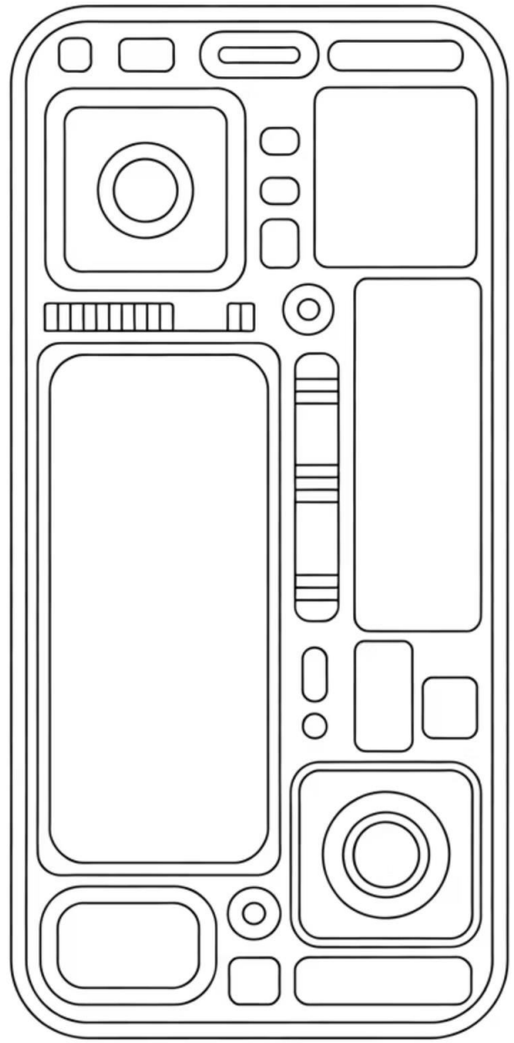
Mobile Device – The Playground for Flutter

Mobile = a small computer in your pocket

Has CPU, RAM, ROM, Battery, Sensors

Works with OS + Network to run apps

Example: Camera app → needs camera hardware + OS permission. Flutter app → same.



How a Mobile Device Works

Input

Touch, mic, sensors

Processing

CPU + GPU

Output

Screen, speaker, vibration

Memory

RAM for running, ROM for storing

Interactive: Ask: *"When you swipe Instagram feed, which part of mobile is working hardest?"* (GPU + Network).

Mobile & Flutter Connection

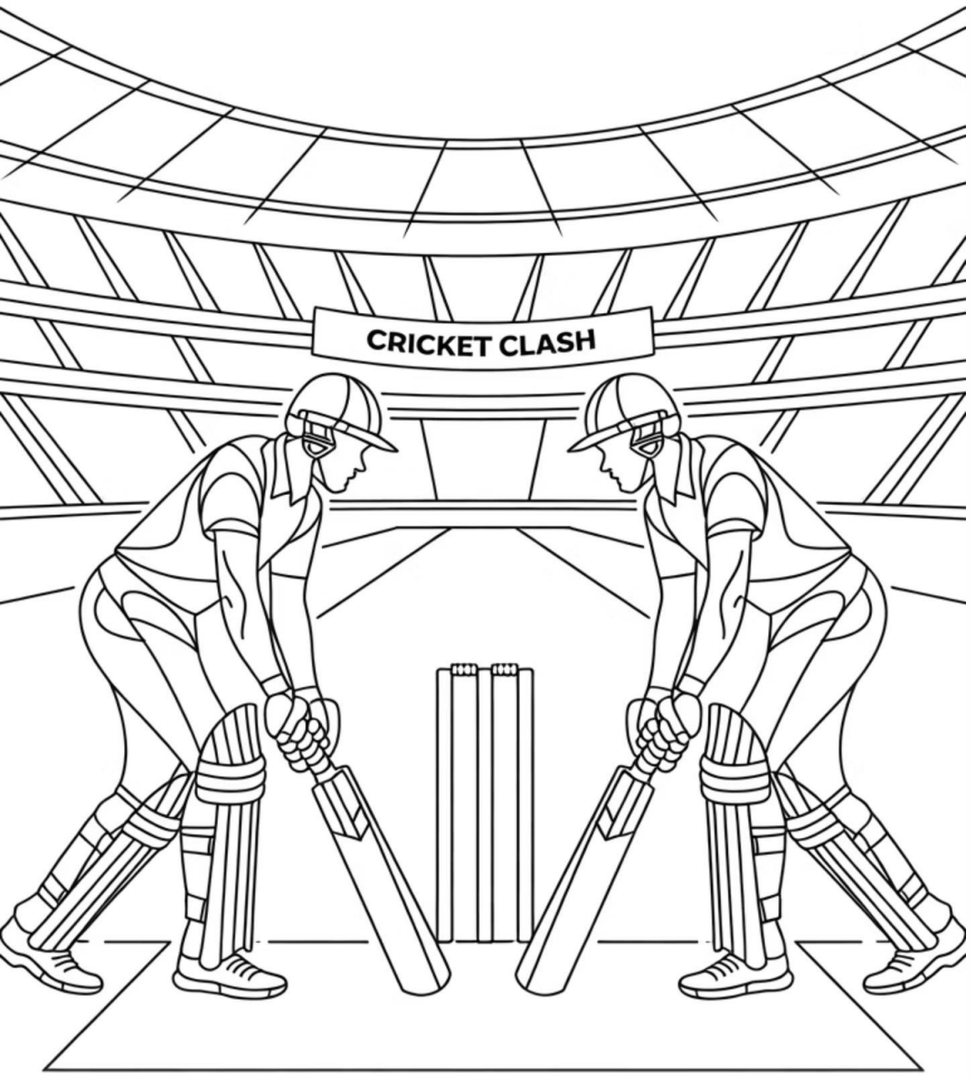
Cross-Platform Power

Flutter is cross-platform: One code → Multiple devices.

But devices have different RAM, OS, battery limits.

That's why developers need to optimize apps.

> Story: "I built an animation-heavy Flutter app. Worked great on my phone. Crashed on my friend's 3GB RAM phone. Lesson: test on low-end mobiles too!"



Android vs iOS War

Think of Android & iOS as two rival cricket teams. Both play the same game → but different styles. As Flutter devs, we must please BOTH audiences.

Story: "My first Flutter app looked perfect on Android… but on iPhone, it looked like an alien invasion. That's when I learned about design languages."

Android OS Basics

- Developed by Google.
- Open source, customizable.
- Runs on thousands of device models.

Flutter Relevance: Your Android Flutter app must run on low-end & high-end phones (Samsung to Symphony).

iOS OS Basics

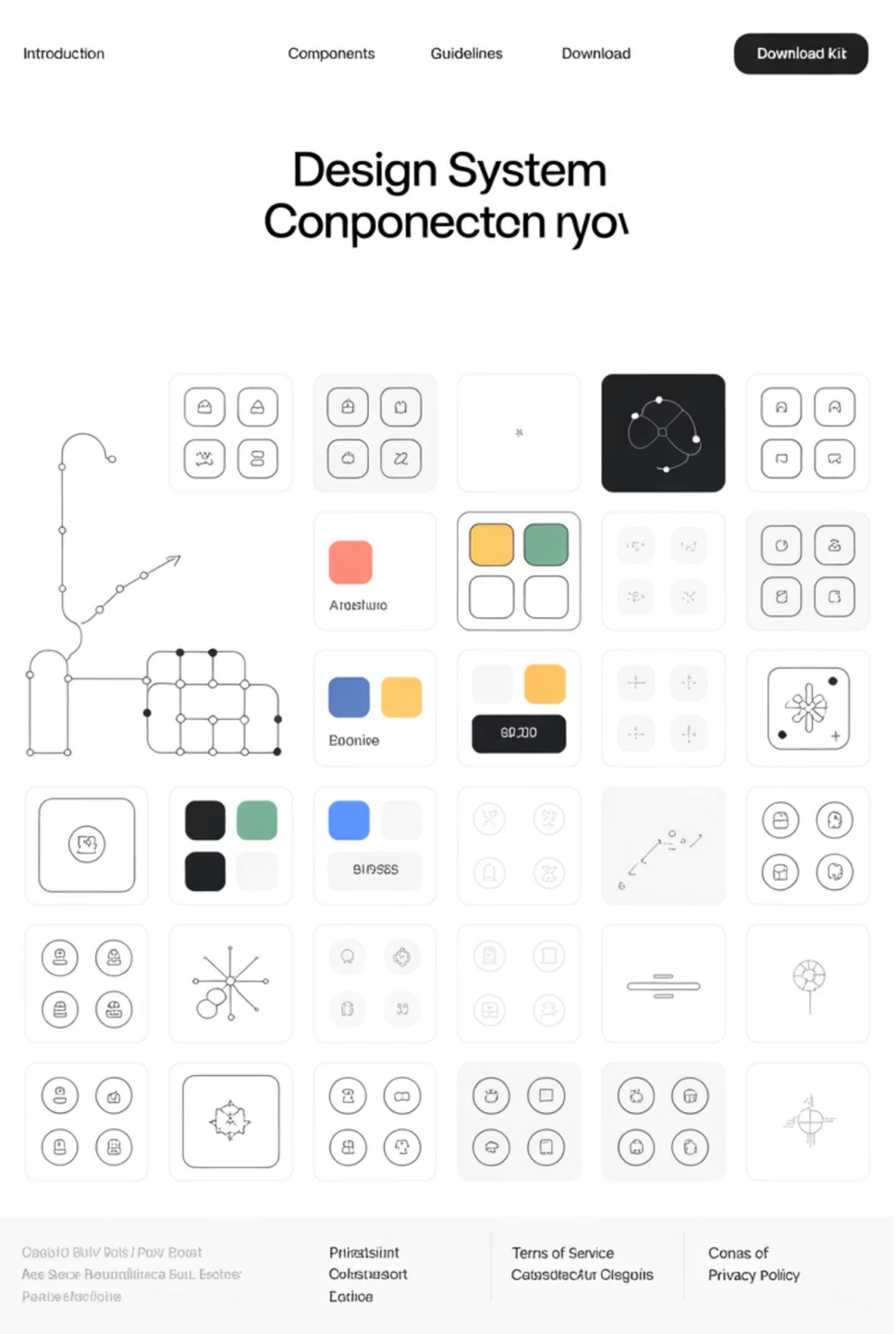
- Developed by Apple.
- Closed, highly controlled.
- Limited devices → iPhones, iPads only.

Flutter Relevance: Your iOS Flutter app must follow Apple's strict design & App Store rules.

Why OS Matters for Flutter Devs

- Flutter writes one codebase.
- But Android & iOS handle UI & performance differently.
- Example: Permissions (camera, storage) → Different flows.

Interactive: Ask: "Who here uses Android? Who uses iPhone? Which feels smoother to you?"



Design Language – What is It?

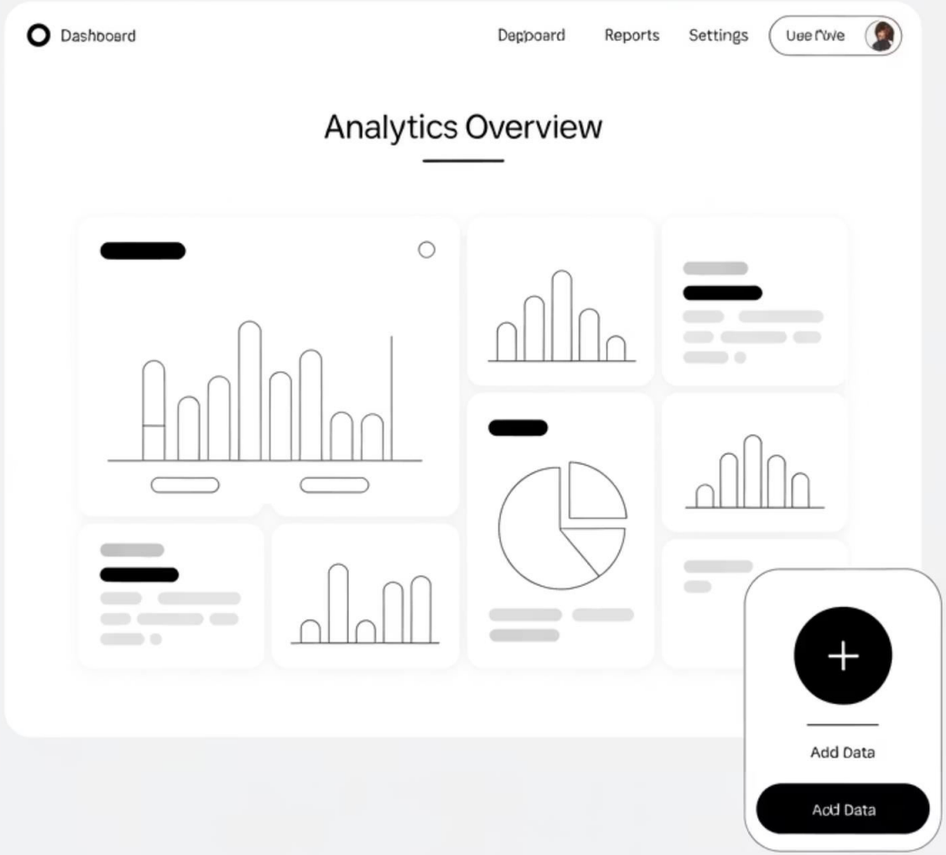
- A set of rules that define how apps "look & feel."
- Colors, buttons, animations, icons, spacing.
- Without design language → apps feel messy & inconsistent.

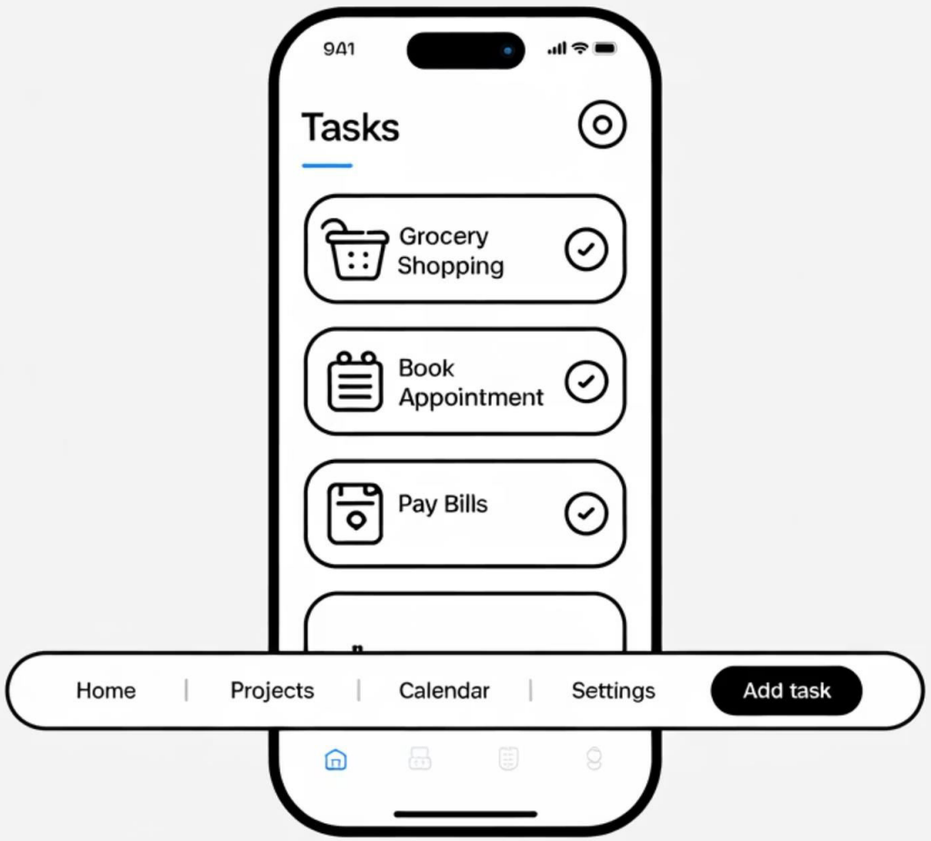
Example: Imagine a Facebook app where buttons look different on Android vs iOS → confusing!

Material Design (Google's Language)

- Born from Android.
- Clean, bold, flat look.
- Floating Action Buttons, Snackbars, Cards.

Flutter Relevance: Most Flutter widgets (Scaffold, AppBar, FAB) are built using Material design.





Cupertino Design (Apple's Language)

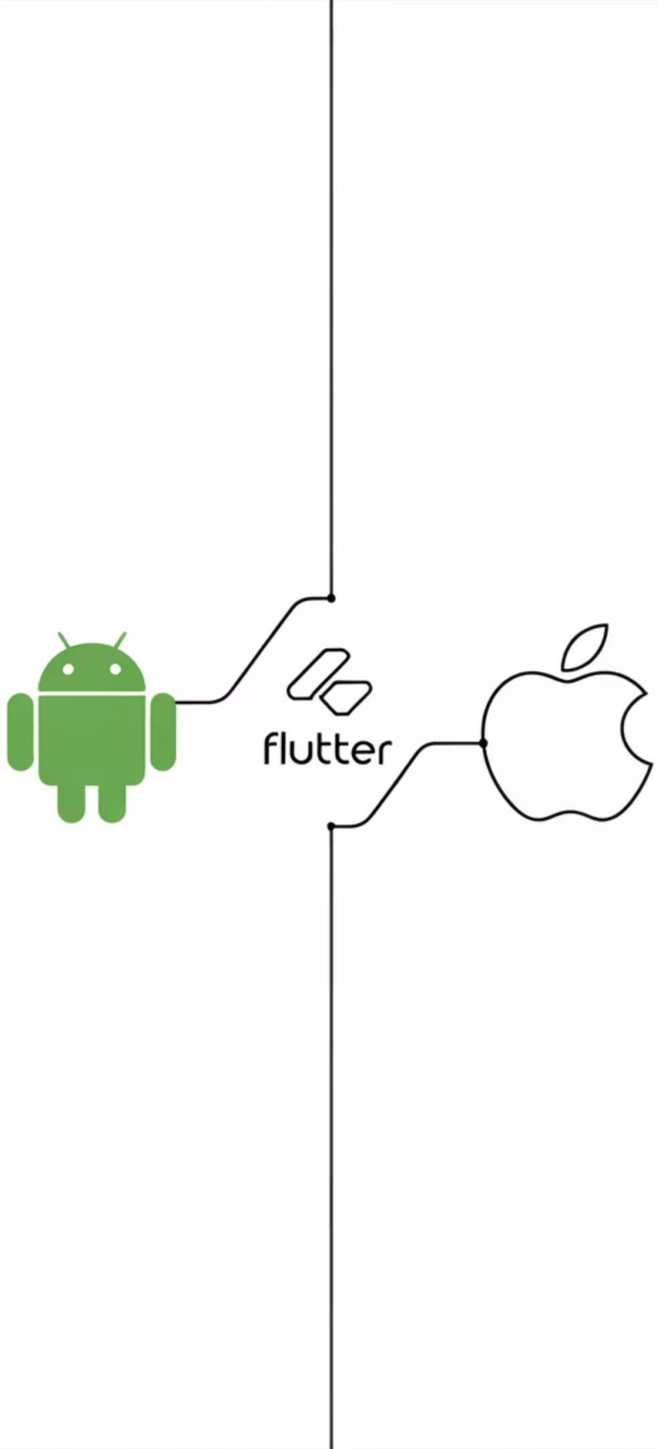
- Born from iOS.
- Smooth, elegant, "glass-like" design.
- Bottom tabs, sliding navigation, switches.

Flutter Relevance: Flutter provides Cupertino widgets (CupertinoButton, CupertinoSwitch, etc.) to make iOS users feel at home.

Material vs Cupertino (Side by Side)

Feature	Material (Android)	Cupertino (iOS)
Buttons	Bold, filled, flat	Rounded, smooth, elegant
Navigation	Drawer, AppBar	Bottom tab bar, gestures
Animation	Fast, snappy	Smooth, elastic

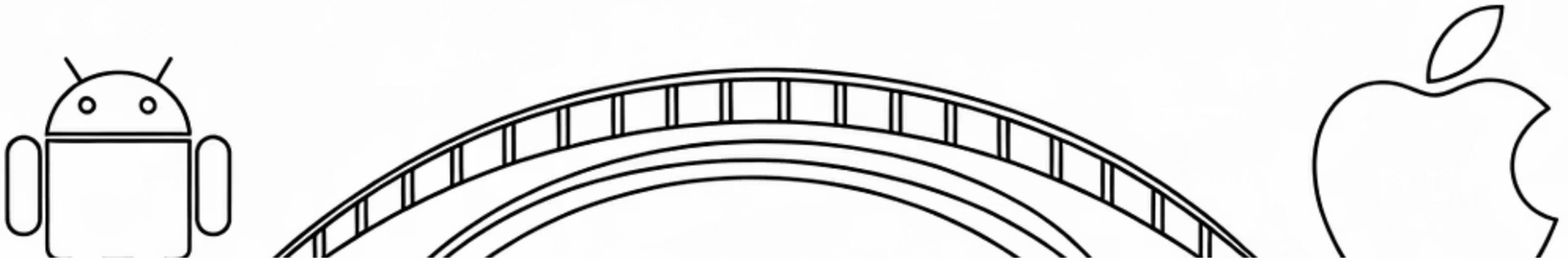
Story: "I once used Material design on an iOS app. My iPhone-using friend said: 'Bro, this feels like a cheap Android copy.' Never again!"



Flutter's Superpower

- With Flutter, you can mix & match.
- MaterialApp → Android look.
- CupertinoApp → iOS look.
- Hybrid? Use both in same app!

Interactive: Ask: "If you build a banking app, would you keep Android bold design or iOS elegant look?"



Wrap-Up & Takeaway

- Android OS → open, customizable, wide audience.
- iOS OS → controlled, premium, strict rules.
- Material Design = Android's style.
- Cupertino = iOS's style.
- As Flutter devs → You're the bridge. One code, two worlds.

Ending Question: *"Next time you design a button in Flutter, will you choose Material or Cupertino?"* 😉

Students Will Clearly See

- Why OS matters (different ecosystems)
- Why design languages matter (user expectations)
- How Flutter handles both with ease

👉 **Students will clearly see:**

- **Why OS matters (different ecosystems).**
- **Why design languages matter (user expectations).**
- **How Flutter handles both with ease.**