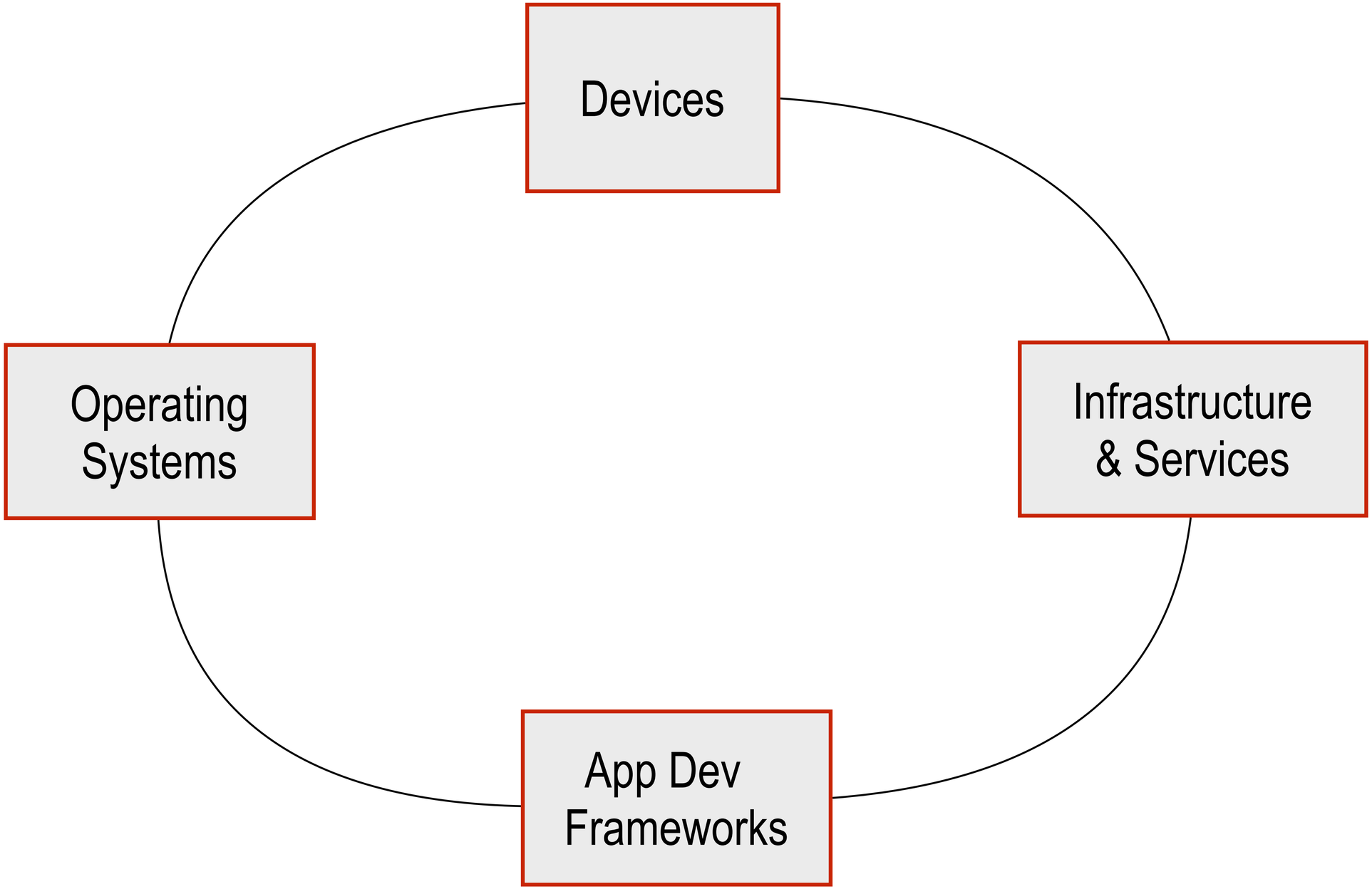
EPFL

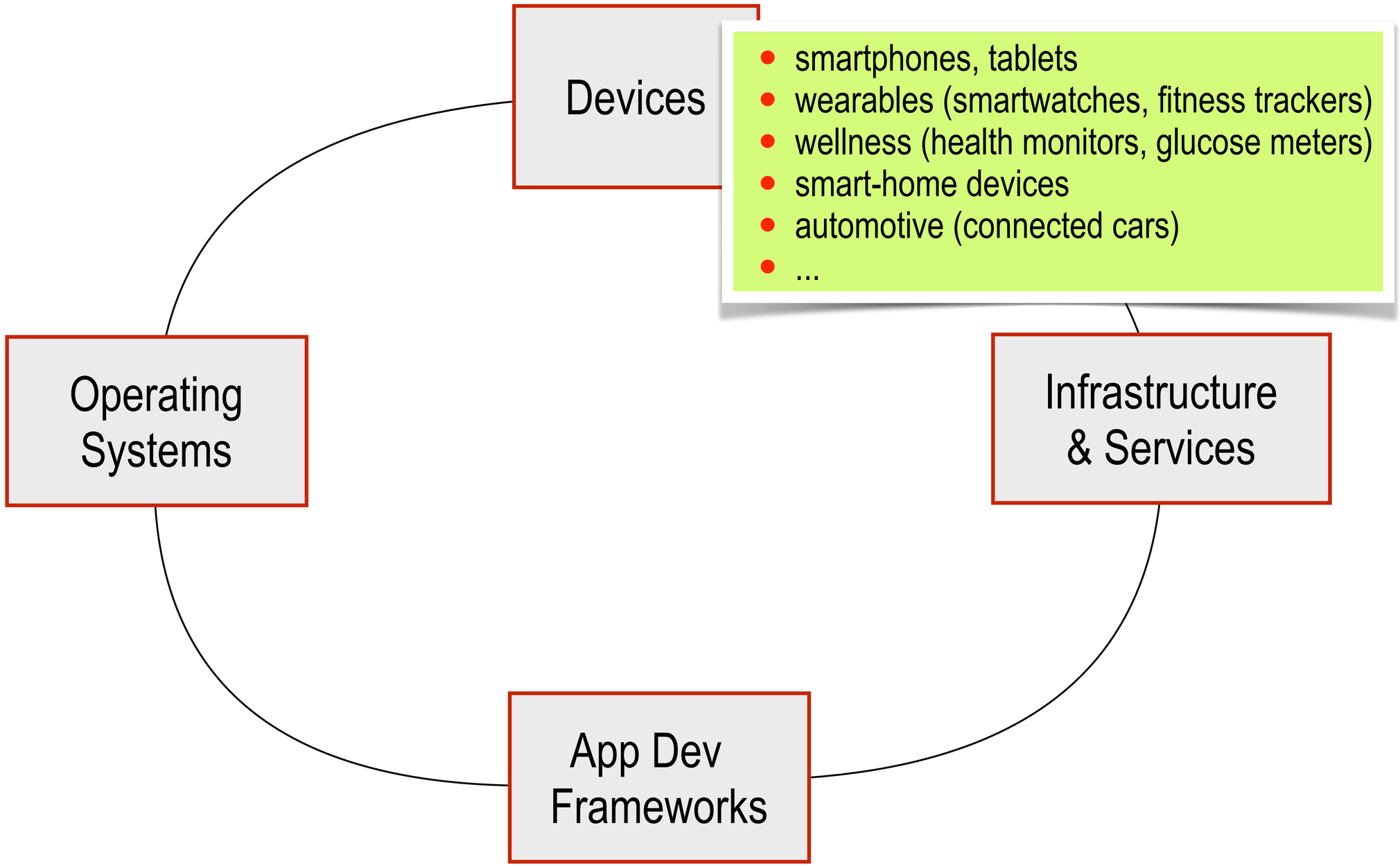
CS-311: The Mobile Platform

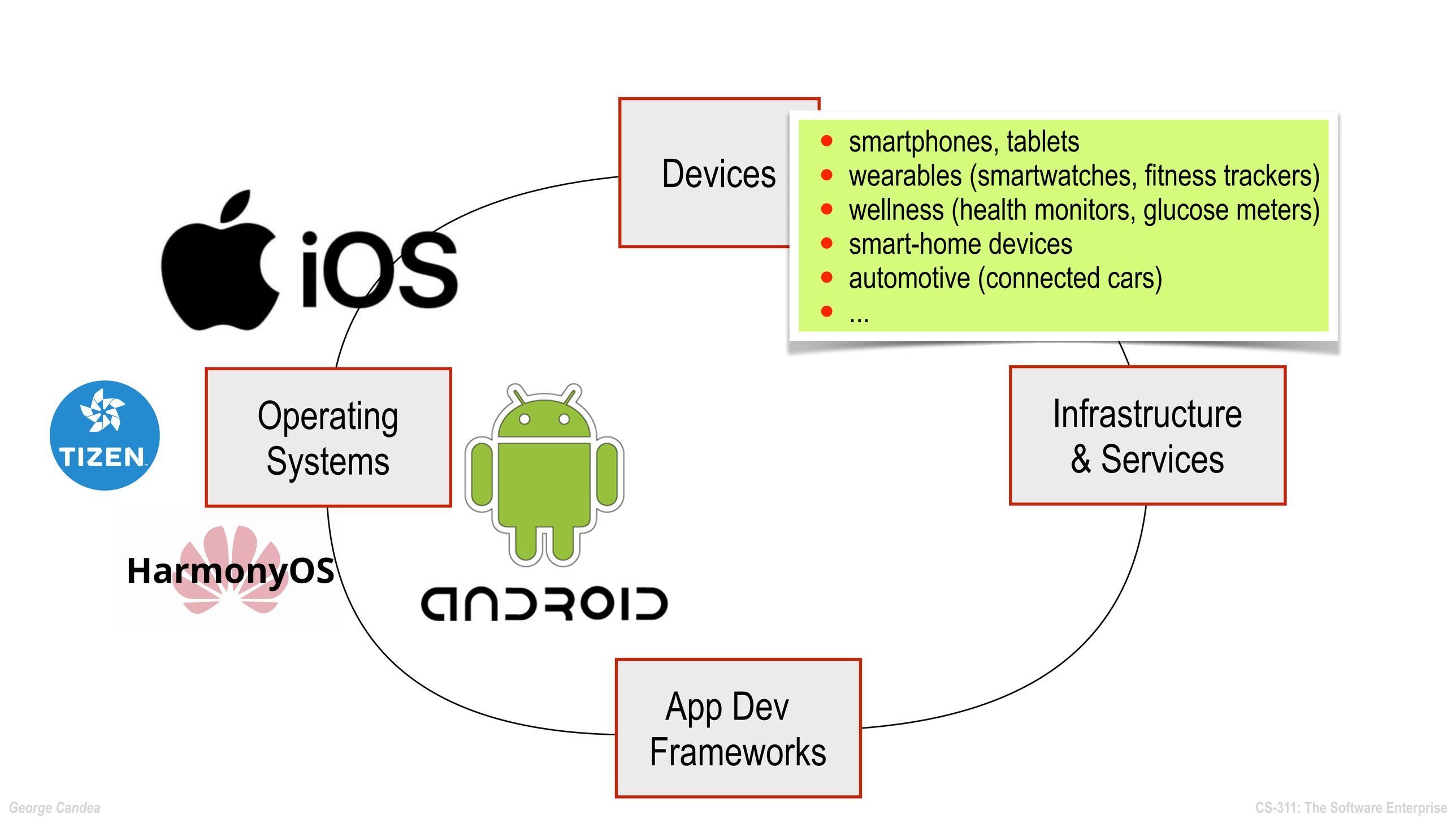
Prof. George Candea
School of Computer & Communication Sciences

Mobile Platform

The ingredients that come together to enable the creation, distribution, and operation of mobile content and services


Devices
Infrastructure & Services
App Dev Frameworks
Operating Systems

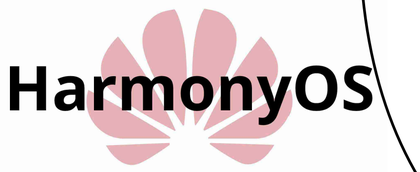

Devices
smartphones, tablets
wearables (smartwatches, fitness trackers)
wellness (health monitors, glucose meters)
smart-home devices
automotive (connected cars)
...
Infrastructure & Services
App Dev Frameworks
Operating Systems


Devices
smartphones, tablets
wearables (smartwatches, fitness trackers)
wellness (health monitors, glucose meters)
smart-home devices
automotive (connected cars)
...
iOS
Operating Systems
TIZEN
HarmonyOS
ANDROID
Infrastructure & Services
App Dev Frameworks

Devices

- smartphones, tablets
- wearables (smartwatches, fitness trackers)
- wellness (health monitors, glucose meters)
- smart-home devices
- automotive (connected cars)
- ...

Operating Systems



Infrastructure & Services

- app distribution (play/app stores)
- push notification
- authentication & identity services
- content delivery networks
- cloud computing platforms
- find-my-device, MDM
- 5G, WiFi
- ...

App Dev Frameworks

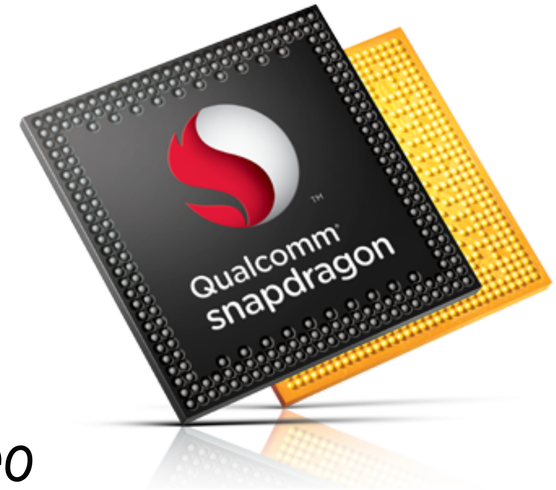
Outline

- Mobile Devices
- Mobile Operating Systems
- Mobile Infrastructure & Services
- Mobile Applications
- Mobile User Experience

Mobile Devices

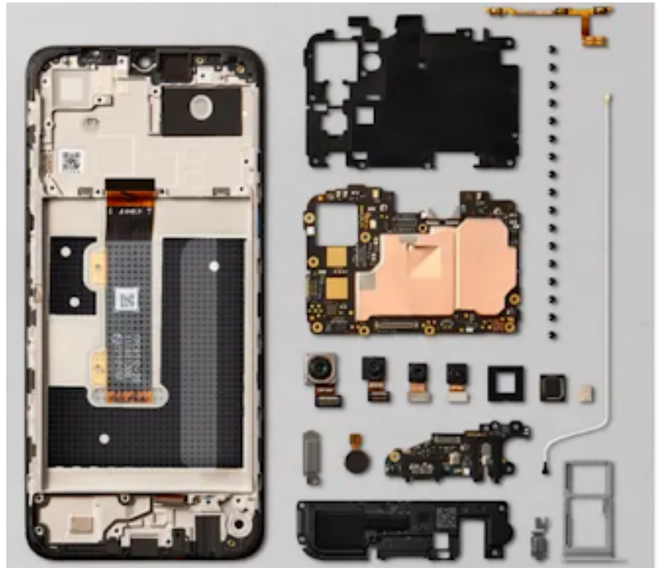
Device Architecture: Core Processing

- CPU (central processing unit)
  - *the "brain" that coordinates all the activities, runs the apps*
- GPU (graphics processing unit)
  - *specialized processor for rendering images, animations, and video*
  - *used for gaming and high-resolution video playback, and smooth visual transitions*
  - *parallel processing capabilities make GPUs well suited for image processing too*
- HSM (hardware security module)
  - *specialized processor dedicated to handling secrets*
  - *OS and apps never gain direct access to the secrets, only via controlled API*



Device Architecture: Data

- Memory (RAM)
  - *more RAM --> more apps can be active simultaneously without slowdown*
  - *mobile app lifecycle and memory mgmt is different from desktop apps*
  - *high-end phones can have as much as 18GB*
- Storage
  - *Internal storage holds the OS, installed apps, and user data*
  - *NAND flash with UFS*
  - *high-end phones can have as much as 1TB*



https://www.ghacks.net/wp-content/uploads/2023/03/MWC-2023-Nokia-unveils-an-affordable-new-Android-smartphone-2.png

Device Architecture: Battery & Sensors

- Battery
  - *Capacity: 1'000 mAh to 10'000 mAh*
  - *energy consumed depends on usage, device's power efficiency, screen, processors*
  - *fast charging and wireless charging add convenience*
- Sensors
  - *Accelerometers and gyroscopes detect device orientation and motion*
  - *Proximity sensors detect when the phone is close to the ear*
  - *Ambient light sensors adjust screen brightness*
  - *Fingerprint readers, facial recognition, and other biometrics enhance security*

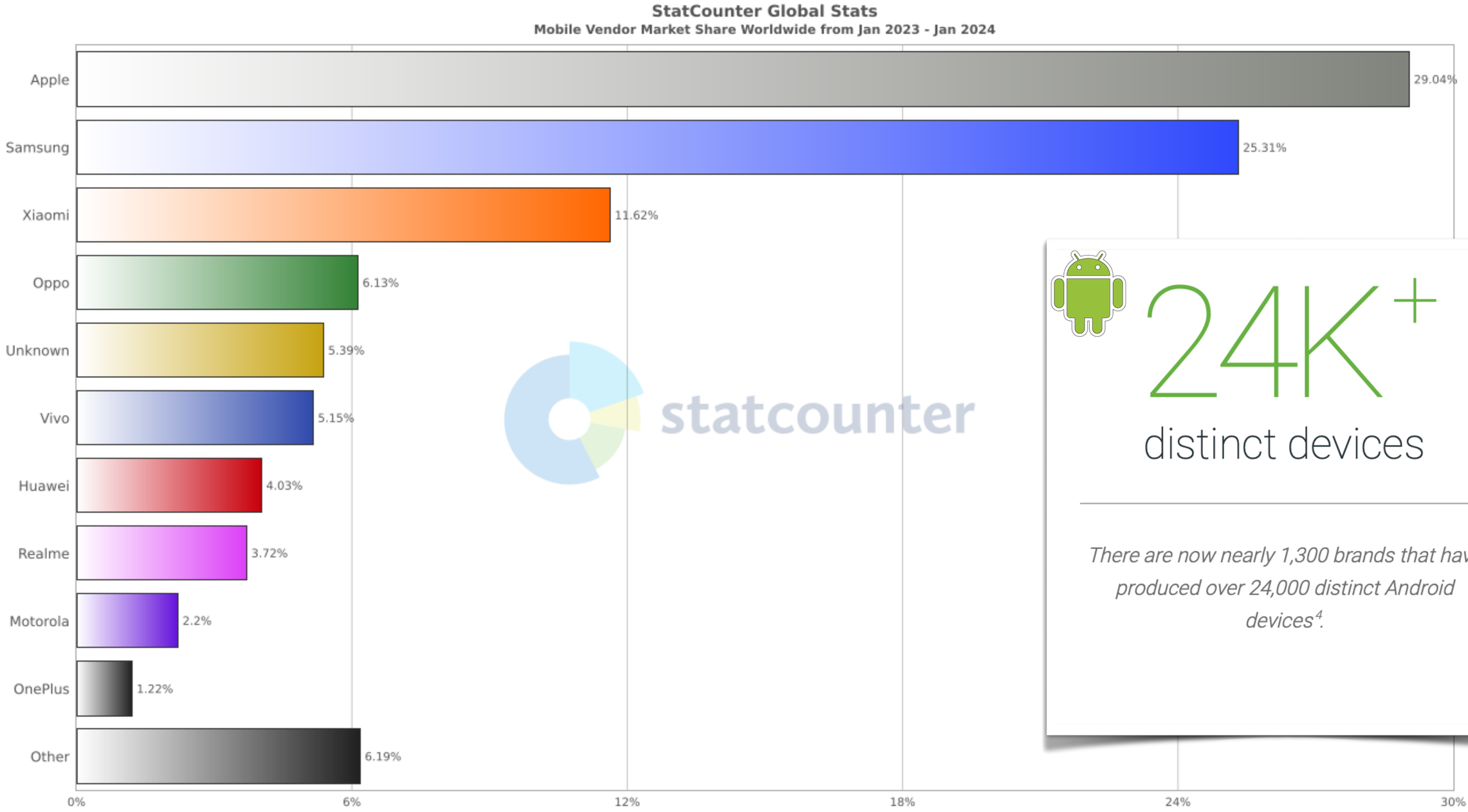
https://www.theverge.com/2021/1/11/22223494/qualcomm-ultrasonic-fingerprint-reader-samsung-galaxy-s21-announcement

https://mybroadband.co.za/news/smartphones/500497-smartphones-with-removable-batteries-could-make-a-comeback.html

Device Architecture: Interaction with the Outside

- Display
  - *Display sizes: 4" to 6.9" for phones, bigger for tablets*
  - *Pixel density: 180 to 640 PPI for currently selling phones*
  - *Resolution: 480x854 to 1644x3840 (4K)*
- Cameras
  - *Rear cameras: wide-angle, telephoto, ultra-wide, and macro lenses*
  - *Front-facing cameras for selfies and video calls*
  - *Megapixels: 2MP to 200MP*
- Connectivity
  - *GPS, Cellular (4G LTE, 5G), WiFi, Bluetooth, NFC*

https://static1.srcdn.com/wordpress/wp-content/uploads/2023/01/samsung-galaxy-s-22-ultra-camera.jpg?q=50&fit=crop&w=825&dpr=1.5



24K+

distinct devices

There are now nearly 1,300 brands that have produced over 24,000 distinct Android devices[4].

Outline

- Mobile Devices
- Mobile Operating Systems
- Mobile Infrastructure & Services
- Mobile Applications
- Mobile User Experience

Mobile Operating Systems

Applications

- non-Mobile
  - *User explicitly starts an app*
  - *User explicitly switches between apps*
  - *User closes app*
  - *OS takes a passive role*
  - *Preemptive schedulers*
- Mobile
  - *Interacting with app through multiple points*
  - *Diff. between running vs. not is fluid*
  - *Apps can be killed/restarted anytime*
  - *Notifications*
  - *Cooperative execution model*

Security

- non-Mobile
  - *Multi-user (think parents' computer)*
  - *Mutually distrustful users*
  - *Files with permissions*
  - *Applications run with user's privileges*
  - *OS protects apps from each other*
    - but not I/O from apps
  - *Security is an afterthought*
    - anti-virus, sandboxes, ...

- Mobile
  - *Single-user (who might be "stupid")*
  - *Mutually distrustful apps*
  - *OS protects user's data*
  - *Tightly controlled execution env*
  - *You don't get "root" or "Administrator"*
  - *HW isolation (e.g., FaceID)*
  - *App asks for permissions at install time*
    - *access contacts, camera, microphone, location information, SMS, WiFi, user accounts, body sensors, etc.*

Operating Environment

- Energy and power management
- Mobile OS specialized for the device
- Less backward compatibility on mobiles
- Less storage capacity on mobiles
- Desktop used to tethered network, mobile is intermittent
- Input via keyboard/mouse vs. voice/gestures
- Output limited on mobile

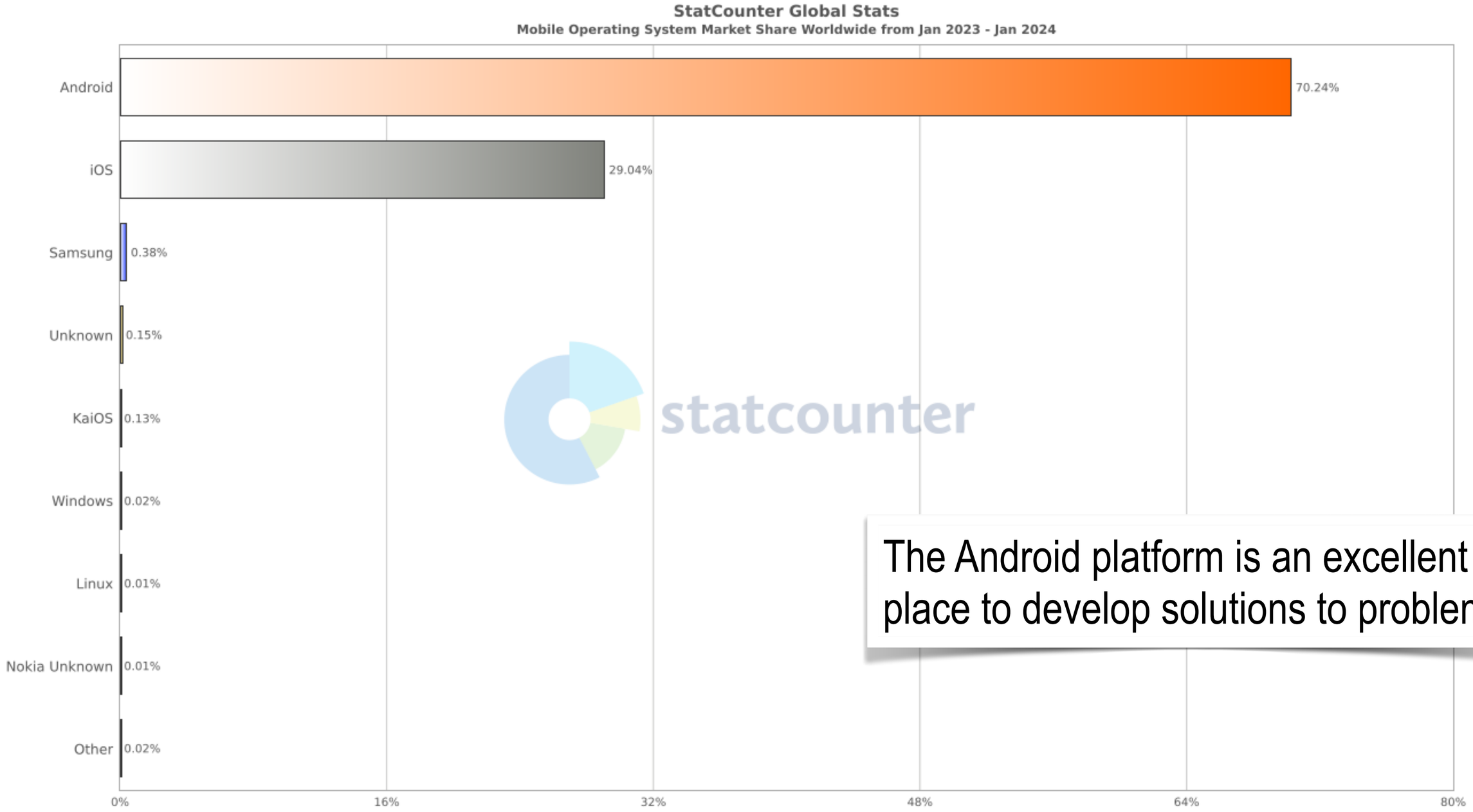
Android vs. iOS

- open-source, highly customizable
- tens of thousands of devices
- variety of UI skins
  - *Samsung One UI, Xiaomi MIUI, ...*
- ("everyone", i.e., all possible users)
- based on Linux
- hybrid runtime: JVM and ART

- closed, controlled ecosystem
- only iPhones and iPads (and a few others)
- uniform and consistent user experience
- (high-revenue users)
- based on XNU
- fully native execution

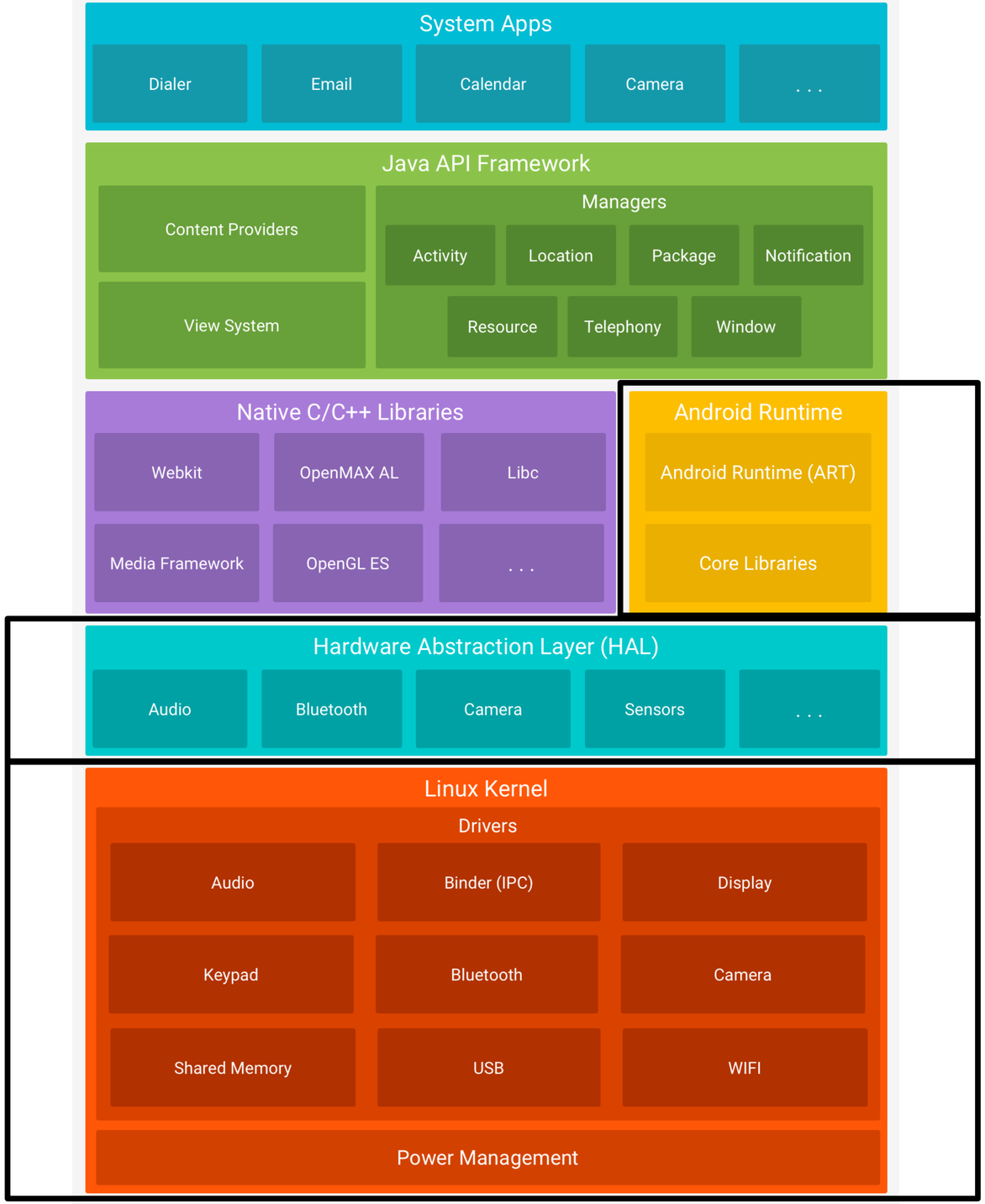
Mobile Operating Systems

Example: Android



The Android platform is an excellent place to develop solutions to problems

Android Stack



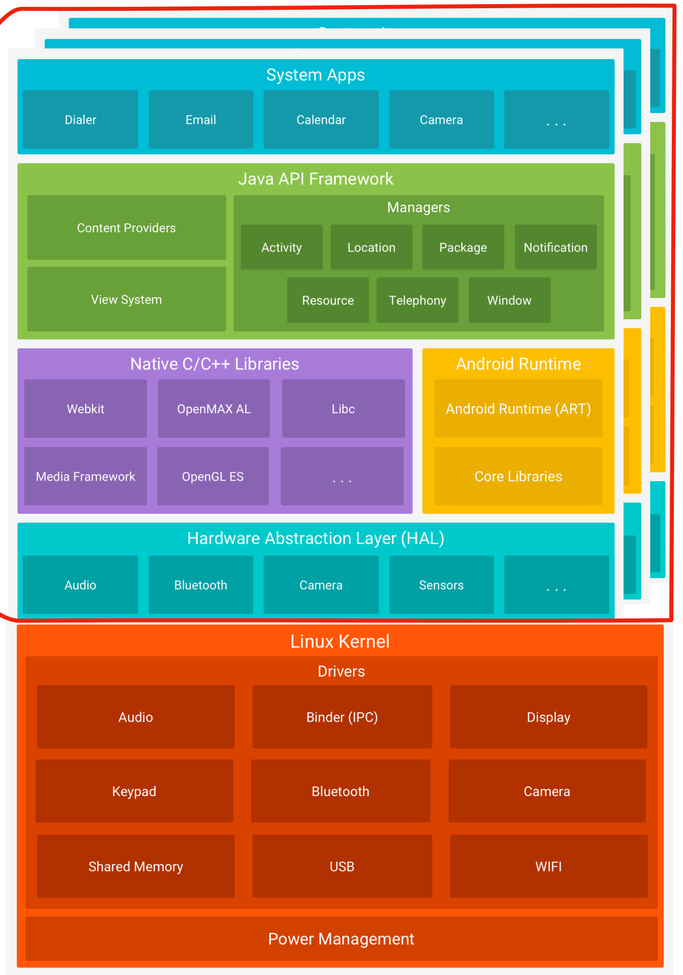
Env. for executing app bytecode

Interface to HW capabilities

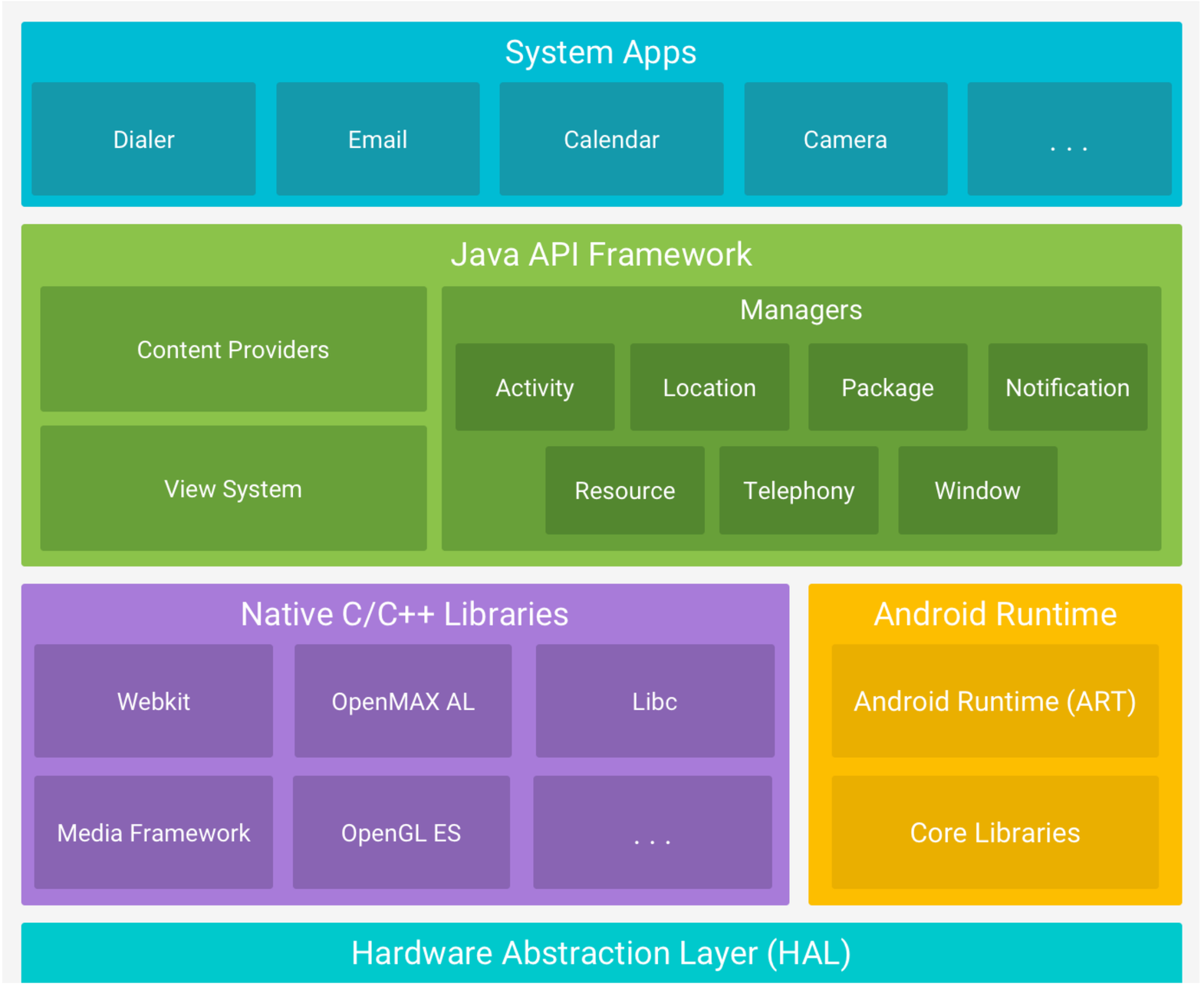
Inner core of the operating system

Apps executes in a per-app security sandbox

- Apps written in Kotlin (or Java, C++)
- Deployment: on device (APK) or in app store (AAB)
- App executes in a per-app security sandbox
  - *1 Linux user per app, 1 Linux process per app*
    => replication of top of Android stack
  - *Follows principle of least privilege*
- Sharing is possible through explicit mechanisms
  - *Apps from same developer can directly share files*
  - *With the user's permission, apps can access device data*
- Priority-based scheduling + process containers (cgroups)



Top of Android Stack



Persistent State in Android

- Shared Preferences
  - *key-value pairs of primitive types (e.g., <"theme", 4>, <"sound_enabled", true>)*
  - *for small amount of key-value data*
- Internal storage
  - *files private to your app (e.g., images and text documents)*
  - *for moderate amounts of data*
- External storage
  - *files on shared storage space (e.g., SD card, cloud)*
  - *can be accessed by the user and potentially other apps, if you grant permissions*
  - *suitable for large files meant to be shared*

Structured Persistent State

- SQLite Databases
  - *structured data(e.g., user profiles, shopping cart contents, chat history)*
  - *best for substantial amounts of structured data*
  - `android.database.sqlite` *package*
- Room Persistence
  - *abstraction layer over SQLite*
  - *Entities, Data Access Objects, Databases*
  - *smooth integration within the language*

```
@Entity
data class User(
    @PrimaryKey val userId: Int,
    val name: String
)

@Dao
interface UserDao {
    @Insert
    fun insert(user: User)

    @Query("SELECT * FROM User WHERE userId = :id")
    fun getUserById(id: Int): User?
}

@Database(entities = [User::class], version = 1)
abstract class AppDatabase : RoomDatabase() {
    abstract fun userDao(): UserDao
}
```

Outline

- Mobile Devices
- Mobile Operating Systems
- Mobile Infrastructure & Services
- Mobile Applications
- Mobile User Experience

Mobile Infrastructure & Services

Split-App Model

- Client-side (frontend)
  - *runs on the user's device, provides the UI and UX*
- Server-side (backend)
  - *runs on a server or in the cloud*
- Client interacts with cloud services
  - *typically REST APIs or GraphQL*
  - *send HTTP request*
  - *get back JSON or XML*

https://hacker-news.firebaseio.com/v0/newstories.json

https://hacker-news.firebaseio.com/v0/item/39456069.json


```
[39456086,39456083,39456079,39456069,39456026,
39456015,39455995,39455986,39455982,39455981,3
9455961,39455952,39455939,39455928,39455918,39
455915,39455911,39455903,39455902,39455891,394
55874,39455873,39455839,39455827,39455825,3945
5821,39455810,39455783,39455770,39455750,39455
746,39455745,39455724,39455683,39455682,394556
76,39455633,39455632,39455617,39455607,3945559
3,39455576,39455572,39455553,39455546,39455536
,39455535,39455533,39455522,39455503,39455473,
39455421,39455336,39455319,39455305,39455301,3
9455294,39455275,39455271,39455267,39455265,39
455257,39455230,39455220,39455187,39455185,394
55184,39455177,39455173,39455170,39455167,3945
5146,39455131,39455122,39455108,39455103,39455
102,39455080,39455075,39455057,39455044,394550
33,39455029,39455027,39455022,39454997,3945499
3,39454961,39454959]
```



```
[{
"by":"tosh",
"descendants":0,
"id":39456069,
"score":2,
"time":1708533765,
"title":"Imperfect Compression",
"type":"story",
"url":"https://mathspp.com/blog/problems/imperfect-
compression"
}]
```

Backend/Network Services

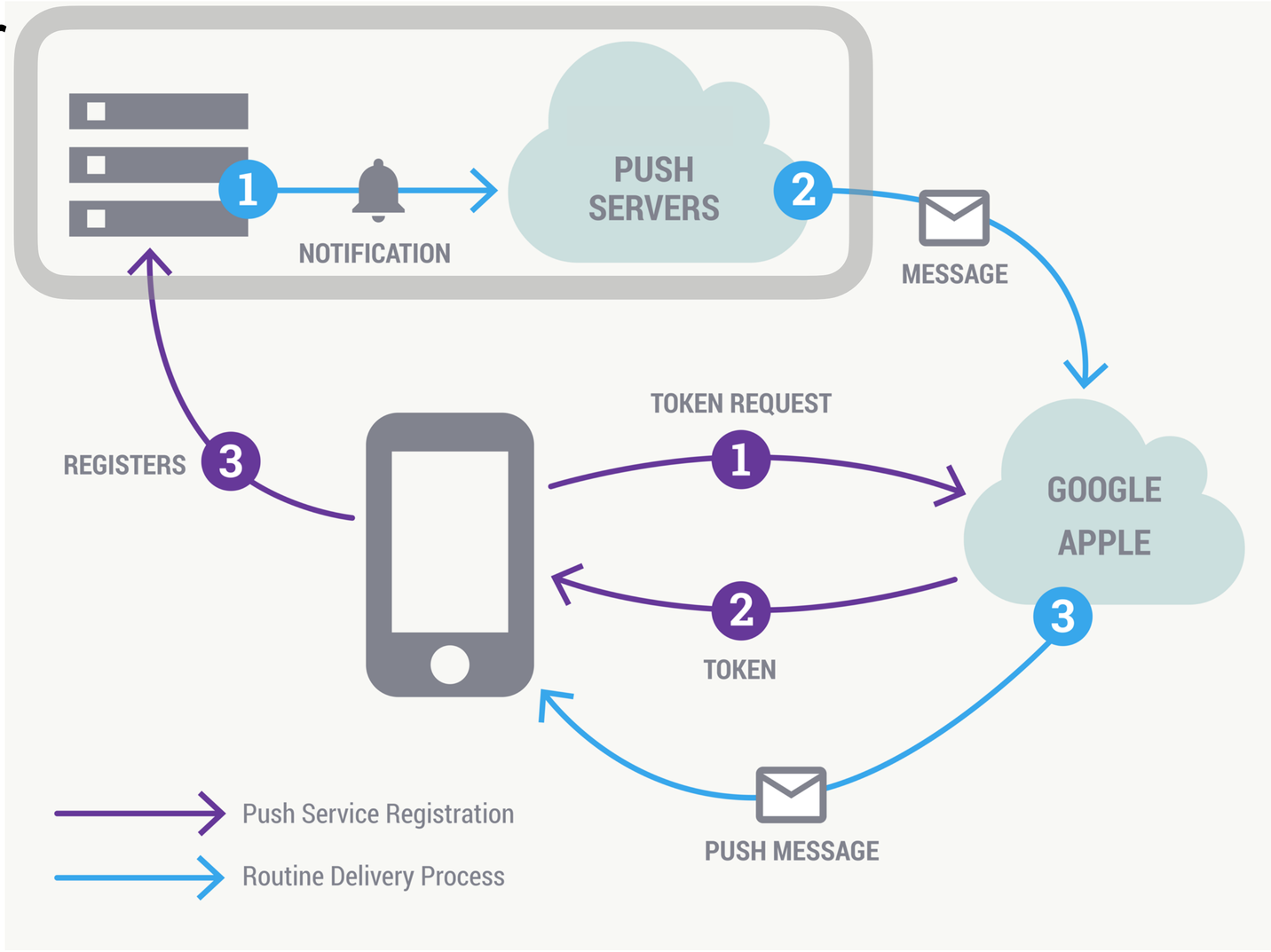
- Authentication and authorization
  - *user sign-up, sign-in, and sign-out processes; access control and permissions*
- Data storage and mgmt
  - *user data and app content, typically with a CRUD interface*
- Real-time data sync
  - *real-time updates of data across user devices*
- Payment processing
  - *in-app purchases, subscriptions, payment verification and receipts*

Backend/Network Services

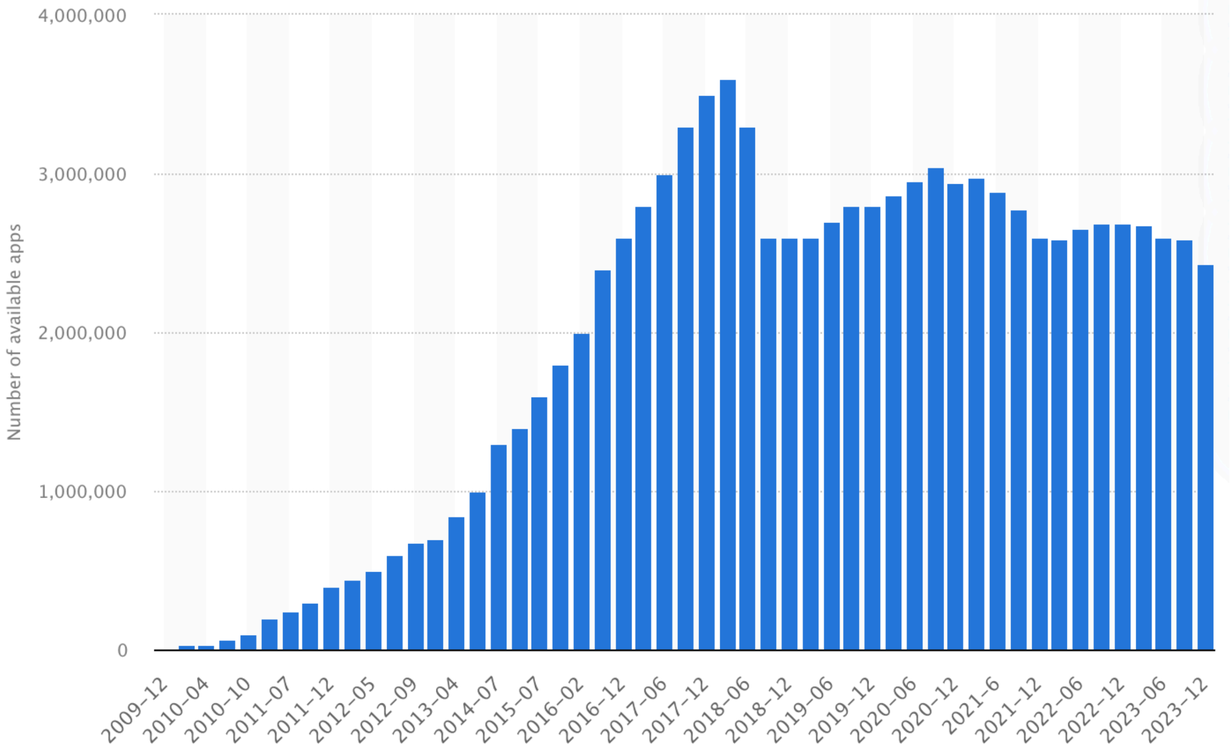
- Content Distribution Networks
- Analytics and reporting
- Email and SMS
- Backup and recovery
- Geo-location
- Integration services

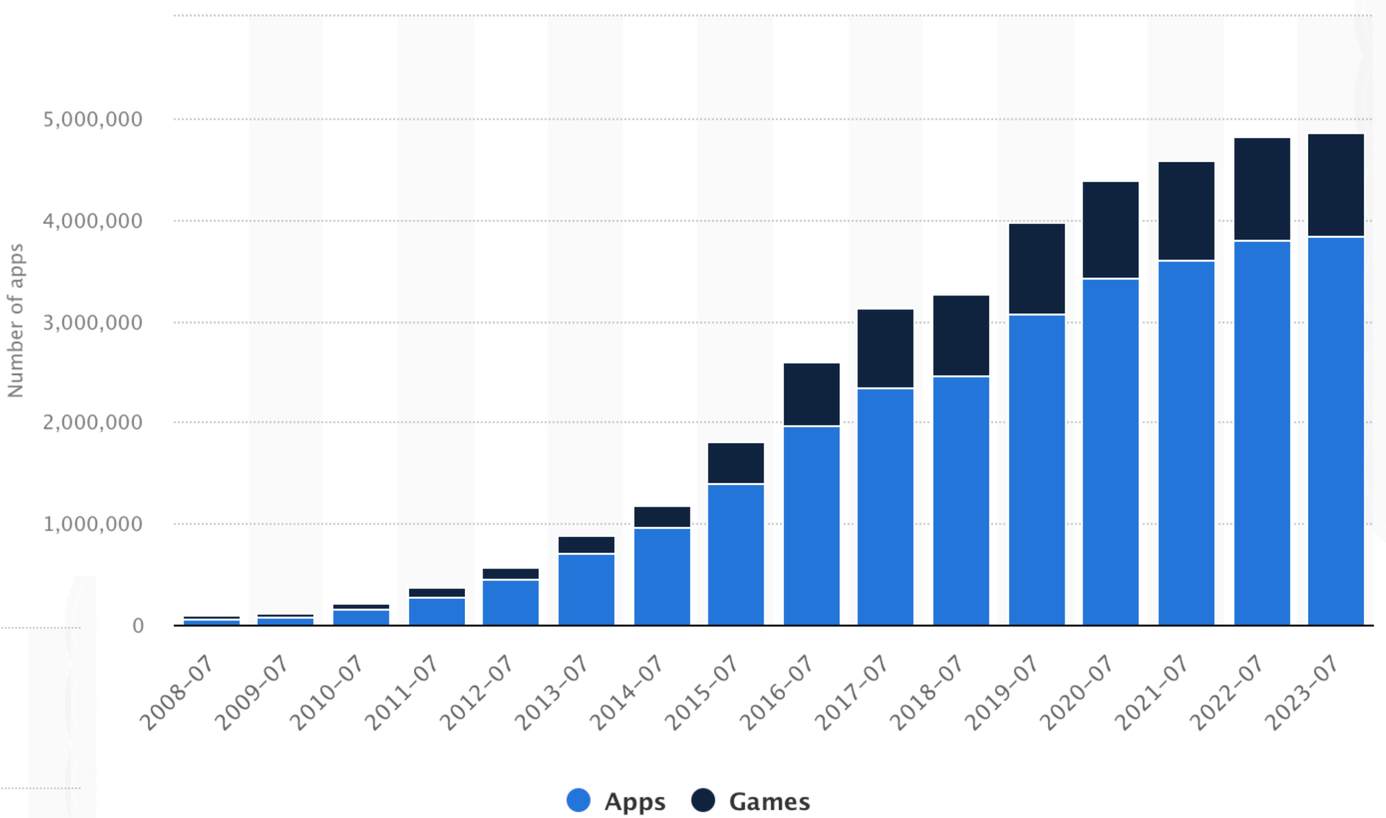
Push Notifications

- Messages sent to the user by the backend of the app
- Each OS has its own push notification service



https://knowledge.opsview.com/docs/getting-started-with-push





Charts courtesy of Statista.com

App Vetting

- Focus: Security, Privacy, UX
- Stringent guidelines on functionality, design, safety, monetization, legal
- Automated initial screening
- In-depth scrutiny by humans
- Common reasons to reject: bugs, crashes, inappropriate content, guideline violations, deception

- Focus: Openness, Flexibility, Scale
- Developer guidelines give more flexibility
- Automated screening
- Monitor apps post-release and rely on user reports to catch bad apps
- Applies more stringent reviews in sensitive categories (financial apps, kids apps, etc.)

Outline

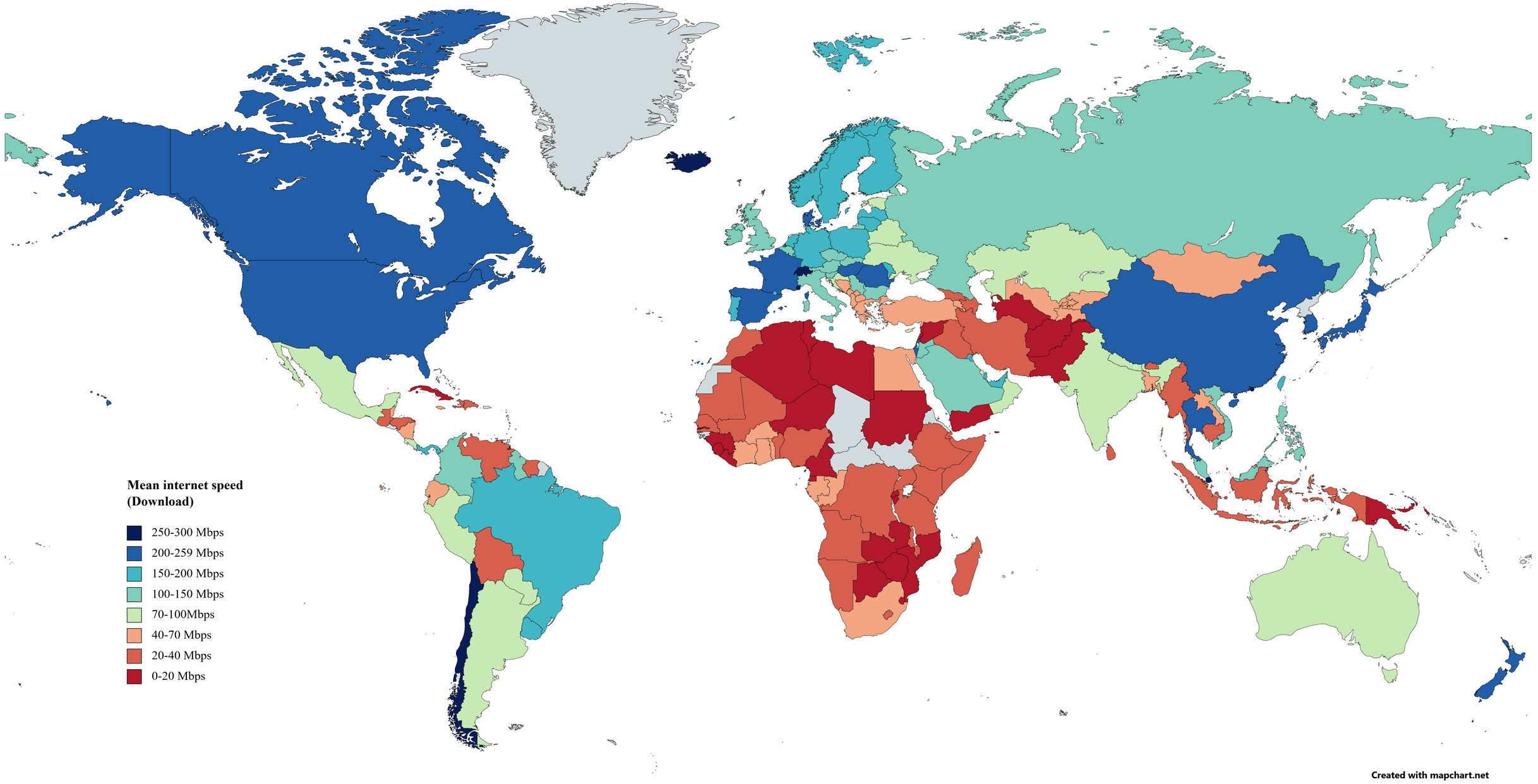
- Mobile Devices
- Mobile Operating Systems
- Mobile Infrastructure & Services
- Mobile Applications
- Mobile User Experience

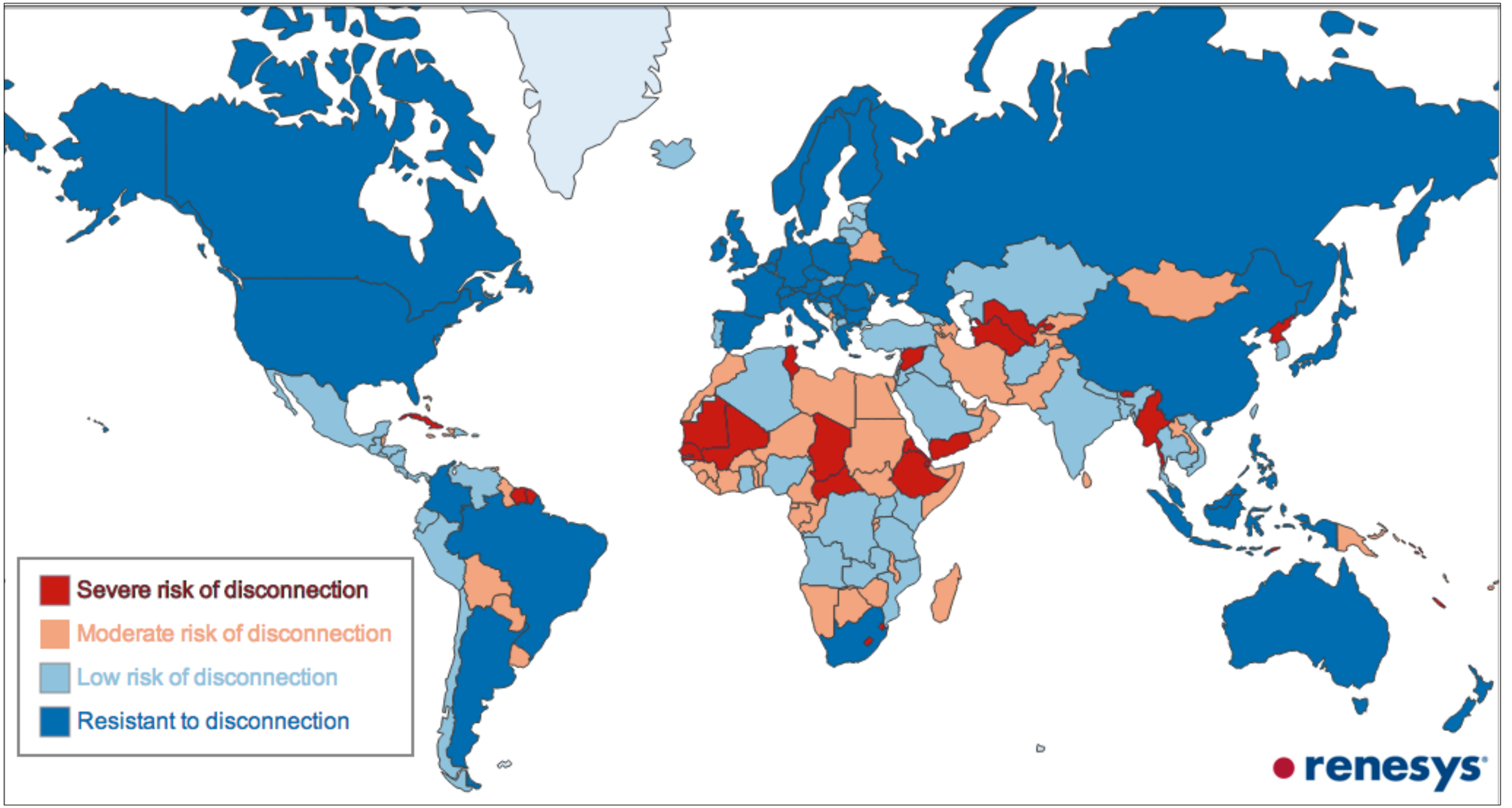
https://www.eatthis.com/wp-content/uploads/sites/4/2022/05/coffee.jpg

Mobile Applications

Offline-First Design

- Network access can be unreliable / intermittent (or missing completely)
  - *app should function effectively under these conditions*
- E.g., PolyFood on train going through tunnel
  - *cache the user's last viewed menu => user can browse menus even when offline*
  - *can order from the menu even when offline — order stored in a local database*
  - *automatically synchronize with the PolyFood server once connectivity returns*




Severe risk of disconnection
Moderate risk of disconnection
Low risk of disconnection
Resistant to disconnection
renesys

Battery Usage

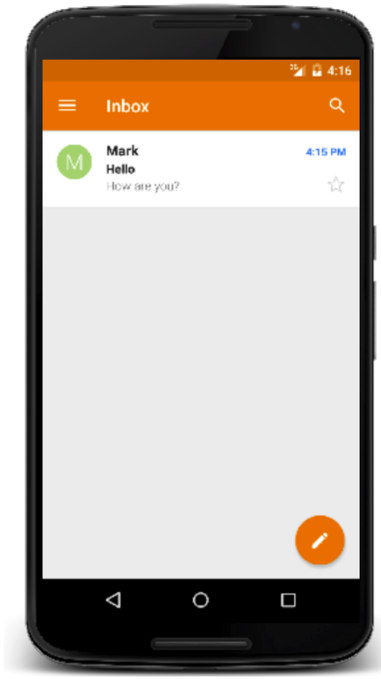
- Need to worry about how much energy your app uses
- Defer non-urgent background tasks
- Android WorkManager
  - *specify constraints that need to be met for the task to run (e.g., network connectivity type, charging status, battery not low, storage not low, device idle)*
  - *tasks will only run then (even if the app exits or the device restarts)*
- E.g., PolyFood
  - *syncing user reviews or menu updates after the tunnel*
  - *high-frequency location updated vs. low-frequency updates*

Mobile Applications

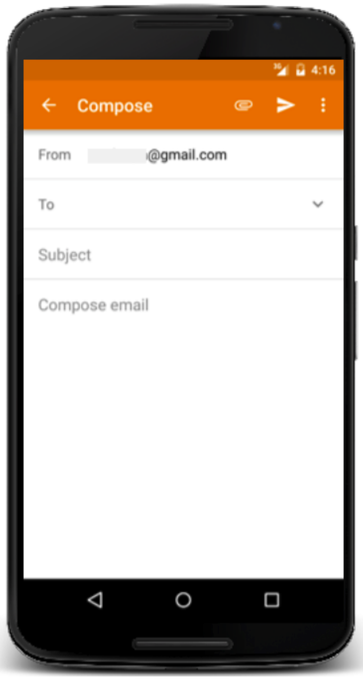
Example: Android Apps

Android App Components

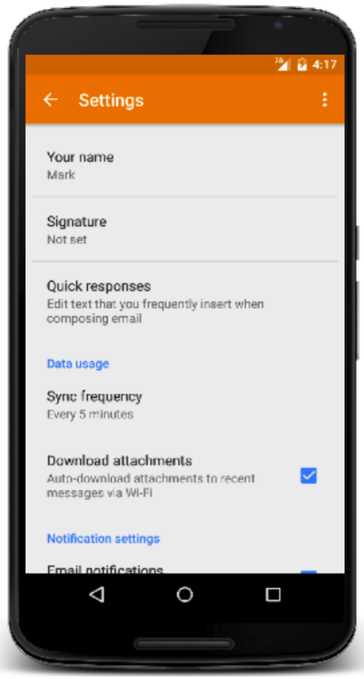
- Activities
  - *Entry point for interacting with the user (1 screen + UI)*
  - *One app can start an activity in another app (e.g., camera can email photo)*



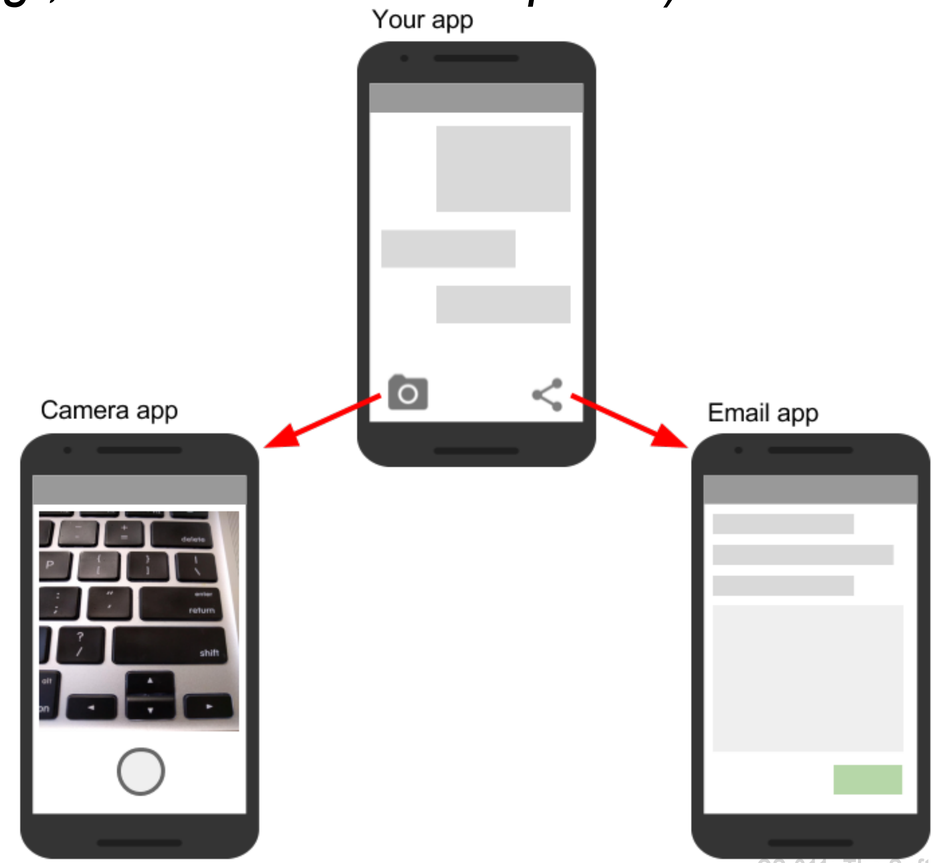
Messages Activity



Compose Activity

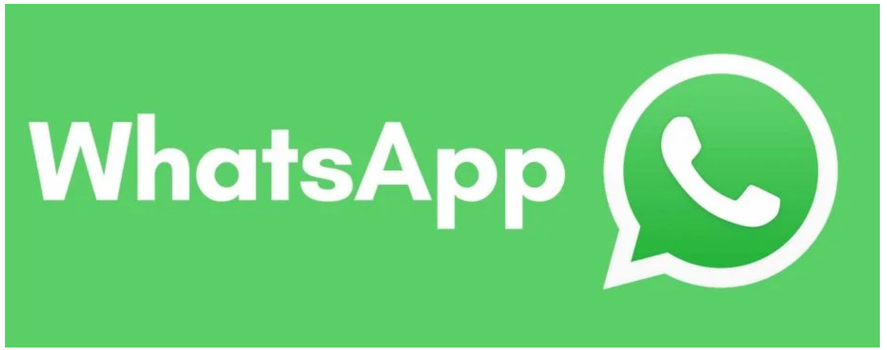


Settings Activity



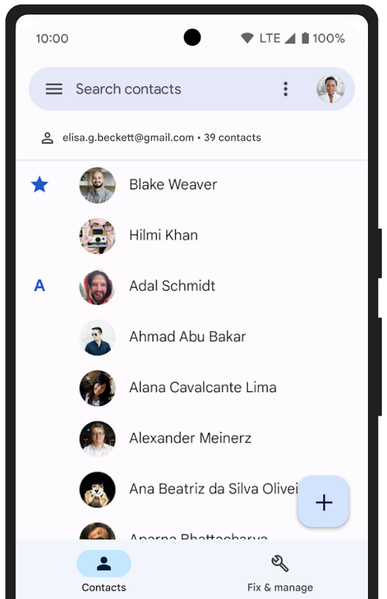
Android App Components

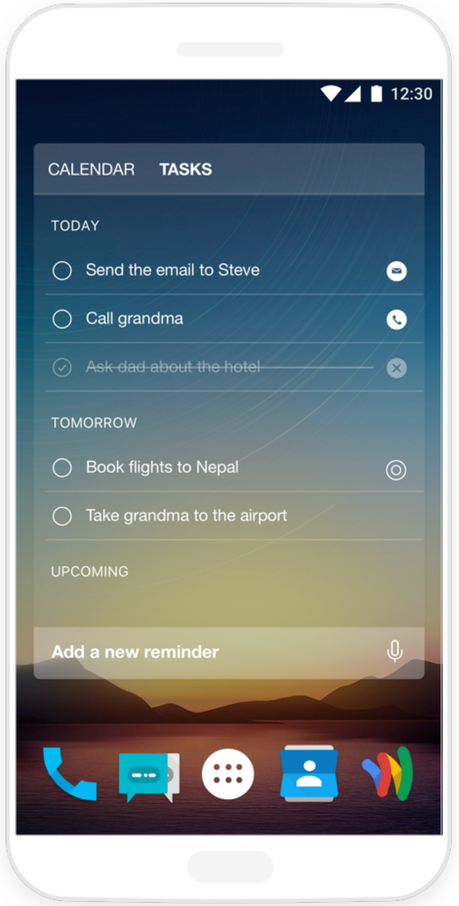
- Activities
  - *Entry point for interacting with the user (1 screen + UI)*
  - *One app can start an activity in another app (e.g., camera can email photo)*
- Services
  - *Run in background without UI*



Android App Components

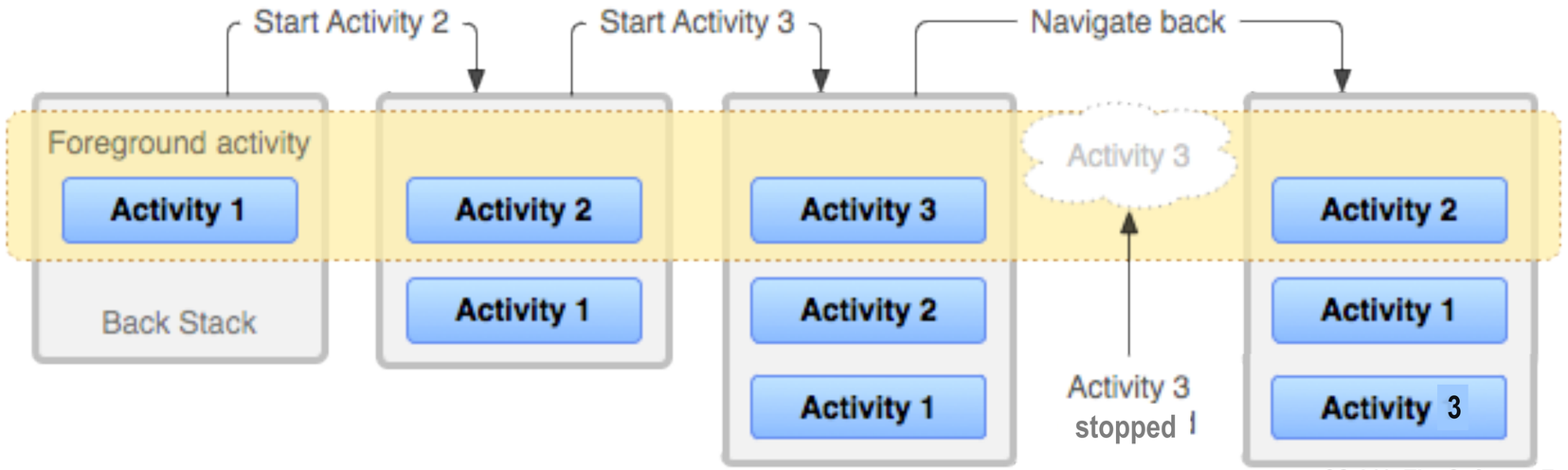
- Broadcast receivers
  - *Receive "out of band" event broadcasts*
  - *App doesn't need to be running to receive broadcast*
- Content providers
  - *Manage data that is of interest to multiple apps*





Activities

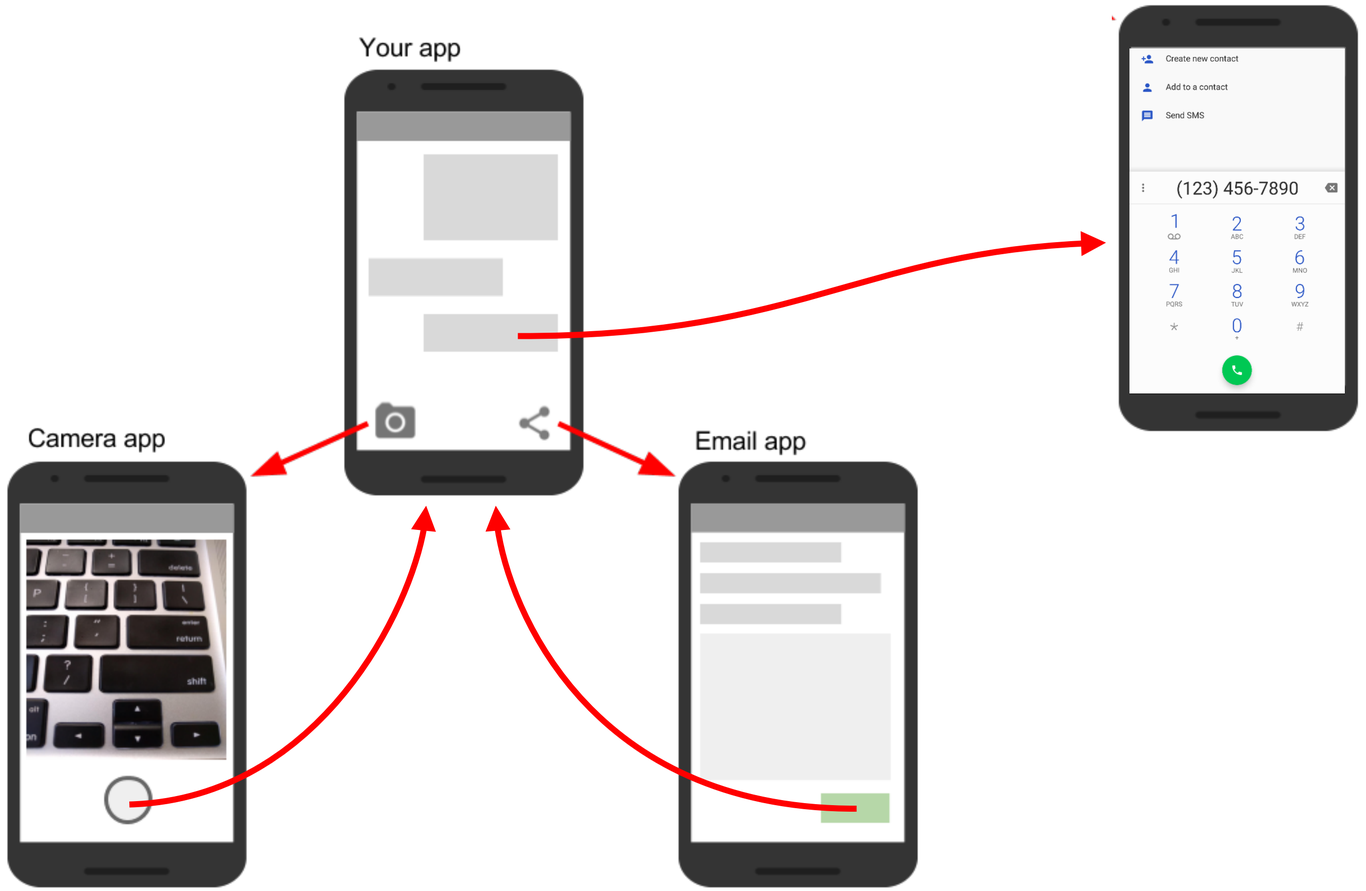
- Key part of an Android app
- Provides the window in which the app draws its UI
- One screen per activity
  - *Each app has multiple screens => multiple activities*
- Activities registered in the app's manifest



https://xamarin-android-guide.peruzal.com/activities/

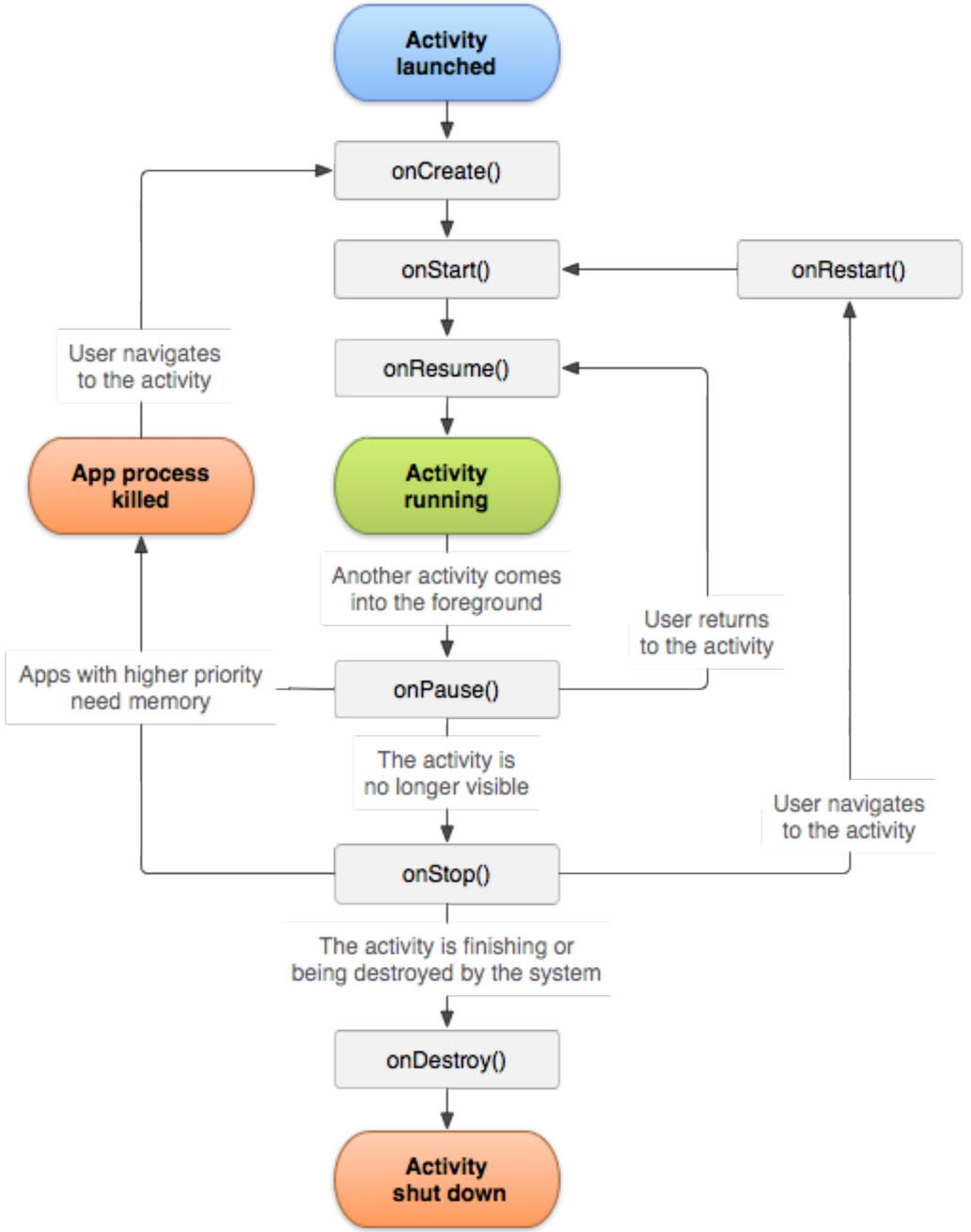
Inter-App Communication via Intents

- An app can cause another app's component to start
  - *E.g., WhatsApp takes a photo with the Camera app*



Inter-App Communication via Intents

- An app can cause another app's component to start
  - *E.g., WhatsApp takes a photo with the Camera app*
  - *Activation is via an Intent object (think of it as an "active message")*
- To start activities and services: <action, URI>
  - *E.g., <ACTION_DIAL, "tel:123456789"), <ACTION_WEB_SEARCH, "app developers")*
- To register for broadcast announcements: <announcement>
  - *E.g., <ACTION_BATTERY_LOW>, <ACTION_POWER_DISCONNECTED>*
- For content providers: pull instead of push
  - *Activated by request from a ContentResolver*



https://developer.android.com/guide/components/activities/activity-lifecycle

iOS Lifecycle

- Very similar to Android
  - *onCreate on Android → viewDidLoad on iOS*
  - *onStart → viewWillAppear*
  - *onResume → viewDidAppear*
  - *onPause → viewWillDisappear*
  - *onStop → viewDidDisappear*
  - *onDestroy → dealloc*
- iOS app may enter a "Suspended" state after being in the background
  - *remains in memory but is not executing code*
  - *Android may keep the process but doesn't have an explicit "Suspended" state*

MAD: Modern Android Development

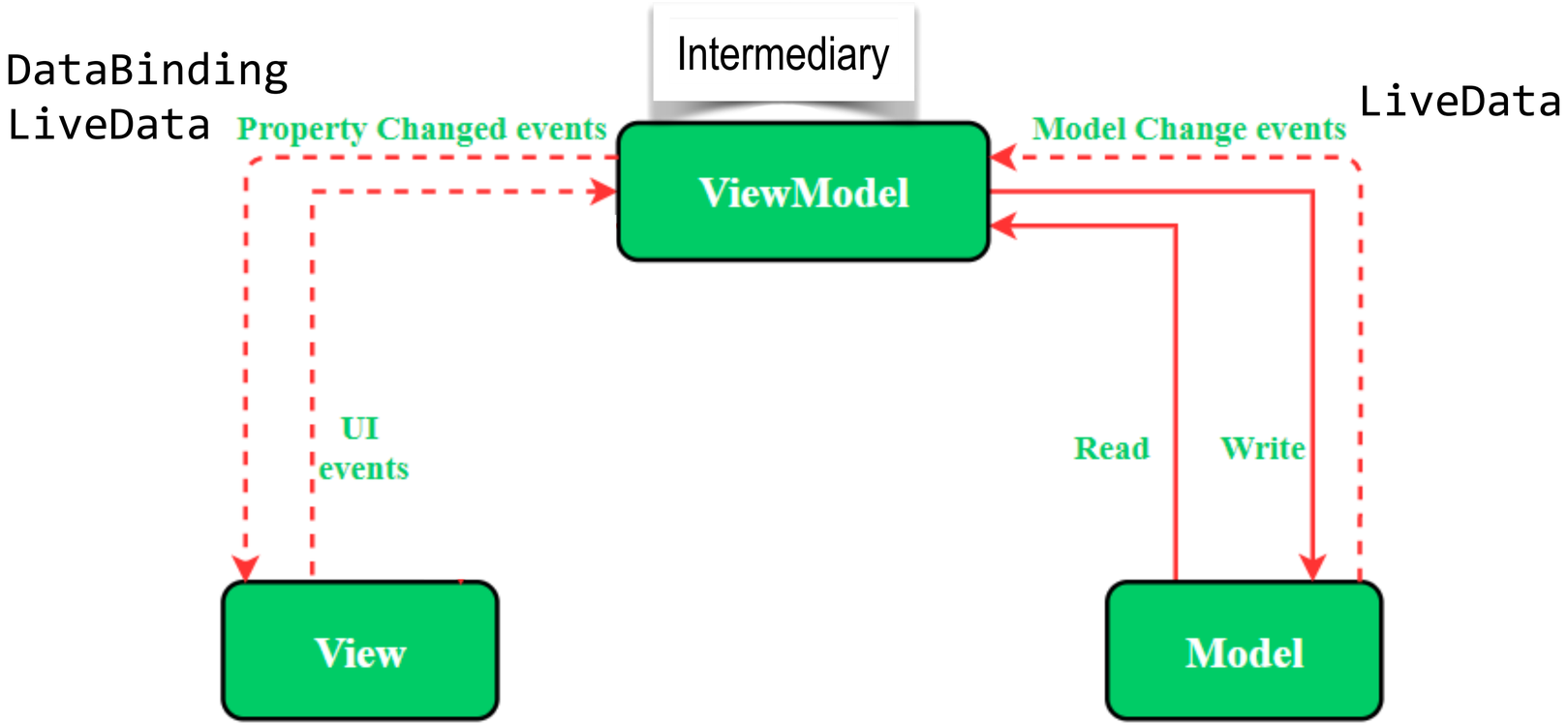
- MAD = languages, libraries, best practices for Android devs
- Kotlin = primary programming lang for Android
  - *concise syntax*
  - *null safety features*
  - *coroutines for asynchronous programming*
  - *extension functions instead of inheritance*

```kotlin
fun String.isEmailValid(): Boolean {
    val emailPattern = Regex(
        "^[A-Za-z0-9._%+-]+@"+
        "[A-Za-z0-9.-]+\\.[A-Za-z]{2,6}$"
    )
    return emailPattern.matches(this)
}
```

- Jetpack
  - *libraries and tools that provide abstractions to reduce boilerplate code*
  - *components for architecture, UI, behavior, and foundations*

The MVVM Design Pattern

MVVM


DataBinding
LiveData
Intermediary
LiveData
Property Changed events
Model Change events
ViewModel
UI
events
Read
Write
View
Model
User interface
Application data
(domain knowledge, business logic, data)
- unaware of the View

Observability and Data Binding in Android

- Observability
  - = *ability of the UI (View) to observe changes in the app's data state (ViewModel)*
  - *e.g., LiveData is an observable data holder class*
- Data binding connects observers to observables
  - *e.g., UI components bound to data sources in the ViewModel*
  - *when the data changes, the bound UI component automatically updates*
  - *no explicit coding, DataBinder performs the update*
- UI layouts bind UI elements to data objects from the ViewModel
  - *reduces boilerplate code, creates readable, declarative layouts*

Benefits of MVVM

- Decouples UI logic from business logic
  - *View is conceptually a function of the ViewModel => can be entirely computed from the ViewModel every time*
  - *e.g., app may run in background and need no UI — can destroy View and re-create from the ViewModel whenever needed*
- Reactiveness leads to responsive, dynamic UIs
- Testability
- Lifecycle Awareness
- Maintainability, evolvability, reuse

Outline

- Mobile Devices
- Mobile Operating Systems
- Mobile Infrastructure & Services
- Mobile Applications
- Mobile User Experience

Mobile User Experience (UX)

User Experience (UX)

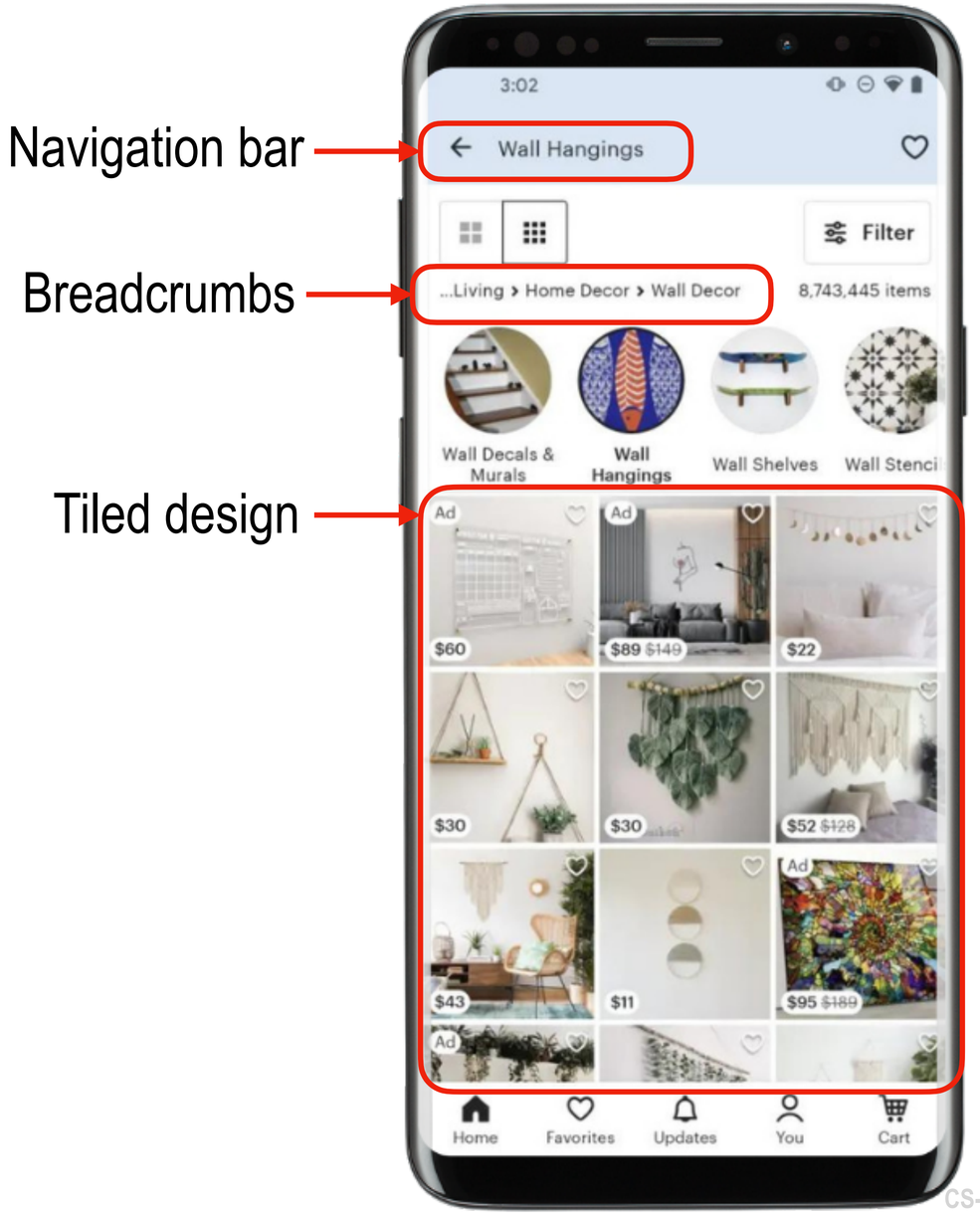
- UX≠UI
  - *UI = the visual elements of an app, such as layout, colors, and typography*
  - *UX = the overall feel and efficiency of using the app*
- Intuitive, responsive, navigable, usable interface for the app
- Think who is your audience
  - *personas*
  - *user stories*

"If you think good design is expensive, you should look at the cost of bad design."

(Ralf Speth, former CEO of Jaguar Land Rover)

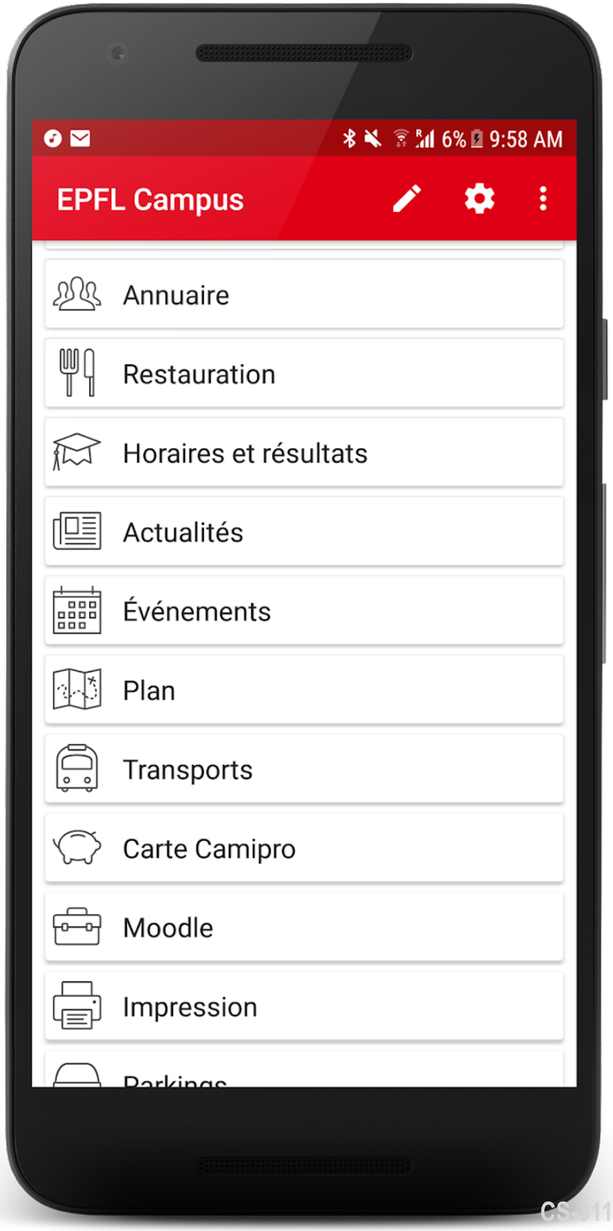
Basics

- Navigation
- Personalization
- Readability
- Simplicity and clarity



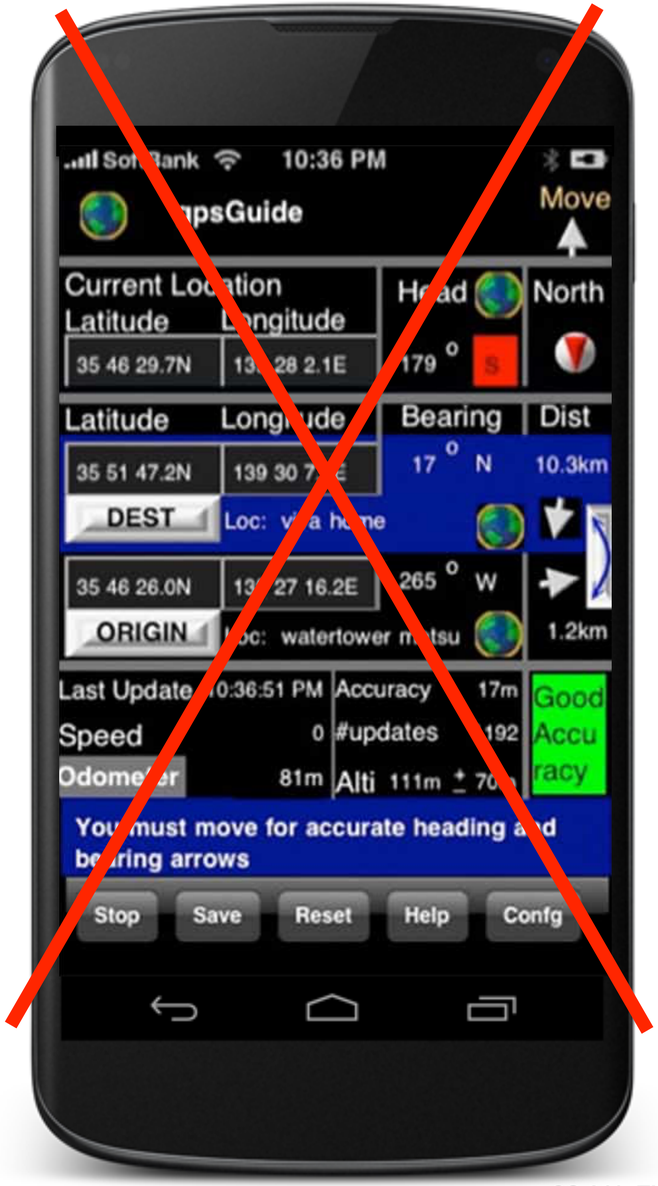
Basics

- Navigation
- Personalization
- Readability
- Simplicity and clarity



Basics

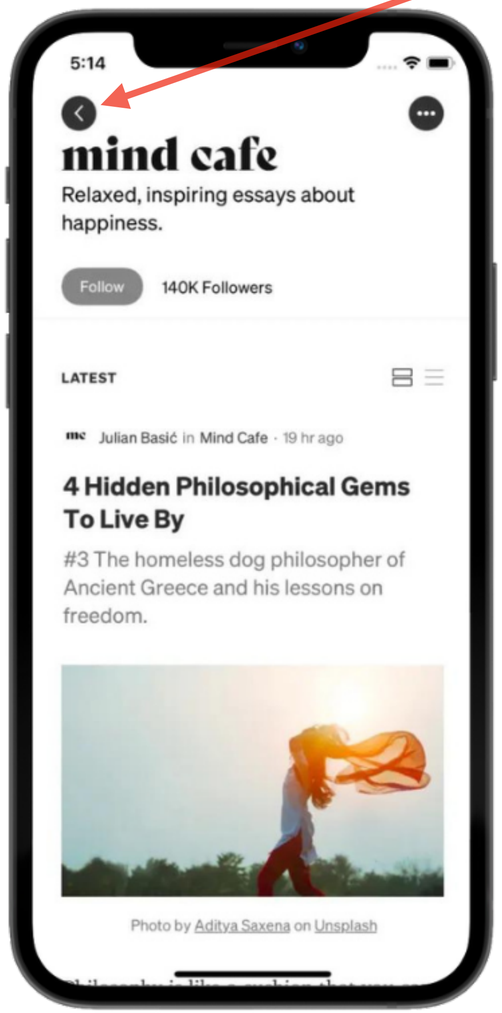
- Navigation
- Personalization
- Readability
- Simplicity and clarity

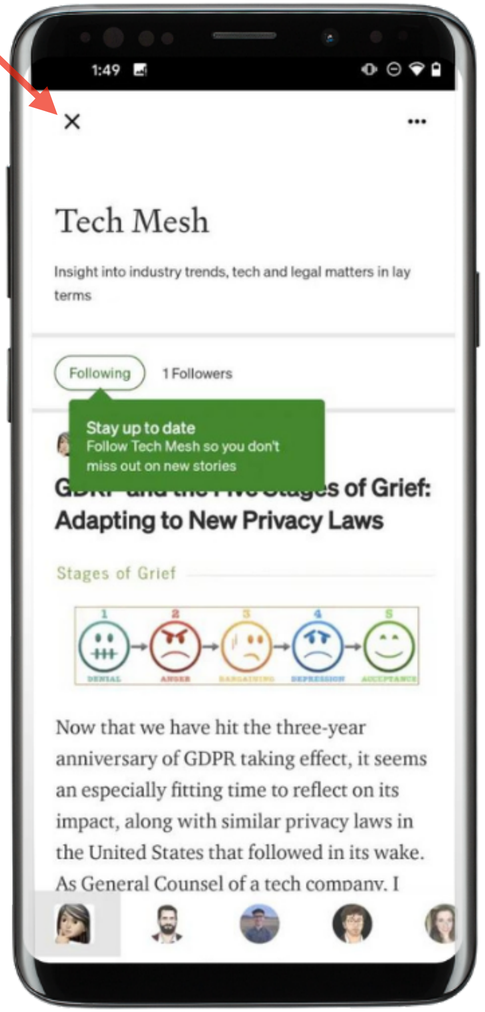


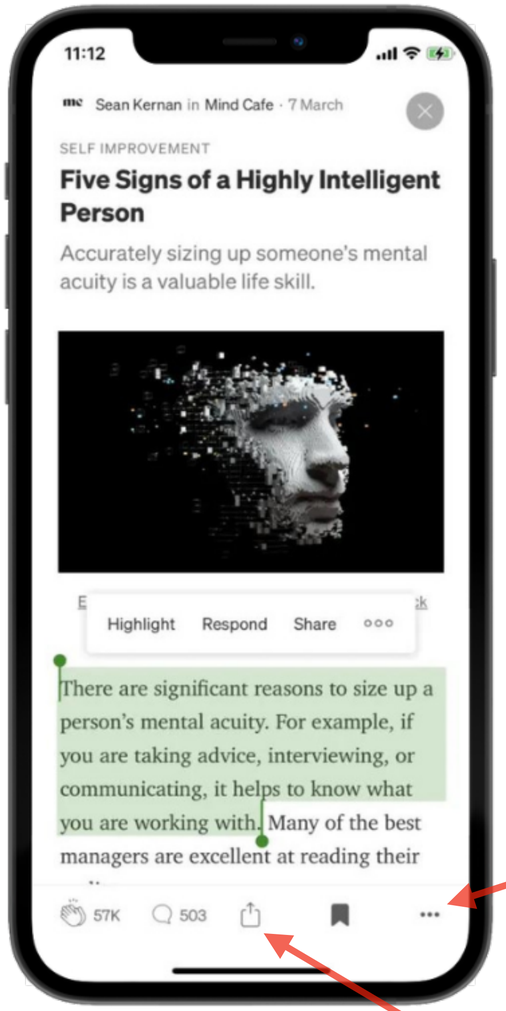
Usability

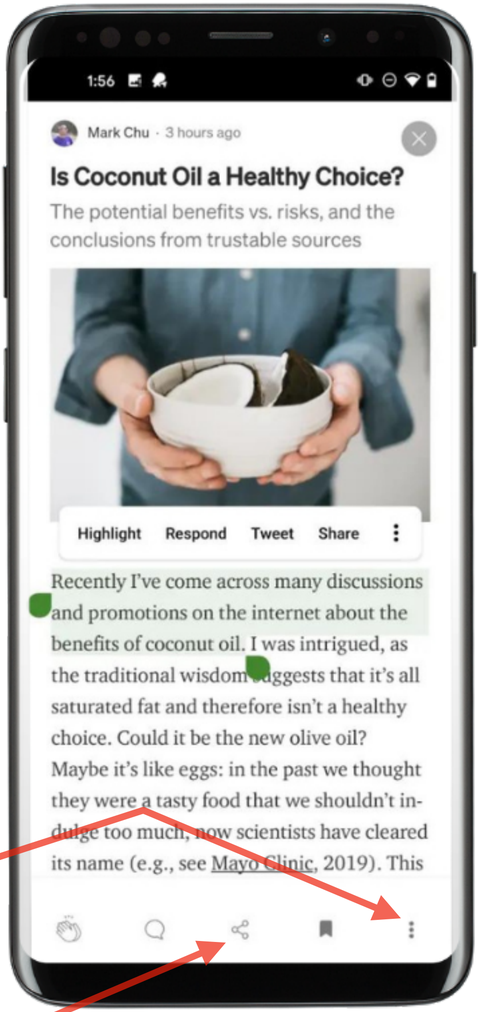
- Think of the thumb zones
  - *49% of people use a single thumb to operate various functionalities on their smartphones*
  - *Place actionable items in the most accessible zone*
- Leverage gesture controls
  - *Long-press, hold, swap, drag, …*
- Augmented reality, voice controls, etc.
- Use what your users expect on that device
  - *adopt the device's look & feel*

Embrace the Platform's Look & Feel

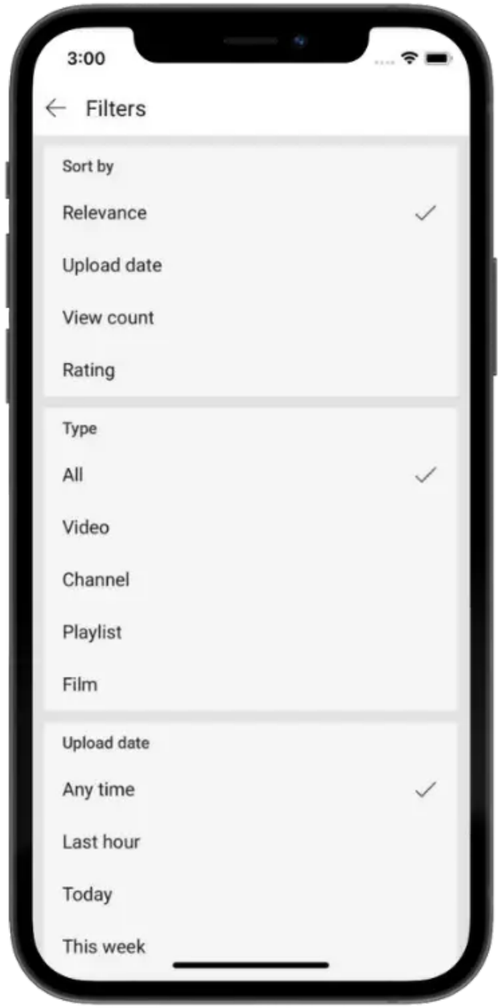


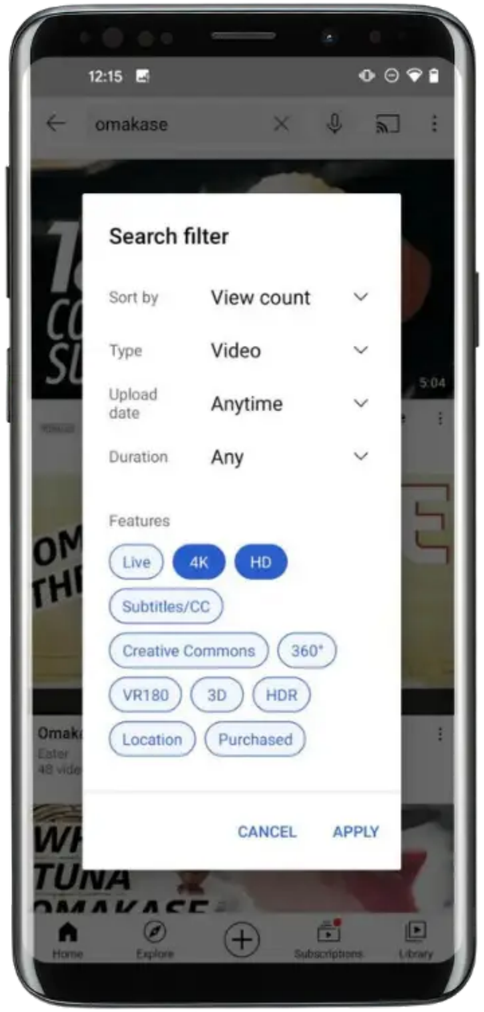


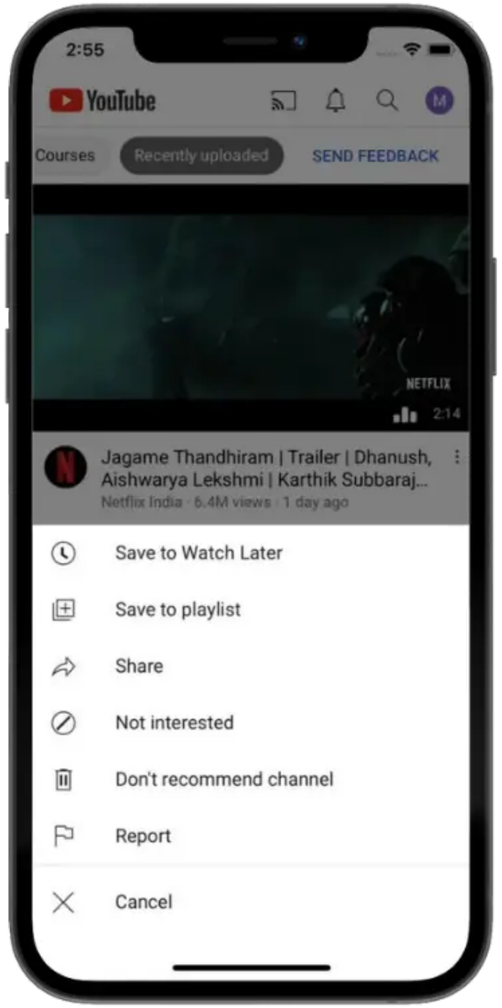


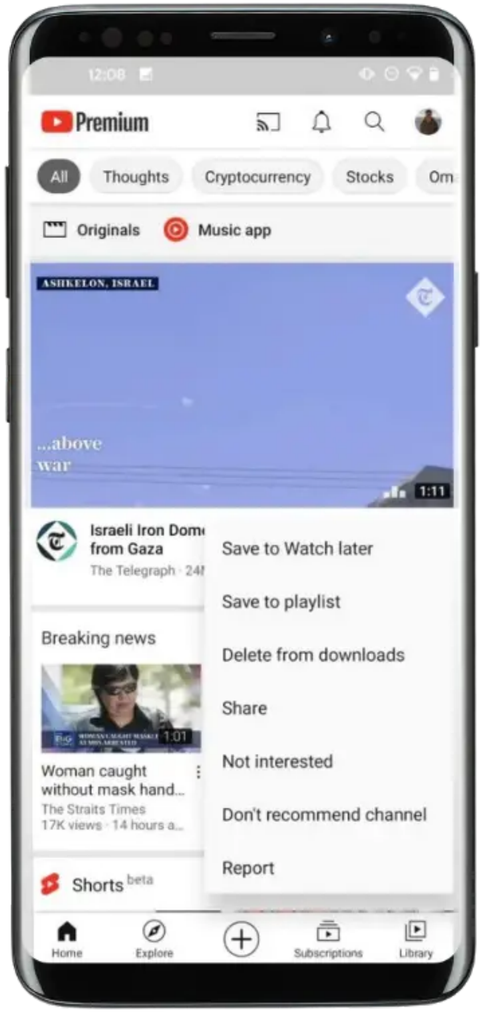


Embrace the Platform's Look & Feel



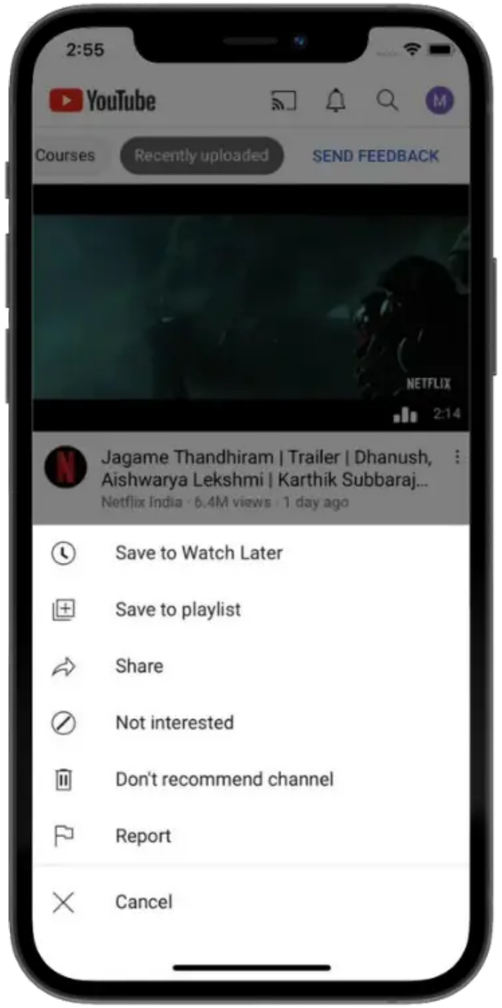


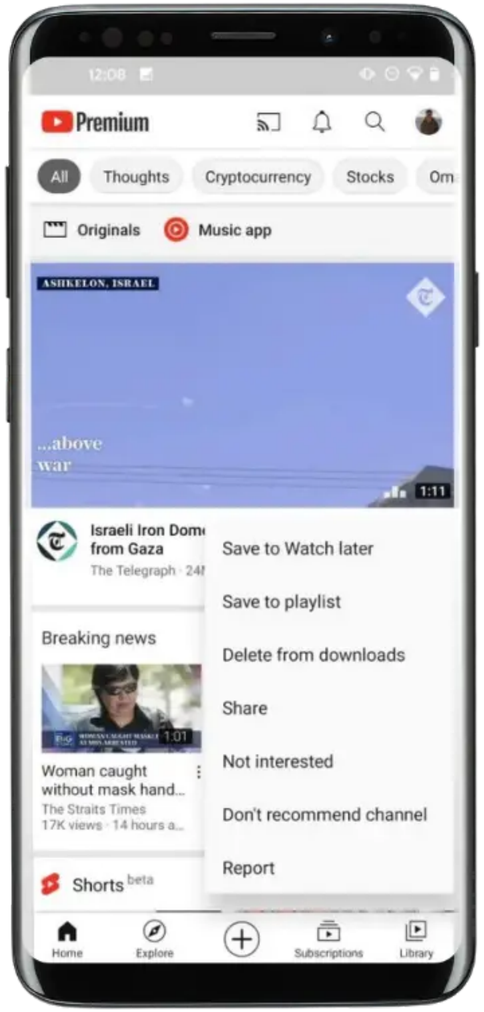


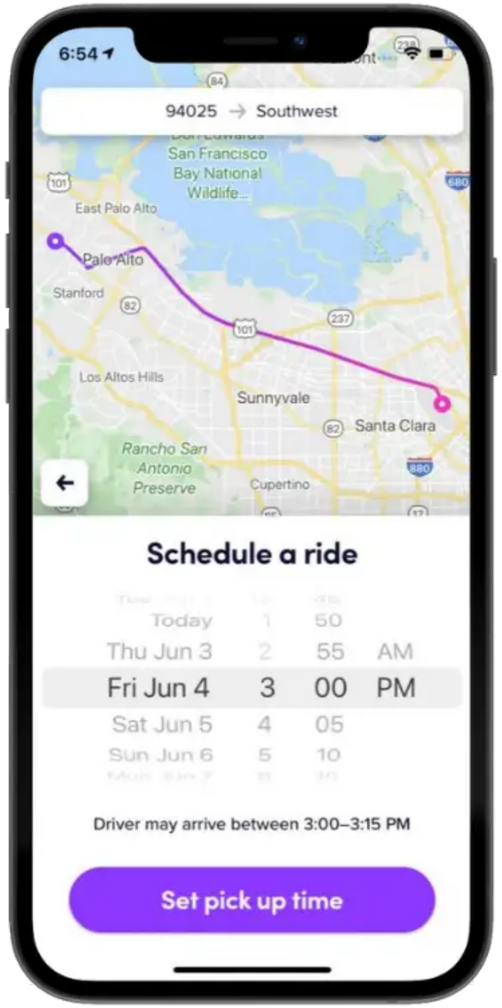


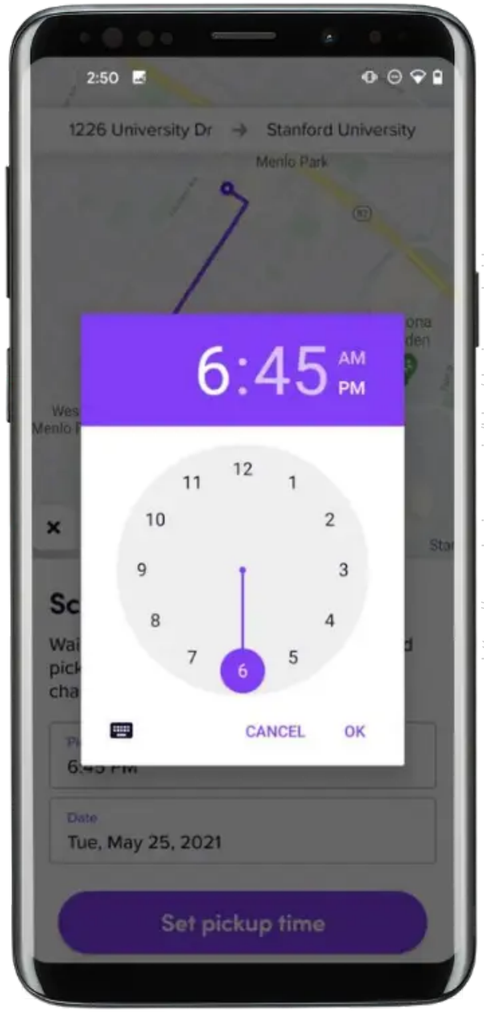
https://www.designware.io/blog/design-android-app

Embrace the Platform's Look & Feel









Responsiveness and Adaptivity

- Responsive design
  - *UI adjusts layout and content dynamically to screen size, resolution, orientation, ...*
  - *goal: maintain readability, usability, and aesthetic appeal*
  - *employ fluid layouts that use relative units for dimensions, flexible grid systems*
- Adaptive design
  - *UX alters functionality and content presentation based on the device's capabilities and user context*
  - *conceptually multiple versions of UI components/layouts specifically designed for different devices, screen sizes, or orientations*

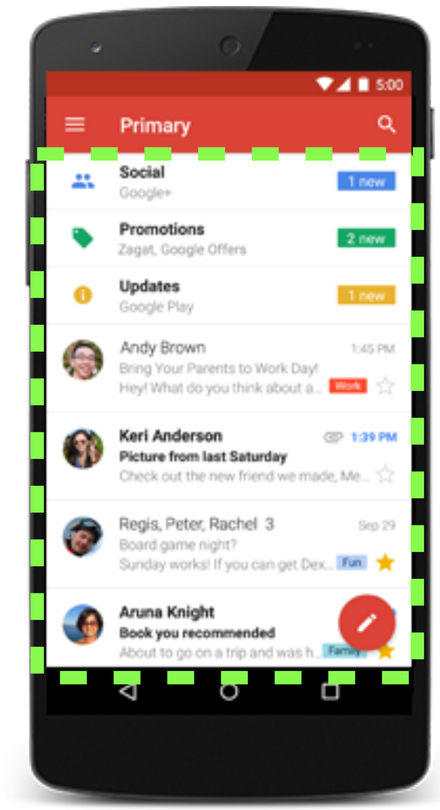
Mobile User Experience

Example: Android UX/UI

Components of an Android UI

- Activities = entry points for user interaction
- Fragments = modular sections of the UI within an Activity
- Fragments allow breaking the UI into reusable components
  - *can dynamically construct different layouts*
  - *can reassemble the UI easily*

Fragments



WebView Component

- app can display web content within the app using WebView
  - *like in a mini web-browser (rendered using Blink, a fork of WebKit)*
  - *e.g., Google authentication*
- you can customize WebView behavior
  - *e.g., app may inject JS into a WebView via* `webView.evaluateJavascript()`
  - *to interact with the web content, apply custom styles, or bridge communication between the web content and the native app code*

User Flow

- = path that a user takes through an app, from the entry point to the final interaction

Placing a Food Order

- Starting Point: User opens PolyFood.
- Search: User enters a cuisine type or restaurant name in the search bar or browses through recommended categories.
- Selection: User selects a restaurant from the search results or recommendations.
- Menu Browsing: User browses the selected restaurant's menu and reviews dish details.
- Adding Items to Cart: User adds chosen dishes to the cart, specifying quantities and any special instructions.
- Review Cart: User reviews the items in the cart, making any necessary adjustments.
- Checkout: User proceeds to checkout, where they confirm the delivery address and choose a payment method.
- Order Confirmation: User reviews and confirms the order details, then submits the order.
- Order Tracking: User receives an order confirmation and can track the order status until delivery.

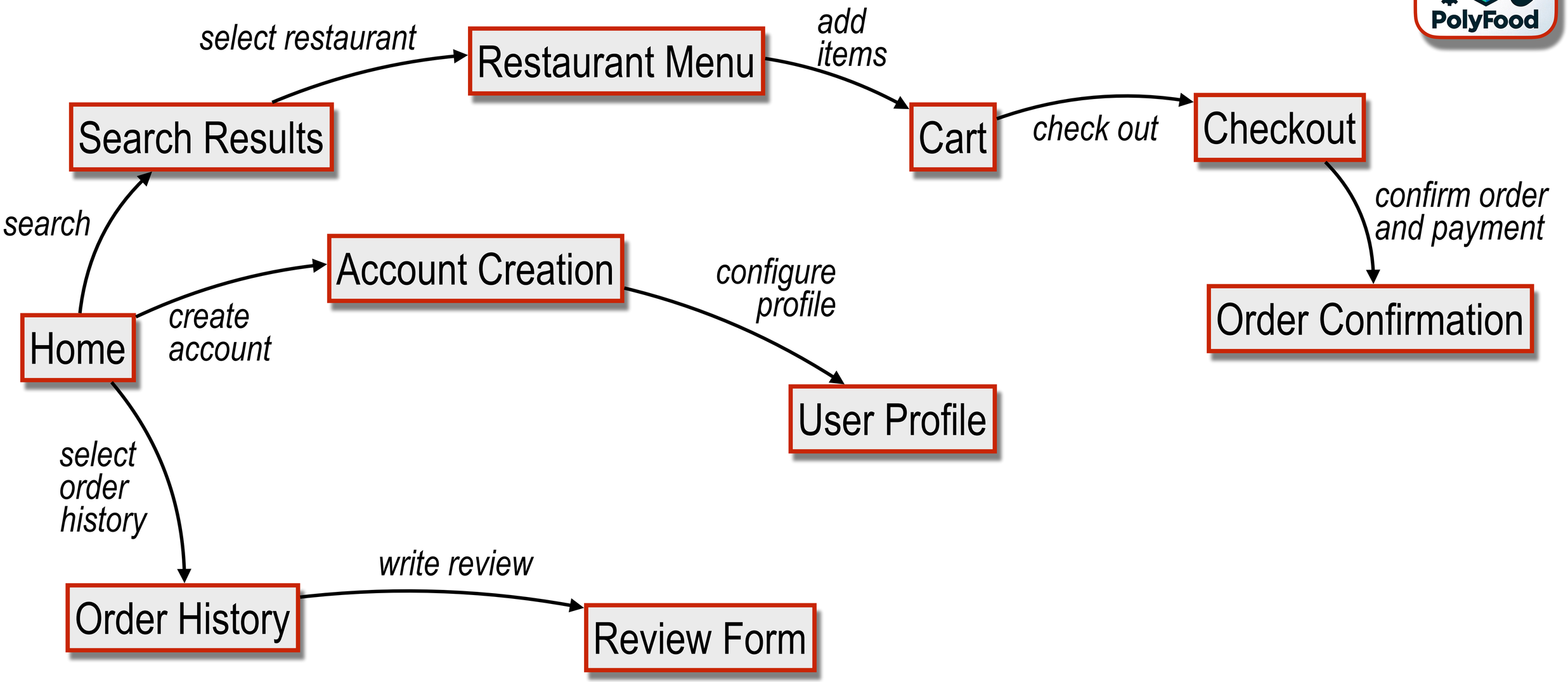
Writing a Review

- Starting Point: User logs into their PolyFood account.
- Order History: User navigates to their order history to find a past order they want to review.
- Select Order: User selects the order or specific dishes they want to review.
- Write Review: User clicks on the "Write a Review" option and is presented with a form to rate and review the restaurant and/or dishes.
- Submission: User fills out the review form, assigning star ratings and writing comments, then submits the review.
- Confirmation: User receives a confirmation message thanking them for their feedback.

Creating a New User Account

- Starting Point: User launches the PolyFood app and selects the option to create a new account.
- Form Filling: User fills in the required information, such as name, email, and password, and optionally, phone number and delivery address.
- Preferences Setup: User sets preferences, such as favorite cuisines or dietary restrictions.
- Verification: User verifies their email (or phone number) through a link or code sent by PolyFood.
- Account Confirmation: User receives confirmation that the account has been successfully created.
- Initial Setup: User is optionally guided through an initial setup process, like setting up payment information or exploring popular restaurants.

Navigation Graph



Jetpack Compose

- Used to build native UIs
  - *Google has been investing a lot in this lately*
  - *Gives a more declarative way (akin to SwiftUI or React) for building UIs*
- Follows a single-activity pattern without fragments (by default)
  - *will see the contrast as part of the bootcamp*

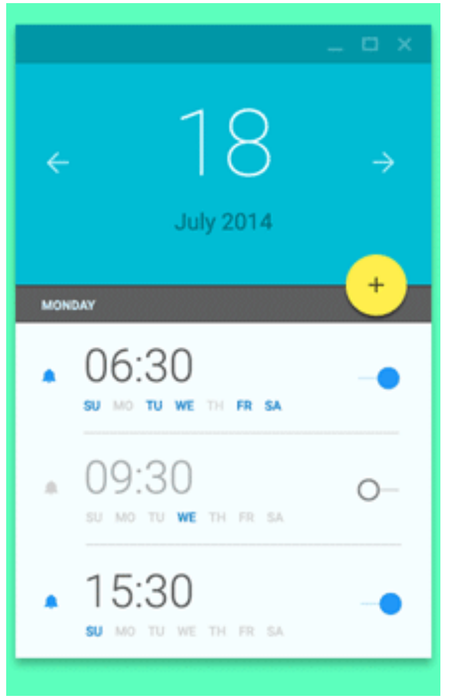
Material Design

- MD = adaptable system of guidelines, components, and tools that support the best practices of UI design
  - *provides components (buttons, sliders, navigation drawers) that adhere to MD philosophy*
  - *various aspects of user input, touch mechanics, visual feedback for actions, etc.*
- Philosophy
  - *digital spaces should mimic the properties of physical materials*
  - *natural surfaces and edges provide cues to users for how elements should move, interact, and be arranged*
  - *seams and shadows provide meaning about what you can touch and how*
  - *bold colors, typography, grids, and imagery to create hierarchy, meaning, and focus*
  - *motion conveys relationships between elements and guides user focus*

Simple Comparison

Flat Design

Material Design



Outline

- Mobile Devices
- Mobile Operating Systems
- Mobile Infrastructure & Services
- Mobile Applications
- Mobile User Experience

Next steps...

- bootcamp session 11:15-12:00
- submit team composition by Tuesday midnight