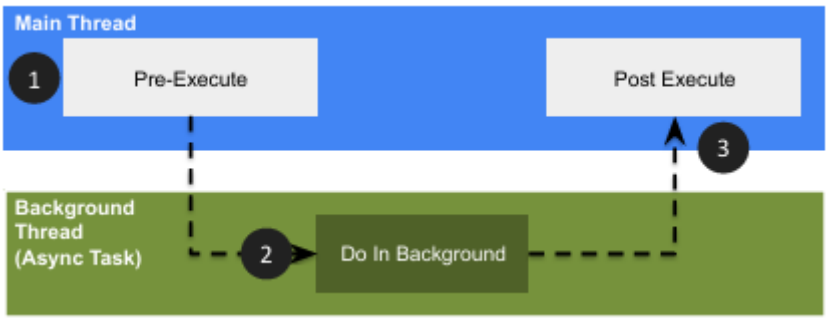
Table of contents

C4.1: Resources and Data Storage

Main Thread (UI Thread) vs Background Thread

	Main Thread (UI Thread)	Background Thread
	• **Renders the ui** • **Handles user input** • **Ui updates must happen on this thread, ui will freeze or lags if main threat busy**	• **perform task that not directly impact the ui** • **Offload tasks to bg threads (file sharing, writing, long calc** **Keeps the main thread free so the app stays smooth** • **Perform bg operations and public results on ui threats**
Use Cases	UI updates and user input handling	Network Requests, database operations, complex calculations, file I/O operations (reading and writing files)
Asynchronous programming	-	*futures, async, await*
		

... Asynchronous programming (3)

Two (2) asynchronous tasks in MySPM mobile application are:

- Eligibility Check – When a user logs in to check their healthcare entitlement (e.g., RM 250 for household, RM 125 for senior citizen), the app sends a request to the backend server in the background. Once the response is received, the result is published on the UI thread to display the user's remaining balance.
- Appointment Booking – When scheduling an appointment with a clinic, the request is sent asynchronously to the server. The app waits for the confirmation in the background, and upon success, updates the UI with the appointment details and reminders without freezing the interface.

1. Futures

A future represents a placeholder for a result that will be available later.

- In MySPM:
- Checking eligibility (e.g., querying user's health subsidy quota from the central database).
- Retrieving health record history from the backend.
- Fetching list of empanelled clinics/hospitals near user location.

Example: When a user taps "Check Eligibility", a future is created for the server response. The UI doesn't freeze — user can still scroll or navigate while waiting.

2. Async
Declaring a function as async means it runs asynchronously and can contain await calls.

- In MySPM:
- async function for loading appointment data from the cloud.
- async method to save updated medical history securely to the backend.
- async request to send push notification tokens to the server.

Benefit: MySPM can run multiple background requests (e.g., fetch health records + location services) without blocking the UI.

3. Await
await is used to pause execution until a future completes, then continue with the result.

- In MySPM:
- await response from ProtectHealth servers when checking remaining healthcare entitlement (RM 250, RM 125, RM 75).
- await Google Maps API to fetch and show nearby clinic locations.
- await confirmation when booking an appointment before updating the appointment reminder UI.

This ensures results are displayed only when data is ready, preventing incomplete or broken UI states.

Data Saving (3)

Data Files • Big files	Shared Preferences • Small, simple settings	SQLite • many structured data
What it is: Saving data as files (text, JSON, PDF, images, audio, etc.) in the device storage. Best for: • Large data (images, documents, downloaded files) • User-generated content (photos, recordings, notes) • Data that must be available offline or shared with other apps Example Scenarios: You download a PDF report and save it so user can open it later without internet. A user records audio, and you	What it is: A simple key–value store, like a small dictionary. Best for: • Storing small, simple settings or flags • Data that configures user experience Limitations: • Can only store primitive原始 data (int, double, String, bool, List<String>) • Not for large/complex data like images or full records Example Scenarios: Save “dark mode = ON”	What it is: A local relational database (tables, rows, columns). Best for: • Large amounts of structured/repeating data • When you need search, filter, query, or relationships • When data is private to the app Example Scenarios: A note-taking app saving 1,000+ notes with title, content, date An offline contacts/address book Caching lots of product data

save it as a .mp3 file. You cache JSON response from an API to show offline content.	Save “username = JohnDoe” Save “first time opened app = true” Save “last volume level = 80%”	from an e-commerce API for fast searching

(i) Treatment Records – SQLite (4 marks)

Reason: Treatment records involve structured data (dates, diagnosis, prescriptions, doctor notes) that need to be queried, updated, and maintained efficiently.

SQLite provides a lightweight relational database on the device, supporting tables, indexing, and queries, making it most suitable for storing and retrieving medical records securely and in an organized way.

(ii) Photo of Identification Card – Data File (4 marks)

Reason: A photo is a large unstructured binary object (BLOB) that does not fit well in shared preferences or database tables.

Storing it as a data file (internal storage) is most suitable, since it allows secure storage of images in the device’s file system with proper access permissions and encryption if required.

(iii) Contact Number – Shared Preferences (4 marks)

Reason: A contact number is small, simple, key-value pair information that doesn’t require complex queries.

Shared Preferences is most suitable as it is lightweight and designed for storing simple user-specific data such as phone numbers, settings, or preferences that can be retrieved quickly by the app.

Internal vs External Storage

Internal Storage

Definition	**Storage that is directly accessible to your app**
Use cases	Storing app data, preferences and cache
Permissions	**No permission is required** to perform read / write operations
Suitable for	• **Shared Preferences** • **SQLite Database** * Due to security and sensitivity of data storage

External Storage

Definition	**Storage that is accessible to multiple apps on the device and outside of the device**
Use cases	Storing large files or media

<table>
<tr><td>Features</td><td>
<table>
<tr><th>Feature</th><th>Android</th><th>iOS</th></tr>
<tr><td>Expandable Storage</td><td>Yes</td><td>No (generally)</td></tr>
<tr><td>Removable</td><td>Possible (SD Card)</td><td>No</td></tr>
<tr><td>File System Access</td><td>More flexible</td><td>More restricted</td></tr>
<tr><td>Permissions</td><td>Requires explicit user permission</td><td>More controlled by the system</td></tr>
<tr><td>Data Sharing</td><td>Easier to share files between apps</td><td>More limited sharing options</td></tr>
</table>
</td></tr>
<tr><td>Considerations</td><td>

Permissions
Ensure that your app has the necessary permissions to access storage
Data security
Consider encrypting sensitive data before storing it
Performance
Optimize file I/O operations to avoid performance bottlenecks
User experience
Provide clear feedback to the user during file operations, such as progress indicators

</td></tr>
</table>

C4.2: Network Operations

Mobile-to-Server Communication

Front-End (User Side)	Network (Bridge)	Back-End (Server Side)
1. Mobile Client The actual app installed on the device. Handles UI, user input, local logic, validation, caching. **Suitable Apps:** Instagram / TikTok → User interacts with buttons, swipes, likes. Banking Apps → Validate form inputs (e.g., check if phone number is valid before sending to server). **2. Local Database** Stores data locally for offline access and to reduce server	Responsible for sending requests and receiving responses using HTTPS, WebSockets, or gRPC. Ensures secure communication. **Suitable Apps:** Zoom → Uses WebSockets for real-time video calls. E-commerce apps (Shopee, Lazada, Amazon) → Uses	**1. API Server** Acts as a middleman between the mobile app and databases. Handles authentication, authorization, business logic, routing. **Suitable Apps:** Facebook Login → API server verifies credentials, gives you an access token. Food Delivery Apps (GrabFood, Uber Eats) → Processes orders and routes to restaurants/drivers. **2. Database Server** Stores and retrieves structured data (users, orders, products, transactions). Handles queries, indexing, consistency. **Suitable Apps:**

requests. **Suitable Apps:** WhatsApp → Stores recent chat history locally so you can still read messages offline. Spotify → Stores downloaded songs locally so you can play without the internet. Google Maps → Stores cached maps for offline navigation.	HTTPS/REST APIs to fetch product listings. Mobile Banking → Secure HTTPS for transactions.	Amazon / Shopee → Product catalogs, prices, and stock levels stored in structured tables. Bank Apps → Transactions, balances, account details. **3. Cloud Database** A scalable, distributed database hosted in the cloud. Ensures global access, redundancy, scalability. **Suitable Apps:** Netflix / YouTube → Needs cloud DB to handle millions of users streaming worldwide. TikTok → Stores user videos, profiles, likes in distributed databases. Firebase-based Apps (startups, chat apps) → Developers use Firebase for quick scaling and global access.

Server Options - Factors to Consider

- Before choosing a server option, you need to evaluate:
- Project Scope (项目范围) – How complex is your app? How much data will it handle?
- Small MVP vs. Large-scale global app
- Budget (预算) – How much can you spend on development and ongoing maintenance?
- Team Expertise (团队专业知识) – Do you have backend/server-side skills in-house?
- Scalability Requirements (可扩展性要求) – Do you expect rapid growth or steady/predictable traffic?
- Time-to-Market (上市时间) – Do you need to launch fast or do you have time to build robust infrastructure?

Server Options (3)

Term / Aspect	DIY (Do-It-Yourself)	Subscribe (BaaS / Managed Services)	Mix & Bang (Hybrid)
	Role: Full control over infrastructure, data, and customization. **Advantages:** Maximum control, compliance with regulations, long-term cost optimization. **Disadvantages:** High initial investment, requires strong backend skills, ongoing maintenance burden.	Role: Use Backend-as-a-Service (BaaS) providers (e.g. Firebase, AWS Amplify, Supabase). **Advantages**: No server management, built-in features (AI, auth, push notifications), quick launch, auto-scaling. **Disadvantages:** Vendor lock-in, limited customization, costs may rise as usage grows.	Role: Combine DIY and BaaS. Use providers for some parts, custom-build others. **Advantages:** Flexibility, scalability, balance between control and speed. **Disadvantages:** More complex integration requires moderate expertise. **Best Suited For:** • Growing startups scaling from MVP to production → Start with BaaS, migrate critical parts (payments,
	Best Suited For: • Large enterprises or fintech apps (e.g.	**Best Suited For:** • Early-stage startups or solo developers →	

	banking apps) → Need full control, security, compliance. • Projects with unique/custom requirements → Custom algorithms, proprietary pipelines. • Apps with predictable high traffic → Long-term cost optimization by owning servers. • Not Ideal For: Small teams, early-stage startups.	Fast MVP launch. • Social media apps with unpredictable growth (e.g. a viral app) → Auto-scaling prevents crashes. • Projects needing advanced features quickly → AI, NLP, analytics out-of-the-box. • Proof-of-concept or MVPs → Validate idea quickly and cheaply.	recommendations) to own servers. • Apps with mixed needs → Sensitive data handled in-house, commodity services (notifications, storage) on BaaS. • Teams with moderate backend skills → Enough expertise to manage key modules but not everything. • Scalability-focused projects → Can scale certain modules independently.
Server Maintenance	Full responsibility for setup, monitoring, patching, scaling	No server management; handled by provider	Shared – provider handles commodity parts, team manages critical modules
Data Security & Compliance	Maximum control; can enforce strict compliance (GDPR, HIPAA, PCI-DSS)	Relies on provider's compliance certifications; less granular control	Sensitive data can be in-house, rest outsourced → flexible balance
Customization	Unlimited; tailor everything (custom algorithms, APIs, workflows)	Limited; bound by provider's features and restrictions	Moderate; custom for critical areas, use provider for standard features
Initial Cost	High (hardware, licenses, engineers)	Low (pay-as-you-go)	Medium (BaaS for most, DIY cost for selective parts)
Ongoing Cost	Potentially lower long-term (own infra, optimized scaling)	Costs scale with usage (risk of bill shock)	Balanced; can control expensive parts by moving them in-house
Speed of Development / Time to Market	Slow (setup + coding everything from scratch)	Very fast (plug-and-play APIs, auto-scaling)	Medium (quicker than DIY but more work than pure BaaS)
Scalability	High, but requires skilled team to set up clusters/load balancing	Automatic scaling managed by provider	Flexible – can scale commodity services via BaaS and optimize in-house
Technical Skills Required	Very high (DevOps, backend, security, networking)	Low (just app developers, minimal backend skills)	Moderate (need to integrate APIs + manage some infra)
Best Use Cases	Banking, healthcare, government apps (compliance-heavy, sensitive data, predictable traffic)	Startups, MVPs, viral apps (fast launch, unpredictable growth)	Growing startups, mid-stage apps, projects with mixed sensitive + general needs

The most suitable backend model for MySPM is the **Hybrid (Mix and Bang) model**.

Sensitive healthcare data such as treatment records and identification details should be stored on government-managed servers (DIY) to ensure compliance with security and privacy regulations.

At the same time, less sensitive services such as appointment reminders, push notifications, and analytics can rely on BaaS providers for faster deployment and scalability. This hybrid approach

balances security, compliance, cost-effectiveness, and scalability, making it ideal for a nationwide healthcare initiative like MySPM.

C5: Location-Based Services

**Source of Location Data Type (2)

Type	GPS	Network-Based
Data Accuracy	High accuracy (up to a few meters) in outdoor environments.	Lower accuracy (hundreds of meters) depending on cell tower/Wi-Fi density.
Speed	Slower, may take several seconds to minutes (needs satellite lock).	Faster, almost instant location fix using nearest towers/Wi-Fi.
Power Consumption	High. due to continuous satellite signal processing.	Lower as it uses existing mobile/Wi-Fi signals.
Environment	Outdoor; poor performance indoors or underground.	Indoor and outdoor. especially in urban areas with many towers/Wi-Fi hotspots.

Battery Drain factor (3)

Aspects	Details
Accuracy	• Refers to the **precision of the location data** • **Higher accuracy** often requires **more power and resources**
Frequency	• Determines **how often the device's location is updated** • A **higher frequency** can **drain the battery faster**
Latency	• Measures the **delay between the actual location change** and the **time it takes for the app to receive the updated location** • The **lower latency** will **drain battery faster**

Location-based services factor to consider (3)

Factors	High	Medium	Low
Accuracy	• GPS-level, ~1 - 5m • Needed when exact positioning matters • Examples: turn-by-turn navigation, ride-hailing pickup, fitness tracking (running / cycling routes) • Trade-off: Drains battery faster, slower to lock indoors	• ~10 - 50m • Good enough when you just need to know the general area • Examples: food delivery tracking (customer view), local business search, "nearby friends" features • Balance between precision and efficiency	• 100m - 1km • Suitable when approximate location is fine • Examples: weather apps, location-based ads, regional analytics • Very battery-friendly
Frequency	• Every second or	• Every few seconds /	• Every few minutes /

	continuous • Needed for real-time tracking • Examples: navigation apps, live sports tracking, emergency response • Trade-off: heaving battery and data usage	minutes • Works when updates don't need to be instant but should feel "live" • Examples: delivery driver updates, ride-hailing driver en route, fitness apps logging every 5 - 10 seconds	hours • Best for background or periodic updates • Examples: weather refresh, location check-ins, passive analytics • Very efficient, minimal battery impact
Latency	• Minutes delay • Fine for non-urgent, background tasks • Examples: weather updates, location-based reminders, background analytics)	• A few seconds delay • Acceptable when slight delay doesn't harm user experience • Examples: food delivery tracking, social apps showing "last seen nearby"	• Instant / near real-time • Needed when immediate response is critical • Examples: ride-hailing pickup, emergency services, AR gaming (like Pokémon Go)

Types of Accuracy (4)

Type	Precision	Hardware Use	Power
High Accuracy	Most precise location possible	GPS	High
Balanced Power Priority	City block (100m)	Wi-Fi or cell tower	Less
Low Power	City-level (10km)	Wi-Fi or cell tower	Less
No Power	Receives locations from other apps	None	Very minimum

Location Strategies (3)

Global Positioning System (GPS)	• **High Accuracy**
	  ○ Provides **precise location data**
	• **Power Consumption**
	  ○ Can **drain the battery quickly,** especially when used continuously
	• Use Cases
	  ○ Navigation apps, outdoor activity tracking, geofencing

	Extra info: • Best for: Outdoor, high-accuracy needs • Examples: Navigation, fitness tracking, ride-hailing 打车服务 • Strengths: Very precise (within a few meters) • Weaknesses: High battery drain, slower to get a fix, poor performance indoors or in dense urban areas 密集城区

Network-Based Location	• **Lower Accuracy**
	  ○ Relies on cell tower and Wi-Fi triangulation 三角定位
	• **Lower Power Consumption**
	  ○ Less demanding on the device's battery
	• Use Cases
	  ○ General location-based services, weather apps, local search

	Extra info: • Best for: Indoor or urban environments when approximate location is enough • Examples: Weather apps, location-based ads, check-ins • Strengths: Faster response, lower battery usage, works indoors • Weaknesses: Less accurate (50 - 500m range depending on density of towers / Wi-Fi)

Hybrid Approach	• **Balanced Accuracy and Power Consumption**
	  ○ **Combines GPS and network-based location**
	  ○ Network-based for quick approximation
	  ○ Refine with GPS if needed
	• Use Cases: Most mobile apps that require location data

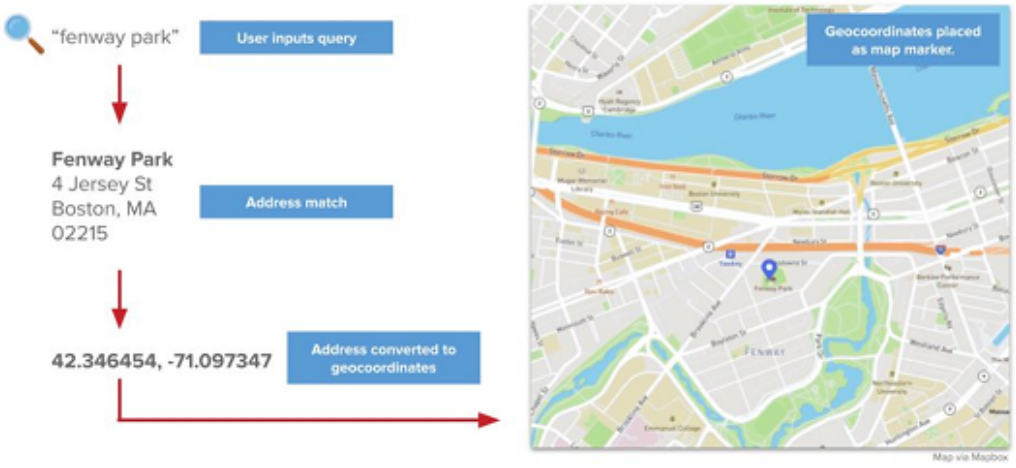
Types of Location Permission (2)

Task / Aspect	**Foreground Location**	**Background Location**
Share Location	• Once • Predefined period	Constant
Visibility of UI	Visible	Not visible
Show persistent notification	Yes	No
Examples	• Within a navigation app, a feature allows users to get turn-by-turn directions • Within a messaging app, a feature allows users to share their current location with another user	• Within a family location sharing app, a feature allows users to continuously share location with family members • Within an IoT app, a feature allows users to configure their home devices such that they turn off when the user leaves their home and turn back on when the user returns home
Insert Permission to Manifest	• Foreground - Coarse 粗略 Location ○ Allows an app to	• For Android 10 (API 29) and above • *<uses-permission*

File	access approximate location ○ Returns a location with an accuracy approximately equivalent to a city block 返回的位置精度大约相当于一个城市街区 ○ *<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />* ● Foreground - Fine Location ○ Allows an app to access precise location ○ *<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />*	*android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />*

Location Services

Services	**Details**
Geocoding	● **Convert address into geographic coordinates (latitude and longitude) or the reverse** ● Geocoding APIs can **compensate for ambiguous address** 补偿模糊地址 **- ones that misspelled or inaccurate** ● Examples: ○ Street addresses can change ○ Address coordinates won't ○ When you type an address into a food delivery app and it finds the exact spot on the map, that is geocoding ● Geocoding **integrates and aligns address information with geographic codes to verify the address** 地理编码将地址信息与地理代码整合和对齐，以验证地址 ● Possible errors: ○ No location data provided ○ Invalid latitude or longitude used ○ No geocoder available ○ No address found

Geolocation	• **Use the device location detection hardware to determine current location**
	• Uses GPS, Wi-Fi, cell towers or IP address to estimate latitude and longitude
	• Example
	  ○ When your phone shows "You are here" on Google Maps, that is geolocation
	• Use Cases
	  ○ Navigation
	  ○ Ride-hailing pickup 乘车接送
	  ○ Location-based reminders
	  ○ Geo-fencing 地理围栏

Map	• **Display maps and markers**
	• Visual representation of geographic data, often layered with geolocation and geocodiing results 地理数据的可视化显示，通常与地理定位和地理编码结果分层显示
	• **Uses mapping libraries or providers to render roads, landmarks and overlays**
	• Examples of map service providers that can be used to integrate maps into Flutter app:
	  ○ Google Maps
	  ○ Mapbox
	  ○ Apple Maps
	  ○ Here
	  ○ Azure Maps
	  ○ OpenStreetMap
	  ○ TomTom
	• Use Cases
	  ○ Navigation
	  ○ Route planning
	  ○ Data visualization
	  ○ Location-based analytics

*Location Best Practices (4)

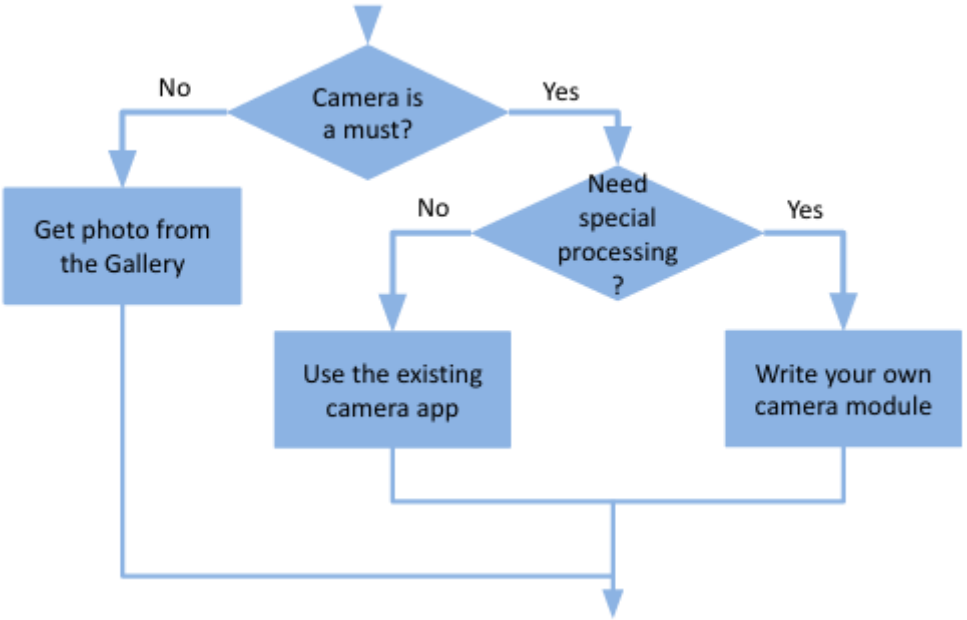
Aspects	Details
Remove location	• **Use *Geolocator.removeListener()***

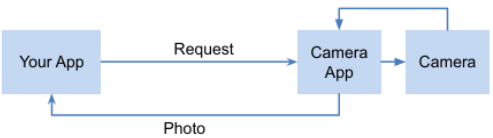
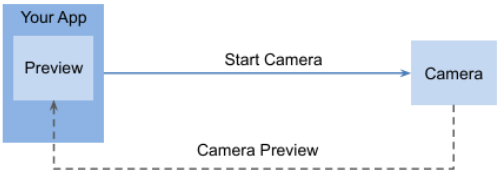
updates	○ When you no longer need to track the user's location, remove the listener to conserve battery and resources ● **Check app state** ○ Only request location updates when the app is actively using them ● **User interaction** ○ Trigger location updates based on user actions, such as tapping a button or opening a specific screen
Set timeouts	● **Reasonable timeouts** ○ Set appropriate timeouts for location updates to prevent excessive battery drain ● **Dynamic timeouts** ○ Adjust timeouts based on the app's current state and user activity ● **Background mode timeouts** ○ If using background location updates, consider setting longer timeouts to balance accuracy and power consumption
Batch Requests	● **Combine multiple requests** ○ If your app needs multiple location updates, combine them into a single request to reduce network overhead and battery usage ● **Throttle Requests** ○ Limit the frequency of location updates, especially when the user is not actively using the location-based features
Passive location updates	● **Minimize battery consumption and improve user experience** ○ Rely on system-provided location data rather than continuously tracking the user's location

C6: Specialized Instrument and Devices

Camera

- Basic app features:
  - **Visual content creation** - take photos and videos
  - **Document scanning**
  - **QR code scanning**
- Advanced features:
  - **Augmented Reality (AR)**: Cameras can be used to overlay digital content onto the real world, creating immersive experiences
  - **Image Recognition**: Apps can recognize objects, text or faces in images to provide additional information or functionality
  - **Facial Recognition**: This technology can be used for secure authentication, unlocking devices or enabling personalized experiences
- Considerations:



Implementation	**Use an existing camera app**	**Build your own camera function**
Complexity	Simple implementation - using existing packages	Complex implementation - write your own code
How	Obtain a photo from the camera app or the gallery	Create a camera preview
Permission	No need permission	Request the Camera permission
Diagram		
Use Cases	• Profile picture upload (social apps, e-commerce, HR apps) ○ User just needs to snap	• Augmented Reality (AR) apps (IKEA Place, Pokémon GO, Snapchat filters)

	or pick a photo ● Attach receipts or documents (banking expense tracking, insurance claims) ○ No need for live edge detection or AR overlays ● Basic QR code scanning (event tickets, Wi-Fi setup, payment links) ○ Can rely on external scanner or OS-level intent ● Simple photo / video sharing (chat apps, feedback forms) ○ The native camera app already provides capture UI	○ Need live camera feed to overlay 3D models or effects ● Document scanning apps (CamScanner, Adobe Scan) ○ Require edge detection, perspective correction and batch capture ● Real-time QR / barcode scanning (GrabPay, delivery apps, inventory systems) ○ Continuous scanning without leaving the app ● Facial recognition / biometric login (banking apps, secure enterprise apps) ○ Needs direct access to frames for authentication ● Custom camera UI (Instagram, Tiktok, VSCO) ○ Filters, stickers, timers, mutli-shot modes - all require control over preview and capture pipeline ● Object or text recognition (Google Lens, translation apps) ○ Must process frames in real time for OCR or classification

Media

- Plays local (assets) and external (streaming) files
- Supports any media codec that is provided by the platform and those that are device-specific
- Recommendation: use core media formats
- Use *video_player* plugin to play videos stored on the file system, as an asset or from the internet (not available on Linux and Windows)

Core media file formats:	
Audio	.3gp, .mp3, .mp4, .mid, .wab, .ogg
Picture	.jpg, .gif, .png, .bmp
Video	.3pg, .mp4

Sensors

Motion Sensors

Sensor	**Details**
Accelerometers 加	● Provides data on the rate of change in **velocity** - the speed in

速度计	combination with the **direction** of **motion** of an object, excluding gravity ● App features that use it: ○ Step counters / fitness trackers (e.g. Google Fit, Strava) ■ Detect walking, running, cycling by tracking movement patterns ○ Shake-to-trigger actions (e.g. undo typing, refresh content) ■ Apps like Gmail or note-taking apps use shake gestures ○ Fall detection (e.g. health monitoring apps for elderly users) ■ Sudden acceleration changes signal a potential fall ○ Driving behavior analysis (e.g. insurance apps) ■ Detect harsh braking 急刹 or rapid acceleration
Gravity Sensors	● It measures the **direction** and **intensity of gravity** ● Checks the **relative direction** of a device within a space ● App features that use it: ○ Screen orientation control (e.g. auto-rotate in video players or games) ■ Switch between portrait and landscape modes ○ Tilt-based controls 基于倾斜的控制 in games (e.g. racing games, ball maze games) ■ Use gravity vector to steer or balance 转向或平衡 ○ Leveling tools / spirit level apps 水平仪应用程序 (e.g. construction or DIY apps) ■ Check if a surface is flat or tilted 倾斜
Gyroscopes 陀螺仪	● It helps the accelerometer out with understanding which way your phone is **oriented** 它能帮助加速度计了解手机的方向 ● App features that use it: ○ 360° panorama photography (e.g. Google Street View) ■ Track phone rotation to stitch images 跟踪手机旋转以拼接图像 ○ VR and AR experiences (e.g. Google Cardboard, IKEA Place) ■ Detect head movement and orientation ○ Motion-controlled games (e.g. flight simulators, shooting games) ■ Real-time rotation for immersive control ○ Gesture-based navigation (e.g. wrist flick 轻触手腕 to switch apps on wearables ■ Detect subtle rotational gestures 检测细微的旋转手势
Rotation Vector Sensors 旋转矢量传感器	● To **monitor and measure turning movements** ● **Combination of an angle and an axis** ● App features that use it: ○ AR object placement and tracking (e.g. Snapchat filters, AR measuring tools) ■ Maintain stable virtual object orientation ○ Star-gazing apps 观星应用程序 (e.g. SkyView, Star Walk) ■ Align phone with celestial coordinates 将手机与天体坐标对齐 ○ Indoor navigation / orientation (e.g. museum guides, shopping mall maps) ■ Determine user's facing direction ○ Immersive 3D experiences (e.g. educational apps, architectural

	walkthroughs) ■ Rotate and explore virtual environments

Environmental Sensors

Sensor	Details
Barometers 气压计	● It **measures atmospheric pressure** ● It assists the GPS chip inside the device to get a faster lock by instantly delivering altitude data (vertical location) ● App features that use it: ○ Altitude tracking 海拔跟踪 in fitness apps (e.g. hiking, climbing, cycling) ■ More accurate elevation gain / loss 海拔增高/降低 than GPS alone ○ Weather forecasting apps (e.g. AccuWeather, Windy) ■ Use pressure trends to predict storms or changes ○ GPS enhancement (e.g. navigation apps, location-based games) ■ Faster and more precise vertical positioning ○ Drone flight control apps ■ Use barometric pressure for altitude stabilization
Photometers 光度计	● It **sense the amount of ambient light present** ● App features that use it: ○ Auto-brightness adjustment (system-level or custom UI apps) ■ Saves battery and improves visibility ○ Camera exposure control (e.g. photography apps) ■ Adjust ISO and shutter speed based on lighting 根据光线调整 ISO 和快门速度 ○ Reading apps / eBook readers (e.g. Kindle Moon + Reader) ■ Switch between light / dark mode or adjust contrast ○ Smart home apps (e.g. lighting automation) ■ Trigger lights based on room brightness
Thermometers	● It **measures temperature within a mobile device** ● To **avoid overheating of the internal components** ● App features that use it: ○ Device health monitoring apps (e.g. CPU-Z, AIDA64) ■ Warn users about overheating or throttling 节流 ○ Gaming apps (high-performance games) ■ Adjust graphics or frame rate to prevent overheating ○ Battery management apps ■ Alert users when temperature affects battery health ○ Thermal throttling control 热节流控制 in productivity apps ■ Reduce background tasks or limit processing when hot

Position Sensors

Sensor	Details
Orientation	● **Measures the orientation of a device relative to an orthogonal**

Sensors	**coordinate frame** 测量设备相对于正交坐标框架的方向 ● App features that use it: ○ Tilt-based gaming controls (e.g. racing games, flight simulators) ■ Steer or navigate by tilting the device ○ Panorama or 460° photo capture (e.g. Google Street View) ■ Track device orientation to stitch images ○ AR experiences (e.g. Pokémon GO, IKEA Place) ■ Align virtual objects with real-world orientation ○ Virtual tours / 3D walkthroughs (e.g. real estate apps, museum guides) ■ Rotate the phone to explore environments ○ Educational apps (e.g. astronomy apps) ■ Point the device to the sky to identify stars or planets
Magnetometers	● **Determines your location with respect to Magnetic North (or South)** ● App features that use it: ○ Digital compass apps (e.g. hiking, navigation tools) ■ Show direction even without GPS ○ Navigation apps (e.g. Google Maps, Waze) ■ Improve heading accuracy, especially indoors or when GPS is weak ○ AR object placement (e.g. measuring tools, interior design apps) ■ Align virtual objects with real-world compass direction ○ Metal detector apps ■ Use magnetic field changes to detect nearby metal objects ○ Location-based games ■ Determine player orientation for directional gameplay

Other Sensors

Sensor	Details
Biometrics	● **Biometric authentication for multi-factor authentication (MFA)** to verify individual's identity that uses possession of a mobile device ● Verify unique biometric identifier ● App features that use it: ○ Secure login / authentication (e.g. banking apps, password managers) ■ Use fingerprint or Face ID for multi-factor authentication ○ App lock / sensitive action confirmation (e.g. WhatsApp, PayPal) ■ Require biometric verification to open app or approve transactions ○ Digital identity verification (e.g. eKYC in fintech apps) ■ Match biometric data with official documents ○ Personalized experiences (e.g. parental control apps) ■ Different profiles unlocked by different users
Bluetooth Low Energy	● A **short-range wireless communication technology designed**

(BLE)	**for low-power applications** ● **Low power consumption, high data rate** ● Applications: wearables, IoT, beacon technology, healthcare, indoor navigation ● App features that use it: ○ Fitness and health tracking (e.g. Fitbit, Garmin Connect) ■ Sync data from wearables like heart rate, steps, sleep ○ Indoor navigation (e.g. airport or mall apps using BLE beacons 信标) ■ Guide users through complex indoor spaces ○ Smart home control (e.g. Philips Hue, smart locks) ■ Connect to IoT devices for lighting, security, automation ○ Contact tracing / proximity alerts 联系人追踪/近距离警报 (e.g. COVID-19 apps) ■ Detect nearby users for exposure notifications ○ Asset tracking / inventory management (e.g. warehouse apps) ■ Monitor tagged items via BLE beacons
Near-Field Communication (NFC)	● A **short-range wireless technology** ● **Allows two electronic devices to communicate with each other** ● Applications: mobile payments, data exchange, access control, product information ● App features that use it: ○ Mobile payments (e.g. Apple Pay, Google Pay, Samsung Pay) ■ Tap to pay at POS terminals ○ Access control (e.g. hotel key cards, office entry apps) ■ Unlock doors or authentication entry ○ Product information / smart posters ■ Tap to view details, promotions or URLs ○ Transit ticketing 公交票务 (e.g. Touch 'n Go, Octopus) ■ Tap to board buses or trains ○ Data exchange / pairing (e.g. Android Beam, smart device setup) ■ Share files or connect devices instantly
Nearlink	● A **short-range wireless communication technology developed by the NearLink Alliance**, led by Huawei ● **High speed, high reliability, high security, low latency, low energy** ● Applications: wearable, IoT, automotive, consumer products ● App features that use it: ○ Real-time wearable syncing (e.g. smartwatches, fitness bands) ■ Faster and more reliable than BLE ○ IoT device control (e.g. smart appliances, sensors) ■ Low-latency commands and feedback ○ Automotive apps (e.g. car diagnostics, infotainment control)

	■ Seamless interaction with vehicle systems ○ High-speed file transfer (e.g. Huawei Share) ■ Send large files quickly between devices ○ Gaming accessories (e.g. controllers, AR / VR gear) ■ Low-lag input for immersive experiences

Technology	**BLE**	**NFC**	**Nearlink**
Range	Up to 100 meters	Up to 4 cm	Up to 100 meters
Speed	Up to 2 Mbps	Up to 424 Kbps	Up to 900 Mbps
Power Consumption	Low	Very Low	Lowest (60% of BLE)
Latency	Moderate	Very Low	Ultra-low

C7: Mobile Application Packaging and Publication

Checklist to Plan Your App Launch

Checklist	Detail

Developer Program Policies	• Mobile app stores have specific guidelines and policies that developers must adhere to
	• Example of mobile app stores:
	  ○ Google Play Store
	  ○ Apple App Store
	  ○ Huawei App Gallery
	• Policies are designed to maintain a safe, secure and positive user experience
	• **Restricted Content**
	  ○ Have strict rules about the type of content allowed, including explicit content, malware and harmful apps
	  ○ Explicit Content: pornography, sexually suggestive content and nudity
	• **Intellectual Property**
	  ○ App reviewers check for potential IP infringement 知识产权侵权 such as copyright infringement 版权侵权, trademark infringement 商标侵权 and patent infringement 专利侵权
	• **Privacy and Security**
	  ○ Developers must have clear privacy policies that explain how user data is collected, used and shared
	  ○ App stores may impose security requirements 提出安全要求 to protect user data from unauthorized access and breaches
	• **Monetization and Ads**
	  ○ App stores provide guidelines govern in-app purchases and subscriptions
	  ○ Ad monetization and in-app purchases must adhere to guidelines, including fair pricing and transparent advertising
	• **Store Listing and Promotion**
	  ○ App stores control the listing and promotion (marketing materials) of apps through strict review processes, guidelines and algorithms
	  ○ Developers must adhere to these rules to ensure their apps are visible, accessible and successful
	  ○ e.g. setting of user's age groups, regions, etc

Developer Account	• A **publishing account** issued by **Platform Provider** to **developer**
	• Enables developer to **post**, **display**, **offer for sale** and distribute apps through the Platform
	• Access developer tools, resources and support provided by the platform
	• Examples:
	  ○ Android: USD 25 (one time)
	  ○ Apple: USD 99 (per year)
	  ○ Huawei: FOC
	  ○ KaiOS: FOC
	  ○ Tizen: FOC

Localization	• Adaptation of an app to meet the needs of a particular **language**, **culture** or desired population's "look-and-feel" • A successfully localized app is one that appears to have been developed within the **local culture**
Device Compatibility	• Ensures that an app runs efficiently **across mobile devices** or **different configurations** • Ensures app run well on mobile devices of different **hardware**, **software** and **operating systems**
Quality Test: Alpha & Beta	• A process to ensure apps **meet specified regulations** and **standards** • It is a series of techniques that developers employ to **prevent issues from occurring** and **ensure they satisfy the customer** with their finished product

Store Listing	• Preparing for Release
	  ◦ To prepare the app so that users can install and run the app on their Android-powered devices
	  ◦ Release-ready .apk file is signed with your own certificate
	  ◦ Minimum requirements:
	  ▪ Cryptographic keys (digitally signature that is owned by the application's developer)
	  ▪ Application icon
	  ▪ End-user license agreement
	  ▪ Promotional and marketing materials
	• Signing in Release Mode
	 
	• Signing Consideration
	  ◦ Update
	  ◦ Modularity
	  ◦ Code and data sharing
	• Store Listing
	  ◦ Distribution
	  ◦ App size < 150 MB + Expansion up to 2GB
	  ◦ Platforms
	  ◦ Fee or Free

Distribution Channels

Distribution Channel	Detail
Marketplace	• **Centralized platform where users can discover, download and update apps or digital products**

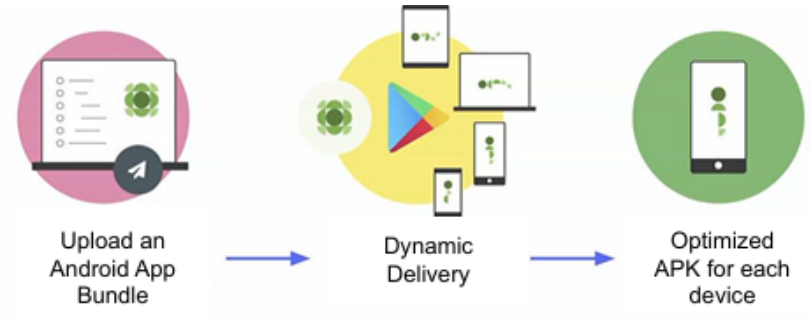
	• Examples:
	  ◦ Google Play Store (Android)
	  ◦ Apple App Store (iOS)
	  ◦ Huawei AppGallery
	  ◦ Microsoft Store
	• Use Cases:
	  ◦ Distributing mobile apps to wide audience
	  ◦ Handling updates, reviews and ratings
	  ◦ Monetization via in-app purchases or subscriptions
	• Benefits:
	  ◦ **Trusted by users**
	  ◦ **Built-in security and update mechanisms**
	  ◦ **Easy discoverability and global reach**

Email	• **Direct distribution of content, links, or app files via email to targeted users**
	• Examples:
	  ◦ Sending beta APKs or TestFlight invites
	  ◦ Delivering promotional content, download links, or onboarding materials
	  ◦ Sharing personalized updates or exclusive access
	• Use Cases:
	  ◦ Private beta testing
	  ◦ Targeted marketing campaigns
	  ◦ Re-engagement or retention strategies
	• Benefits:
	  ◦ **Personalized and direct**
	  ◦ **No need for public testing**
	  ◦ **Useful for controlled or invite-only access**

Website or Server	• **Hosting the app or content on a web server, allowing users to download or access it directly**
	• Examples:
	  ◦ Downloading APKs from a developer's site
	  ◦ Accessing web apps or Progressive Web Apps (PWAs)
	  ◦ Hosting documentation, updates or support tools
	• Use Cases:
	  ◦ Enterprise / internal app distribution
	  ◦ Bypassing app store restrictions (e.g. region locks)
	  ◦ Offering desktop or cross-platform versions
	• Benefits:
	  ◦ **Full control over hosting and updates**
	  ◦ **No marketplace fees or approval delays**
	  ◦ **Ideal for custom or niche deployments**

Distribution Methods - Android

Distribution Method	Detail
Android Go	• OS designed for **entry-level smartphones** with **limited resources** • An initiative that brings the benefits of Android to a wider range of

	users ● Optimized for devices with: ○ **Limited RAM**, typically 1GB or less ○ **Lower processing power** ○ **Less storage space** ● Minimum requirement: ○ Target Oreo (API 26) ○ App size less than 40 MB ○ RAM usage below 50 MB (apps) or 150 MB (games) ○ Start your app under 5 seconds ● Best for: Apps targeting entry-level smartphones with limited hardware ● Suitable circumstances: ○ Your app is designed for **emerging markets** 新兴市场 or **low-spec devices** ○ You need to support devices with **≤ 1GB RAM**, **low CPU** and **limited storage** ○ You are building **lightweight apps** (under 40MB) with **fast startup** and **low memory usage** ○ Ideal for: basic utilities, lightweight social apps, offline-first tools
App Bundles	● Efficient distribution technology for Android apps, especially on **devices with varying screen sizes and capabilities** ● Instead of building separate APKs for each device configuration (screen size, CPU architecture, etc), you create a single .aab (Android App Bundle) file ● Google Play **dynamically delivers only the necessary code and resources for a specific user's device during installation**  ● Benefits: ○ **Small app file** ■ Users download only the parts of your app they need, resulting in significantly smaller download sizes ○ **Faster installation** ■ Faster installation times due to smaller downloads ○ **Good performance** ■ Optimized for specific devices, leading to better performance and potentially reduced battery consumption ○ **Easy to manage** ■ Easier to manage app releases and updates ● Best for: Efficient, scalable distribution across **diverse Android devices**

	• Suitable circumstances:
	  ◦ You want to **optimize app size** and **performance** for different screen sizes, architectures or locales
	  ◦ You are distributing via **Google Play** and want **automatic device-specific delivery**
	  ◦ You need **easy release management** and **smaller updates**
	  ◦ Ideal for: mainstream apps with broad device support - games, productivity tools, media apps

Google Play Instant (obsolete soon)	• **Allows users to try a portion of an app or game before downloading the full version**
	• Users can access instant apps through links in search results, emails, ads or social media
	• App has **limited functionality**, typically **focuses on core app functionality or a specific game level**
	• Benefits:
	  ◦ **Try before you buy**
	  ■ Users can experience the app before committing to a full download
	  ◦ **Reduced storage space**
	  ■ No need to install the full app to try it out
	  ◦ **Faster access**
	  ■ Instant apps load quickly, allowing users to start using them immediately
	• Limitations:
	  ◦ **Size limitations**
	  ■ Instant apps have size restrictions to ensure fast loading times
	  ◦ **Functionality limitations**
	  ■ Only a subset of the app's functionality may be available in the instant version
	• Best for: **App previews** or **lightweight experiences** without full installation
	• Suitable circumstances:
	  ◦ You want users to **try your app or game instantly** before downloading
	  ◦ You are promoting via **ads, search, social media or email**
	  ◦ You are offering a **core feature or demo level** with limited functionality
	  ◦ Ideal for: marketing campaigns, onboarding flows, game trials, lead generation

Support Chrome OS	• An operating system developed by Google that is designed to work primarily with web applications
	• **Uses the Chrome web browser as its main user interface**
	• **Supports Google Play Store and Android apps**
	• Best for: Apps that run on **Chromebooks** or **web-first environments**
	• Suitable circumstances:
	  ◦ Your app is compatible with **larger screens**, **keyboard / mouse input** or **web-based workflows**
	  ◦ You want to reach **education**, **enterprise** or **productivity users** on Chrome OS

	○ You are building **cross-platform apps** that work on both Android and desktop-like environments ○ Ideal for: note-taking apps, IDEs, design tools, educational platforms

Monetize Your App

Costs Involved when Developing Mobile App

Cost	Detail
Development costs	● What it covers: ○ Hiring developers, designers and testers ○ Building core features, UI/UX and backend systems ● Why it matters: ○ This is the largest upfront investment, especially for custom or complex apps ○ Costs vary based on app complexity, platform (iOS / Android) and team size
Tools and technologies	● What it covers: ○ SDKs, APIs, libraries and frameworks ○ Cloud services, analytics platforms and third-party integrations ● Why it matters: ○ Some tools are free but others (e.g. Firebase, Stripe, Mapbox) have usage-based pricing ○ Choosing the right stack affects scalability, performance and long-term cost
Marketing and promotion	● What it covers: ○ App store optimization (ASO), paid ads, influencer outreach 影响者推广 ○ Social media campaigns, email marketing, landing pages ● Why it matters: ○ Even the best app needs visibility - marketing drives download and user acquisition 即使是最好的应用程序也需要知名度--营销可推动下载和用户获取 ○ Budgeting for promotion is essential to avoid "launch and vanish" scenario 为避免出现 "推出后就消失 "的情况，编制推广预算至关重要
Ongoing costs - updates, server hosting, customer support, etc	● What it covers: ○ App updates, bug fixes, feature enhancements ○ Server hosting, database management, CDN ○ Customer support and user feedback channels ● Why it matters: ○ Apps are living products - maintaining performance and user satisfaction requires continuous investment ○ These costs scale with user base and app complexity
Legal and regulatory costs	● What it covers:

	○ Privacy policies, terms of service, GDPR / PDPA compliance ○ Licensing, intellectual property protection ○ App store registration fees and business incorporation ● Why it matters: ○ Ensures your app is legally safe and compliant, especially if handling user data or payments ○ Neglecting 忽视 this can lead to fines, bans or reputational damage

Factors that Influence the Monetization Model

Factor	Detail
Who are the users?	● **Age, income level, tech savviness, geographic location** ● Why It Matters: ○ Younger users may prefer freemium models with optional upgrades ○ High-income users may tolerate premium pricing or subscriptions ○ Tech-savvy users expect value and transparency - they'll avoid intrusive ads or gimmicks 噱头 ○ Understanding your audience helps tailor pricing, ad formats, and payment flows
How valuable is your app?	● **Does it solve a real problem or offer unique value?** ● Why It Matters: ○ Apps that solve critical problems (e.g., health, finance, productivity) can justify subscriptions or upfront fees ○ Entertainment or novelty 新奇 apps may rely more on ads or in-app purchases ○ The more indispensable your app, the more users are willing to pay 应用程序越不可或缺，用户就越愿意付费
What are your competitors doing?	● **Pricing models, feature sets, user reviews** ● Why It Matters: ○ If competitors offer similar features for free, charging upfront may hurt adoption 如果竞争对手免费提供类似功能，预收费用可能会影响采用率 ○ Studying competitors helps you position your app - either as a better free alternative or a premium upgrade ○ Benchmarking helps avoid pricing missteps and identify gaps in the market 制定基准有助于避免定价失误并找出市场差距
How much does it cost to make and maintain the app?	● **Development, hosting, updates, support** ● Why It Matters: ○ High development and maintenance costs may require recurring revenue (e.g. subscriptions) ○ Low-cost apps can afford freemium or ad-supported models

	○ Your monetization must cover costs and leave room for growth
How will it impact user experience?	● **Intrusiveness 侵扰性 of ads, friction in payment, access to features** ● Why It Matters: ○ Too many ads or paywalls can drive users away ○ Seamless monetization (e.g. optional upgrades, non-intrusive ads) improves retention ○ Balancing revenue with usability is key to long-term success

Monetization Model

Monetization Model	Detail
Premium Apps	● Paid ● In-app billing ● Extensive features ● Narrow niche in the market
	Extra info from Copilot: ● Model: One-time payment to download ● Mechanism: In-app billing for purchase ● Key Traits: ○ Offers **full access upfront** — no ads or locked features ○ Targets **niche markets 瞄准利基市场** with high-value features (e.g. pro tools, educational apps) ○ Requires strong **value proposition** to justify the price 需要强有力的价值主张来证明价格合理 ● Examples: ○ Productivity tools (e.g., Tasker, Nova Launcher Prime) ○ Specialized calculators, offline dictionaries, or niche utilities
Freemium Apps	● Free ● Uses in-app billing ● Digital goods ○ Durable ○ Consumable
	Extra info from Copilot: ● Model: Free to download, with optional purchases ● Mechanism: In-app billing for digital goods ● Key Traits: ○ Core features are free; **premium content or tools** are paid ○ Digital goods can be: ■ **Durable**: permanent (e.g. unlocking a new feature or skin) ■ **Consumable**: temporary (e.g. coins, lives, boosts) ● Examples: ○ Games (e.g., Clash of Clans, Candy Crush) ○ Editing apps (e.g., VSCO, Canva)

Subscription	• Free trial • In-app billing • Subscription fee

	Extra info from Copilot:
	• Model: Recurring payment (monthly/yearly)
	• Mechanism: In-app billing with optional free trial
	• Key Traits:
	  ○ Offers **ongoing value** (e.g., content updates, cloud sync, premium support)
	  ○ Ideal for apps with **dynamic content** or **service-based features**
	• Examples:
	  ○ Streaming apps (e.g., Netflix, Spotify)
	  ○ Productivity suites (e.g., Notion, Evernote Premium)

Ads	• Free • AdMob + Google Mobile Ads SDK • Show income-generating ads

	Extra info from Copilot:
	• Model: Free app supported by advertisements
	• Mechanism: AdMob or similar SDKs
	• Key Traits:
	  ○ Generates revenue from **impressions, clicks, or video views**
	  ○ Can include **banner ads, interstitials 插播广告, rewarded videos**
	  ○ Must balance ad placement with user experience
	• Examples:
	  ○ News apps, quiz games, utility apps
	  ○ Free versions of popular tools

E-Commerce	• Free • B2C • Technology, logistics and payment solutions • Sales commissions + Setup fees

	Extra info from Copilot:
	• Model: Selling physical or digital products
	• Mechanism: B2C platform with logistics and payment integration
	• Key Traits:
	  ○ App acts as a **storefront 店面** or **marketplace**
	  ○ Revenue from **sales commissions, setup fees, or product margins 来自销售佣金、安装费或产品利润的收入**
	• Examples:
	  ○ Shopee, Lazada, Amazon
	  ○ Brand-specific apps (e.g., Nike, Sephora)

Rewarded Products	• Free • Share content on social media, perform simple tasks (e.g. scan QR code, answer survey and etc) • Get free stuff, earn rewards

	Extra info from Copilot: ● Model: Users earn rewards by completing tasks ● Mechanism: Social sharing, surveys, QR scans, etc. ● Key Traits: ○ Drives **engagement and virality** 推动参与和病毒式传播 ○ Rewards can be **discounts, points, or free items** ○ Often used in **campaigns or loyalty programs** ● Examples: ○ BuzzBreak, GrabRewards, referral programs ○ Survey apps like Google Opinion Rewards
Service	● Free ● An extension of physical / online services
	Extra info from Copilot: ● Model: App extends a physical or online service ● Mechanism: **Free access**, **monetized through service delivery** ● Example: myTNB app, MAE by Maybank2u app ○ myTNB: bill payments, outage reports ○ MAE by Maybank2u: mobile banking, transfers, investments ● Use Case: Utilities, finance, healthcare, logistic
Data Collection	● Free ● Partnership / Affiliate 联营公司 ● Provide customer service, promotion, run a giveaway 举办赠品活动 and etc
	Extra info from Copilot: ● Model: Free app that monetizes user data via partnerships or ads ● Mechanism: Affiliate marketing 联属营销, behavioral analytics ● Example: Facebook, Chrome, X (Twitter) ○ Collect user behavior, interests, and engagement patterns ○ Monetize via targeted ads, sponsored content, or data partnerships ● Use Case: Social media, browsers, platforms with massive user bases