علامة التبويب 2

# **1) تقسيم طبقات نظيف (Clean Layers)**

**القاعدة:** افصل بين **Presentation** (UI), **Application** (State/Use-cases), **Domain** (Entities), **Data** (Repos & Sources).  
**الفائدة:** سهولة الاختبار، تقليل الترابط، تبديل المصادر أو التصميم بدون تكسير.  
**التطبيق السريع:**

/lib

/features

/catalog/ (UI + controllers + repo + models لهذه الميزة فقط)

/cart/

/checkout/

/core

/domain/ (entities, value\_objects)

/data/ (http\_client, local\_db, cache)

/app/ (router, theme, l10n, di, errors, result)

# **2) إدارة الحالة وحقن الاعتماديات (State & DI)**

**القاعدة:** استخدم **Riverpod** (أو Bloc—اختر واحدًا وثبّته) + DI عبر Providers.  
**الفائدة:** إعادة بناء Widgets بذكاء، اختبارات سهلة، وتجنّب Singleton العشوائي.  
**التطبيق:**

* ProviderScope في main.dart.
* StateNotifier لكل منطق ميزة (CartController, AuthController…).
* Provider لواجهات Repos، وProvider لعملاء الشبكة/التخزين.

# **3) نماذج غير قابلة للتغيير (Immutable Models)**

**القاعدة:** استخدم **freezed** + **json\_serializable** للكيانات ونتائج API.  
**الفائدة:** كود أقل + أمان أكبر + مساواة وقيم منسوخة (copyWith).  
**التطبيق:** @freezed للـEntities (Product, CartItem, User) + توليد fromJson/toJson.

# **4) طبقة المجال (Domain) أولًا**

**القاعدة:** كل ميزة لها **Use-cases** (دوال/كلاسات) تعمل على **Entities** وواجهة **Repository**.  
**الفائدة:** واجهة واضحة للـUI؛ تغيّر الـAPI لا يلمس الواجهة.  
**التطبيق:**

* GetProductList(), AddToCart(), PlaceOrder()…
* abstract class CatalogRepo { Future<Result<List<Product>>> getProducts(...); }

# **5) طبقة البيانات (Data) مع Result/Failure**

**القاعدة:** لا ترمي Exceptions للأعلى؛ أرجِع Result<Success, Failure>.  
**الفائدة:** تعامل موحّد مع الأخطاء في UI (شبكة/سيرفر/تحقق/ذاكرة).  
**التطبيق:**

* sealed class AppFailure { Network(); Server(msg); Cache(); Auth(); Validation(msg); }
* typedef Result<T> = Either<AppFailure, T>;

# **6) التوجيه (Routing) والوحدات**

**القاعدة:** استخدم **go\_router** مع حراس (Guards) مبسّطة، وسمِّ المسارات بوضوح.  
**الفائدة:** Deep-link داخلي بسيط، أكواد أقل، تنقّل يمكن اختباره.  
**التطبيق:**

* /, /catalog, /product/:id, /cart, /checkout, /orders.
* Guard خفيف للتحقق من تسجيل الدخول قبل /checkout.

# **7) تصميم نظام (Design System) موحّد**

**القاعدة:** Theme واحد + Tokens (ألوان/هوامش/زوايا/Typography) + Widgets مشتركة (Buttons, Cards…).  
**الفائدة:** شكل ثابت وسهل التغيير، تقليل التكرار.  
**التطبيق:**

* AppTheme: light/dark، أحجام خط متدرجة، مكوّنات جاهزة.
* Spacing.xs/s/m/l/xl, Radius.s/m/l, Elevation.s/m/l.

# **8) التعدد اللغوي وRTL**

**القاعدة:** فعّل **l10n** بملفات .arb، وادعم RTL افتراضيًا (العربية).  
**الفائدة:** نصوص منظّمة، سهولة ترجمة، تجربة عربية سليمة.  
**التطبيق:**

* flutter gen-l10n, مفاتيح مثل product\_title, add\_to\_cart.
* اختبر تكبير الخط وTextDirection.rtl.

# **9) التخزين المحلي وCache**

**القاعدة:** افصل **Cache** عن **State**؛ استخدم **Hive/Drift** أو SharedPreferences للخفيف.  
**الفائدة:** فتح سريع دون شبكة + عمل أوفلاين أساسي.  
**التطبيق:**

* Cache للـcatalog لمدة N دقائق، وسلة محليّة دائمة.
* سياسة: Stale-While-Revalidate (اعرض القديم ثم حدّث بصمت).

# **10) الشبكة والمرونة**

**القاعدة:** عميل واحد (مثلاً **Dio**) مع Interceptors، وإعادة محاولة **Exponential Backoff**.  
**الفائدة:** قِوة ضد مشاكل الشبكة وشفافية سجلات.  
**التطبيق:**

* Timeout موحّد، توحيد رؤوس الطلبات (Headers)، فحص اتصال connectivity\_plus.
* صف انتظار للطلبات الحرِجة (Queue) عند فقد الشبكة ثم إرسال لاحقًا (idempotency key).

# **11) معالجة الأخطاء وتجربة المستخدم**

**القاعدة:** شاشة أخطاء ودية + Snackbars مخصّصة لكل خطأ Domain.  
**الفائدة:** فهم المستخدم للمشكلة دون ضياع.  
**التطبيق:**

* Global FlutterError.onError + runZonedGuarded مع تسجيل.
* خرائط: Network → “تأكد من الشبكة”, Auth → “سجّل دخولك”, Validation → رسائل حقول.

# **12) الأداء**

**القاعدة:** استخدم const حيثما أمكن، قسّم Widgets، تجنّب العمل الثقيل على الـUI thread.  
**الفائدة:** سلاسة (FPS ثابت) واستهلاك أقل.  
**التطبيق:**

* قوائم كبيرة: ListView.builder/Slivers, AutomaticKeepAlive للمهمات.
* صور: cached\_network\_image + أحجام مناسبة.
* Debounce للبحث، Throttle للنقرات.

# **13) التهيئة والتكوين (Config)**

**القاعدة:** flavors + dart-define، وملف AppConfig يحقن في ProviderScope.  
**الفائدة:** تبديل بيئات ومفاتيح بسهولة بدون لمس الكود.  
**التطبيق:**

* --dart-define=ENV=prod → AppConfig(baseUrl, featureFlags, logLevel).

# **14) الأمان**

**القاعدة:** خزّن الرموز في **flutter\_secure\_storage**، لا تسجّل بيانات حسّاسة، وفكّر في **TLS pinning**.  
**الفائدة:** حماية جلسات المستخدم وتقليل المخاطر.  
**التطبيق:**

* Token في Keystore/Keychain، تنظيفه عند تسجيل الخروج.
* تنقيح السجلات (لا تطبع Headers/Body الحسّاسة).

# **15) الاختبارات (Test Pyramid)**

**القاعدة:** هرَم اختبارات: **Unit** (Use-cases, Repos), **Widget** (Screens), **Integration** (رحلة شراء).  
**الفائدة:** ثقة بالتغييرات + بُقع أعطال أقل.  
**التطبيق:**

* Mocktail لواجهات Repos، Golden tests للـUI الأساسية، سيناريو Checkout كامل.

# **16) السجلّات والمراقبة (محليًا)**

**القاعدة:** Logger مركزي بمستويات (debug/info/warn/error) + مفاتيح إطفاء.  
**الفائدة:** تتبّع أعطالك محليًا بسرعة أثناء التطوير.  
**التطبيق:**

* logger مع Prefix للميزة، طباعة مختصرة في الإنتاج (أو تعطيل).

# **17) قواعد الترميز والمراجعة**

**القاعدة:** lints صارمة (flutter\_lints/very\_good\_analysis) + dart format + PR template.  
**الفائدة:** اتساق الكود وسهولة المراجعة.  
**التطبيق:**

* CI بسيط: analyze, test, format --set-exit-if-changed.

## **مثال هيكل ميزة (Feature) — “Catalog”**

/features/catalog

/presentation

/pages/ catalog\_page.dart

/widgets/ product\_tile.dart

catalog\_controller.dart (StateNotifier)

/domain

product.dart (freezed)

catalog\_repo.dart (abstract)

usecases/

get\_product\_list.dart

/data

catalog\_repo\_impl.dart

catalog\_remote\_ds.dart

catalog\_local\_ds.dart

## **نموذج Result/Failure مختصر**

sealed class AppFailure {

const AppFailure();

factory AppFailure.network() = NetworkFailure;

factory AppFailure.server(String msg) = ServerFailure;

factory AppFailure.cache() = CacheFailure;

factory AppFailure.auth() = AuthFailure;

factory AppFailure.validation(String msg) = ValidationFailure;

}

class Result<T> {

final T? ok;

final AppFailure? err;

const Result.ok(this.ok) : err = null;

const Result.err(this.err) : ok = null;

bool get isOk => err == null;

}

## **قائمة “جاهزية الإطلاق” (داخل التطبيق فقط)**

* الطبقات منفصلة (Presentation/Application/Domain/Data).
* Riverpod/Bloc موحّد في كل المزايا + DI عبر Providers.
* نماذج freezed + json\_serializable مفعّلة.
* Routing بـ go\_router + حراس الدخول.
* Cache محلي + سياسة Stale-While-Revalidate.
* Result/Failure موحّد، بدون Exceptions طافية للـUI.
* شاشة أخطاء وSnackbars موحّدة الرسائل.
* أداء: const، قوائم مبنية، صور مُخزّنة، بحث مُخفَّض (debounce).
* Config عبر flavors وdart-define.
* أمان: secure storage، سجلات منقّحة.
* اختبارات: Unit/Widget/Integration لرحلة Checkout.
* Lints + CI تحليل/اختبارات.

علامة التبويب 3

# **دليل العمل المتكامل لتطبيق Flutter**

(Architecture + Testing + CI/CD + إصدار)

## **1) نموذج الفروع (Git) والالتزام**

* **Trunk-based بسيط**
  + main محمي. كل تطوير عبر **feature branches**: feature/<slug>، إصلاحات حرجة hotfix/<slug>.
  + **Release tag** لكل إصدار: vX.Y.Z (semver) مع **build number** لكل منصة.
* **Conventional Commits**
  + feat: ..., fix: ..., refactor: ..., test: ..., chore: ..., docs: ...

**PR Template (مختصر)**الملخص: …

قبل/بعد (صور إن لزم): …

تحقق:

[ ] اتّبعتُ الهيكل (presentation/application/domain/data)

[ ] لا منطق أعمال داخل Widgets

[ ] تغطية اختبار ≥ 70% للوحدة المتأثرة

[ ] RTL/حجوم خط كبيرة تعمل

[ ] flutter analyze & format نظيفة

## **2) البيئات والتكوين (Flavors)**

* **Flavors:** dev / staging / prod مع --dart-define=ENV=prod.
* **AppConfig** يُحقن عبر DI (baseUrl, featureFlags, logLevel).
* **نسخ بيانات الاختبار** داخل dev فقط.

## **3) هرم الاختبارات (Testing Pyramid)**

**الهدف:** نكسر أي كسر قبل أن يصل لواجهة المستخدم.

### **Unit Tests (A (الطبقة المنطقية/الدومين) — كثيرة ورخيصة**

* تغطي **use-cases, repositories (mock)**، المحوّلات **DTO↔Entity**.
* أدوات: flutter\_test, mocktail, freezed, json\_serializable.

أمر التشغيل:  
flutter test --coverage

### **Widget Tests (B (شاشات/Widgets) — مستوى متوسط**

* تتحقق من العرض والسلوك (زر يضيف إلى السلة، رسائل خطأ).
* استخدم **golden tests** للّقطات البصرية الأساسية (رؤوس/أزرار).
* أدوات: golden\_toolkit.
* قواعد: لقطات منفصلة لـ RTL وحجم خط كبير.

### **Integration / E2E (C (رحلات رئيسية) — قليلة لكنها ذهبية**

* **سيناريو أساسي:** كتالوج → PDP → إضافة للسلة → Checkout (بدون شبكات خارجية؛ mock لطبقة البيانات).
* أدوات: integration\_test.

تشغيل على محاكي:  
# Android على CI مع emulator

flutter test integration\_test -d emulator-5554

### **D) Performance & Accessibility (خفيفة لكن ثابتة)**

* **Performance budgets:**
  + App start < **2.5s**،
  + بناء PDP < **200ms**،
  + Scroll سلس (لا jank ملحوظ).
* **إتاحة:** عناصر مهمة تحمل Semantics وعناوين، دعم RTL وحجوم خط كبيرة.

### **E) حدود الجودة (Quality Gates)**

* تغطية إجمالية ≥ **70%** (Unit+Widget).
* flutter analyze صفر أخطاء، format إجباري.
* اختبارات Integration **للرحلات الحرجة** تمرّ على PRs الكبيرة وعلى كل Release.

## **4) هيكلة المجلدات للاختبارات**

/test

/unit/ // use-cases, repos (mocks)

/widget/ // شاشات وWidgets

/goldens/ // baseline snapshots

/integration\_test/ // رحلات أساسية

## **5) أدوات الـdev\_dependencies (بدون أرقام إصدارات لتفادي التعارض)**

dev\_dependencies:

flutter\_test:

sdk: flutter

integration\_test:

sdk: flutter

mocktail:

freezed:

build\_runner:

json\_serializable:

golden\_toolkit:

very\_good\_analysis:

## **6) CI على GitHub Actions (قابل للتكييف)**

يشغّل التحليلات، التنسيق، الاختبارات (Unit/Widget)، ثم Integration على محاكي Android، ويرفع المخرجات.

name: CI

on:

pull\_request:

branches: [main]

push:

branches: [main]

workflow\_dispatch:

jobs:

analyze-test:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v4

- uses: subosito/flutter-action@v2

with:

flutter-version: 'stable'

- run: flutter pub get

- run: flutter format --set-exit-if-changed .

- run: flutter analyze

- run: flutter test --coverage

- uses: actions/upload-artifact@v4

with:

name: coverage

path: coverage/lcov.info

integration-android:

runs-on: ubuntu-latest

needs: analyze-test

steps:

- uses: actions/checkout@v4

- uses: subosito/flutter-action@v2

with:

flutter-version: 'stable'

- uses: reactivecircus/android-emulator-runner@v2

with:

api-level: 31

arch: x86\_64

script: |

flutter pub get

flutter test integration\_test -d emulator-5554

يمكن إضافة Job على macos-latest لبناء iOS واختبارات Integration على محاكي iOS إن رغبت.

## **7) بوابات الدمج (Merge Gates) وحماية الفرع**

* تفعيل **Branch protection** على main:
  + مطلوب **نجاح CI** (الوظيفتين أعلاه).
  + **مراجعتان** على الأقل، و**CODEOWNERS** للمسارات الحساسة.
  + منع Push مباشر؛ دمج عبر PR فقط.

## **8) الإصدار (Release) والنسخ**

* **SemVer:** MAJOR.MINOR.PATCH، و**Build number** يزيد كل نشر.
* **Android:**
  + versionName من pubspec.yaml، versionCode = build number.
  + إصدار Release Debug-gable **ممنوع**.
* **iOS:**
  + CFBundleShortVersionString = semver، CFBundleVersion = build.
* **وسم الإصدار** على Git: vX.Y.Z + Release notes (ما تغيّر/تصحيحات/تحذيرات).

## **9) التحقق اليدوي قبل الإصدار (Manual QA Smoke)**

* **RTL + تكبير خط** على 2–3 شاشات أساسية.
* **رحلة Checkout** (بدون خدمات خارجية) تعمل على iOS/Android.
* **الأداء**: فتح التطبيق أول مرة، فتح PDP، تمرير في قائمة طويلة.
* **الأخطاء**: شبكات مقطوعة ثم عودة (تجربة Offline/Retry لطيفة).

## **10) مراقبة وتشغيل (بعد الإصدار)**

* **سجلات محلية** مفعّلة بمستوى INFO في الإنتاج، DEBUG في dev فقط.
* **بوّابات صحّة داخل التطبيق** (feature flags) لإيقاف شاشة/تجربة إن لزم.
* **تنبيهات داخلية** (اختياري): عدّادات أعطال/أداء تُعرض في صفحة Debug داخل التطبيق (غير مرئية للمستخدمين).

## **11) معايير الكود والمراجعة**

* **Lints:** very\_good\_analysis أو flutter\_lints مع قواعد صارمة.
* **لا منطق أعمال داخل Widgets** — يجب أن يعيش في StateNotifier/Use-case.
* **التسمية:**
  + ملفات: snake\_case.dart
  + أصناف: PascalCase
  + متغيرات/دوال: camelCase
  + Widgets: …Page, …View, …Tile

## **12) جاهزية المهمة واكتمالها (DoR / DoD)**

**Definition of Ready**

* سيناريوهات قبول وحدود واضحة
* واجهات APIs/نماذج محدّثة
* تأثير الأداء متوقّع (budgets)
* لقطات تصميم/نصوص l10n جاهزة

**Definition of Done**

* اتّباع الهيكل (feature-first + clean layers)
* اختبارات Unit/Widget مضافة، Integration لرحلات حرجة
* تغطية ≥ 70% للوحدة المتأثرة
* analyze/format نظيفان، CI أخضر
* README صغير داخل الميزة يشرح القرارات

## **13) قوالب سريعة (جاهزة للنسخ)**

### **A) أمر تشغيل Goldens وتحديثها**

# تشغيل

flutter test --update-goldens=false

# تحديث baselines عند قبول تغيير تصميم مقصود

flutter test --update-goldens

### **B) مثال Integration Test (مختصر)**

import 'package:integration\_test/integration\_test.dart';

import 'package:flutter\_test/flutter\_test.dart';

import 'package:app/main.dart' as app;

void main() {

IntegrationTestWidgetsFlutterBinding.ensureInitialized();

testWidgets('journey: catalog → pdp → add to cart → checkout', (tester) async {

app.main();

await tester.pumpAndSettle();

// افتح PDP

await tester.tap(find.byKey(const Key('catalog.item.0')));

await tester.pumpAndSettle();

// أضف للسلة

await tester.tap(find.byKey(const Key('pdp.add\_to\_cart')));

await tester.pumpAndSettle();

// إلى السلة ثم الدفع

await tester.tap(find.byKey(const Key('cart.go\_checkout')));

await tester.pumpAndSettle();

// تحقّق عنصر موجود

expect(find.byKey(const Key('checkout.item.0')), findsOneWidget);

});

}

### **C) أمر CI للتحقق الكامل محليًا**

flutter clean && flutter pub get

flutter format --set-exit-if-changed .

flutter analyze

flutter test --coverage

flutter test integration\_test -d <DEVICE\_ID>

* التزموا بـ **Architecture نظيفة + هرم اختبارات + CI بسيط** و ستحصلون على بنية مرتبة، تغييرات آمنة، وإصدارات بلا مفاجآت.

علامة التبويب 4

# **1) الهيكلة (Architecture)**

* نمط معماري واحد ومعلن (مثلاً **Feature-first + Clean Architecture** أو **BLoC/Riverpod + Repository**)، ومكتوب في ملف ARCHITECTURE.md.
* فصل صارم بين الطبقات: Presentation / Domain / Data. ما في استدعاءات شبكة من الـWidgets.
* اعتماد **Repository** واجهات (abstract) وتطبيقات (impl)، وسهولة استبدال المصادر (API، كاش محلي) بالـDI.

# **2) هيكلة المشروع والأسماء**

* تنظيم بالمزايا (feature-first):  
  features/checkout, features/catalog, features/auth … وكل ميزة فيها presentation/, domain/, data/.
* أسماء ملفات/كلاسات موحدة (snake\_case للملفات، PascalCase للكلاسات)، ملف واحد لكل كلاس.
* لا توجد مجلدات “misc/utils” عشوائية—كل أداة تابعة لميزة أو لمجلد core مشروح.

# **3) حالة التطبيق (State Management)**

* اختيار إطار واحد واضح (مثلاً **Riverpod** أو **BLoC**) مع دليل استخدام داخلي.
* الحالة مشتقة/قابلة للاختبار، بدون تمرير context غير ضروري.
* منع إعادة البناء الزائدة (selectors, providers granular, const widgets، مفاتيح Keys صحيحة).

# **4) التصفّح (Navigation)**

* راوتر حديث (مثل GoRouter) مع تعريف مسارات مركزي وحماية (guards) لتوثيق/صلاحيات.
* Deep Links & Dynamic Links تعمل ومختبرة على iOS/Android.
* مخطط تنقل موثق (flow diagrams لصفحات حرجة مثل الدفع).

# **5) طبقة البيانات والشبكة**

* دوال شبكة عبر عميل مركزي (Dio/Retrofit مثلاً) مع **timeouts**, **retry**, **circuit breaker** بسيط.
* نماذج بيانات مولّدة (json\_serializable/freezed) بدون parsing يدوي.
* كاش محلي منظّم (Hive/Isar/Drift) بسياسة صلاحية (TTL) واضحة.

# **6) التصميم والـUI**

* **Design System** موحد: Theme, ColorScheme, Typography, Spacings, Components مخصصة.
* عناصر UI قابلة لإعادة الاستخدام (Buttons, Inputs, Cards) ضمن shared/ui أو core/ui.
* **RTL** مفعل ومختبَر (العربية)، أحجام خطوط تستجيب لتكبير النص.

# **7) الأداء (Performance)**

* أول إطارات الرسم **< 2 ثوانٍ** على جهاز متوسط، ومعدل jank < **1%** في شاشات طويلة التمرير.
* صور مضغوطة ومسبقة التحميل عند الحاجة، وconst قدر الإمكان، وتجنّب العمليات الثقيلة على الـUI thread.
* استخدام flutter build appbundle --analyze-size دوريًا ومتابعة النمو.

# **8) الموثوقية والأخطاء**

* طبقة أخطاء موحدة (AppException) مع خرائط رسائل ودلالات أكواد HTTP.
* تسجيل Logs منظّم (logger/Sentry) مع معرّف للجلسة والمستخدم (بدون بيانات حساسة).
* شاشة/حالة fallback لكل خطأ (شبكة ضعيفة، انتهاء جلسة، 404)، و**Retry UX** واضح.

# **9) الأمان والخصوصية**

* أسرار وKeys خارج الكود (dart-define، أو إدارة أسرار CI). لا أسرار في Git.
* تشفير مخزن محلي للحساس (tokens) واستخدام Keystore/Keychain.
* سياسة تتبع وخصوصية واضحة، وإيقاف جمع Analytics عند طلب المستخدم.

# **10) التحليلات والقياس (مهم لتطبيقك)**

* قاموس أحداث موحد analytics\_events.md (أسماء ثابتة، snake\_case، مع معايير: من أين أتى المستخدم؟ ماذا ضغط؟).
* تتبع مسار الشراء كاملاً: عرض منتج ⇒ إضافة للسلة ⇒ بدء الدفع ⇒ اختيار طريقة الدفع ⇒ تأكيد الطلب.
* تعريف أبعاد ثابتة (user\_id, cohort, app\_version, channel) ومراجعة جودة البيانات شهريًا.

# **11) الجودة والاختبارات**

* Linters مفعلة (flutter\_lints أو very\_good\_analysis) و**صفر** تحذير في flutter analyze.
* تغطية اختبارات **≥ 70%** لوحدات المنطق، مع **Widget tests** للشاشات الحرجة، و**Integration tests** لتدفق الشراء.
* **Golden tests** للمكونات البصرية الأساسية لضمان ثبات التصميم.

# **12) الإصدارات وCI/CD**

* Flavors: dev/stage/prod، مع ملفات Firebase منفصلة لكل بيئة.
* **CI** يشغّل: format + analyze + tests + build لكل PR؛ **CD** يبني النسخ التجريبية تلقائيًا.
* ترقيم إصدارات واضح (Semantic Versioning) وCHANGELOG.md مختصر لكل إصدار.

# **13) الإشعارات والروابط والميزات الخاصة بالتجارة**

* Push notifications عبر خدمة موحدة (FCM) مع قنوات Android وقابلة للنقر إلى الشاشات المناسبة (deep link).
* معالجة الدفع/الدفع عند الاستلام UX نظيف، وسياسة فشل محاولة الدفع واضحة.
* الاسترجاع/الإرجاع/التوصيل مدمج UX مع تتبع حالة الطلب داخل التطبيق.

## **فحص سريع (90 دقيقة) لتقييم الوضع الحالي**

نفّذ هذه الخطوات وسجّل النتائج كنقاط “نجاح/فشل”:

1. **كود وانضباط**

* شغّل:  
  dart format --set-exit-if-changed . (لا تغييرات = ✔️)  
  flutter analyze (0 تحذيرات/أخطاء = ✔️)
* افتح pubspec.yaml: لا توجد مكتبات مكررة/قديمة بشكل خطير، وكل الإصدارات محددة.

1. **الاختبارات**

* flutter test --coverage (تحقّق ≥ 70% = ✔️، وإلا حدّد الملفّات الأضعف وضع خطة رفع).

1. **الأداء**

* شغّل في **profile mode** وجرب أهم 3 شاشات (الرئيسية/القسم/الدفع).  
  راقب jank في DevTools، وحجم build عبر --analyze-size.

1. **البناء والتوزيع**

* جرّب بناء flavors المختلفة بنجاح (dev/stage/prod).  
  تأكد أن مفاتيح Firebase/SDKs الصحيحة تُسحب حسب البيئة.

1. **التحليلات**

* افتح Firebase/GA4: تأكّد أن الأحداث الأساسية تصل وبالأبعاد المتفق عليها، ولا توجد أسماء عشوائية.

1. **الأمان**

* افحص المستودع بحثًا عن أسرار (google-services.json/plist حساسة؟ مفاتيح مكشوفة؟).  
  تأكد من تخزين tokens مشفّر محليًا.

1. **الـUX وRTL**

* اختبر **تكبير الخط** و**العربية RTL** في 3–4 شاشات، وتأكد من خلوّ الانعكاسات/القص.

## **قائمة مراجعة لكل Pull Request (انسخها للفريق)**

* شرح التغيير ولقطات شاشة/فيديو قبل/بعد (لو UI).
* تغطية اختبارات مضافة أو محدّثة.
* لا كسر للـAPI الداخلي—أو موثق في CHANGELOG.
* flutter analyze وflutter test ناجحة.
* لا استيراد متقاطع بين الميزات (احترام الطبقات).
* أحداث Analytics مضافة/محدّثة حسب القاموس.
* فحص أداء سريع (عدم وجود setState كثيف/بُنَى ثقيلة في build).
* لقطات RTL والتحجيم (إن لزم).

## **مؤشرات “خطر” لو ظهرت عندك أصلّح فورًا**

* أكواد شبكة داخل Widgets أو Cubits/Notifiers تتصل مباشرة بدون Repository.
* Utils عملاقة “تفعل كل شيء”.
* تعدد أنماط إدارة الحالة في نفس المشروع بدون سبب.
* غياب الاختبارات تمامًا.
* أسرار داخل الكود أو في Git.
* Theme غير موحد ومكونات متكررة عن غير حاجة.

تعليمات لضبط البنية والاختبارات (Flutter)

# **تعليمات لضبط البنية والاختبارات (Flutter)**

1. المعمارية وهيكلة الملفات

* اعتماد نمط واحد: feature-first مع طبقات presentation/domain/data ونمط Repository.
* توثيق مختصر في ملف ARCHITECTURE.md يذكر: إدارة الحالة المختارة، آلية التنقل، طبقة البيانات، معالجة الأخطاء.
* تنظيم المجلدات لكل ميزة: features/<feature>/{presentation,domain,data} وبنية core/{ui,network,errors,routing}.
* منع استدعاءات الشبكة أو I/O داخل Widgets.
* استخدام أسماء موحّدة: snake\_case للملفات، PascalCase للكلاسات.

1. إدارة الحالة والتنقل

* اختيار إطار واحد لإدارة الحالة (Riverpod أو BLoC) وتطبيقه في جميع الميزات.
* استخدام GoRouter بمسارات مركزيّة وحمايات للمسارات التي تتطلب تسجيل دخول.
* تفعيل واختبار Deep Links/Dynamic Links لشاشات المنتج والعروض.

1. طبقة البيانات والشبكة

* عميل شبكة مركزي (Dio أو Retrofit) مع مهلات زمنية وسياسة إعادة محاولة محدودة.
* نماذج بيانات مولّدة (json\_serializable أو freezed)، بدون parsing يدوي.
* كاش محلي منظم (Hive/Isar/Drift) مع سياسة صلاحية زمنية واضحة.
* طبقة أخطاء موحّدة AppException بخرائط رسائل صديقة للمستخدم.

1. التصميم وملاءمة العربية

* Design System موحّد (Theme/ColorScheme/Typography/Spacing) ومكوّنات UI مشتركة قابلة لإعادة الاستخدام.
* دعم RTL واختبار تكبير الخط 1.3× و1.6× دون قص أو انعكاسات.
* مكونات رئيسية مثل بطاقة المنتج وأزرار الشراء تُبنى كمكوّنات مشتركة.

1. الأمان والخصوصية

* عدم تخزين الأسرار داخل الكود أو Git؛ تمريرها عبر dart-define/أسرار CI.
* تخزين محلي مشفّر للرموز الحساسة باستخدام Keystore/Keychain.
* احترام طلب المستخدم بإيقاف التتبع، وعدم إرسال بيانات شخصية حساسة للتحليلات.

1. التحليلات والقياس

* إنشاء قاموس أحداث analytics\_events.md (أسماء ومتى تُطلق والمعلمات).
* تتبّع كامل لمسار الشراء: view\_item → add\_to\_cart → begin\_checkout → add\_payment\_info (COD) → purchase.
* إرسال أبعاد ثابتة لكل حدث: user\_id، app\_version، channel، currency، value.

1. الاختبارات والجودة

* تفعيل linters وعدم قبول تحذيرات في flutter analyze.
* تغطية اختبارات للمنطق الأساسي لا تقل عن 70%، مع Widget tests للشاشات الحرجة، وGolden tests للمكونات الأساسية.
* اختبار تكاملي لتدفق “إضافة للسلة → تأكيد الطلب (الدفع عند الاستلام)”.

1. الأداء والموثوقية

* أول إطار رسم أقل من ثانيتين على جهاز متوسط، وjank أقل من 1% في القوائم الطويلة.
* فحص حجم البناء دورياً باستخدام --analyze-size وعدم السماح بزيادة غير مبررة.
* شاشات خطأ/انقطاع الشبكة مع زر إعادة المحاولة وتجربة استخدام واضحة.

1. CI/CD وقواعد الدمج

* منع الدمج إلى main إلا عبر Pull Requests مع نجاح: format، analyze، tests، build.
* بناء dev/stage/prod بنجاح على iOS/Android.
* قالب PR يطلب لقطات قبل/بعد، تغطية للاختبارات، وأثر على التحليلات والحجم.

1. فحص القبول قبل أي دمج

* نجاح flutter analyze بلا تحذيرات.
* نجاح flutter test مع تغطية ≥ 70% للملفات المتأثرة.
* نجاح Integration test لتدفق الشراء (COD).
* سلوك Deep Link يعمل إلى صفحة منتج/عرض.
* تقرير حجم البناء تمت مراجعته، ولا زيادة غير مبررة.

# **خطوات التحقق العملية (تشغّلونها أنتم وتوثّقون النتيجة)**

* تشغيل: dart format --set-exit-if-changed .
* تشغيل: flutter analyze (يجب صفر أخطاء/تحذيرات).
* تشغيل: flutter test --coverage (توثيق النسبة).
* تشغيل في profile mode، قياس زمن أول إطار وjank عبر DevTools في الشاشات: الرئيسية، قائمة الفئة، السلة/الدفع.
* بناء Android App Bundle وiOS (no codesign) لبيئات dev وstage وprod.
* محاكاة شبكة ضعيفة وانقطاع تام للتأكد من رسائل الخطأ وزر إعادة المحاولة.
* فتح Firebase/GA4 DebugView والتحقق من وصول أحداث: view\_item، add\_to\_cart، begin\_checkout، add\_payment\_info (COD)، purchase بالقيم الصحيحة.
* اختبار RTL وتكبير الخط 1.3× و1.6× على الشاشات الحرجة مع مراجعة لقطات.
* اختبار Deep Link إلى صفحة منتج وإلى صفحة عروض.
* مراجعة أن لا توجد أسرار داخل المستودع وأن التخزين المحلي للرموز الحساسة مشفّر.

# **ملخص الاستفادة**

* توحيد البنية يقلل الديون التقنية ويجعل الإضافة والصيانة أسرع وأقل أخطاء.
* منع تسرّب الشبكة إلى الواجهة يعزّز الاختبار وإعادة الاستخدام.
* إدارة حالة موحّدة تقلل إعادة البناء وتحسّن الأداء.
* تصميم موحّد وRTL صحيح يرفع جودة التجربة خصوصاً للعربية.
* التحليلات المعيارية تمنح رؤية دقيقة لمسار الشراء وتسهّل قرارات التسويق.
* اختبارات كافية + CI تقللان الأعطال وتمنعان regressions.
* مراقبة الحجم والأداء تحافظ على سرعة التطبيق على الأجهزة المتوسطة والضعيفة.
* أمان وخصوصية صحيحان يحميان البيانات ويكسبان ثقة المستخدم.

لمعرفة مستوئ بنية التطبيق flutter

# **لمعرفة مستوئ بنية التطبيق flutter**

# **1) الهيكل المعماري (Architecture)**

* طبقات واضحة مفصولة: Presentation (UI) / Domain (Use-Cases, Entities) / Data (Repositories, Data-sources).
* إدارة حالة موحّدة عبر المشروع (Bloc/Cubit أو Riverpod) بدون خلط المنطق داخل الـWidgets.
* حقن تبعيات (DI) باستخدام get\_it أو Riverpod providers—ولا يوجد new داخل الطبقات العليا.
* كل ميزة (Feature) داخل Module/Package خاص بها: features/catalog/…, features/cart/…
* العقود أولاً: تعريف واجهات Repositories في الـDomain وتنفيذها في الـData.  
  **اختبار سريع:** ابحث عن كلمة http وSharedPreferences—يجب ألّا تظهر في UI/Bloc بل داخل Data-sources فقط.

# **2) بنية المشروع والملفات**

مسارات ثابتة:  
lib/

core/ (errors, utils, theming, config)

features/

auth/

presentation/

domain/

data/

catalog/

cart/

app.dart (root)

main\_[env].dart

* لا ملفات “عملاقة”: أي ملف > 300 سطر يحتاج تفكيك.
* أسماء بديهية، وتعليقات قصيرة للمقاطع المعقّدة فقط.  
  **اختبار سريع:** احسب عدد الطبقات داخل features/\*—يجب أن تراها الثلاث (presentation/domain/data) لكل ميزة.

# **3) الجودة الساكنة (Linting & Style)**

* مفعّل: flutter\_lints أو very\_good\_analysis + Formatter افتراضي.
* لا تحذيرات Lint في CI.
* كوميتات قصيرة بمعيارية (Conventional Commits).  
  **أوامر:**flutter analyze يجب أن يعود صفر أخطاء.  
  dart format --output=none --set-exit-if-changed . في CI.

# **4) الاختبارات والتغطية**

* Unit Tests لطبقة Domain (Use-Cases) و Repositories.
* Widget Tests للـUI الأساسية (القوائم/التفاصيل/السلة).
* Integration/E2E (مثل Patrol/Flutter Driver/Integration Test) لتدفق شراء كامل.
* حد أدنى للتغطية: **≥ 60%** الآن (ارفعه تدريجيًا إلى 80%).  
  **أوامر:**flutter test --coverage ثم راجع lcov.info.

# **5) الأداء والقياسات (Performance Budgets)**

* وقت تشغيل أولي (cold start): **< 2.5s** على أجهزة متوسطة.
* حجم APK/IPA: **< 30–40MB** (بدون موارد مفرطة).
* FPS مستقر في الشاشات الثقيلة (> 55fps).
* استهلاك ذاكرة منضبط وعدم تسريبات (راقب عبر DevTools).  
  **أوامر:** استخدم Flutter DevTools (CPU, Memory, Performance) + Profile mode.

# **6) الشبكة والـAPI**

* طبقة Network موحّدة (Dio/Http) مع Interceptors: Auth, Retry, Logging (غير فعّال في الإنتاج).
* معالجة أخطاء موحّدة (Failure types) ورسائل مستخدم ودّية.
* Timeouts محددة + Retry/backoff.
* الـModels مولّدة (json\_serializable) وليس “يدوي”.  
  **اختبار سريع:** افصل الإنترنت وتأكد من رسائل الأخطاء وسلوك Retry.

# **7) الاعتمادات والبيئات (Config & Secrets)**

* ملفات بيئة منفصلة: main\_dev.dart, main\_prod.dart + lib/core/config/app\_config.dart.
* لا أسرار داخل الكود/الـrepo (استخدم مفاتيح من CI/CD أو Firebase Remote Config).
* Feature Flags لتفعيل/إيقاف ميزات دون إطلاق نسخة جديدة.

# **8) الاعتمادية والمرونة (Offline & Caching)**

* سياسة Cache واضحة (Hive/Isar/Sqflite) للمنتجات والسلة.
* Offline-first لصفحات رئيسية (تعامل لطيف مع انقطاع الشبكة).
* مزامنة مؤجلة (Queued writes) إن لزم.

# **9) الرصد والأعطال (Observability)**

* Crashlytics مفعّل مع تتبّع الإصدارات (release channel, build number).
* سجلات (Logs) مضبوطة عبر wrapper مركزي (Info/Warning/Error) وتُغلق بالـrelease.
* مؤشرات صحة: Crash-free sessions **≥ 99.5%**، ANR منخفضة، أخطاء الشبكة أقل من **2%** من الطلبات.

# **10) الأمان والخصوصية**

* عدم تسجيل أو عرض أي PII في السجلات.
* TLS إلزامي، التحقق من الشهادات إن أمكن.
* حماية مفاتيح الطرف الثالث، ومراجعة أذونات التطبيق (لا تطلب ما لا تحتاجه).
* حماية الروابط العميقة/المعطيات القادمة من خارج التطبيق (validation).

# **11) واجهة المستخدم (UX/UI) و i18n/a11y**

* Theme موحّد (Light/Dark) وخامات تصميم ثابتة (Spacing, Radius, Typography).
* i18n مفعّل (ARB/Intl) دون نصوص صلبة داخل الأكواد.
* توافق إمكانية الوصول (TextScale, Semantics, Contrast).

# **12) التحليلات والمنتج (Analytics)**

* تعريف Event Map موحّد (اسم الحدث، المعاملات، متى يُطلق).
* عدم تكرار الأحداث، واستخدام User Properties مهمة (المدينة/القناة عند احترام الخصوصية).
* ربط Funnel للشراء: عرض منتج → إضافة للسلة → بدء الدفع → تأكيد الطلب.
* تحقق دوري من اتساق الأرقام بين Firebase/GA4 وبيانات الخادم.

# **13) الأتمتة والإطلاق (CI/CD)**

* بايبلاين:
  1. Format & Lint
  2. Tests + Coverage Gate
  3. Build (dev/stage/prod)
  4. Signing/Versioning آلي
  5. توزيع داخلي (Firebase App Distribution/TestFlight)
* Changelog تلقائي، وRelease Tags واضحة.
* فحوصات صحية بعد الإطلاق (Post-release checklist).

# **14) الوثائق والحَوْكمة**

* README هندسي يشرح التشغيل، البنية، وكيفية إضافة Feature.
* قرارات معمارية قصيرة (ADR) لكل قرار مهم.
* دليل مساهمة (CONTRIBUTING) + معايير Code Review.
* خريطة تبعيات خارجية وتبريرها.

## **خطة تدقيق عملية (3 أيام عمل)**

**اليوم 1:**

* فحص البنية والطبقات + هيكل الملفات + Lint/Analyze + Secrets/Config.  
  **المخرجات:** تقرير سريع مع 10 ملاحظات ذات أولوية.

**اليوم 2:**

* اختبارات (نِسَب، تغطية)، أداء (DevTools)، شبكة/أخطاء، Offline.  
  **المخرجات:** أرقام (start time، FPS، حجم build، معدل أخطاء API).

**اليوم 3:**

* Observability (Crashlytics)، Analytics map، CI/CD، UX/i18n/a11y.  
  **المخرجات:** قائمة تحسينات + حدود قياسية (budgets) + جدول تنفيذ.

## **معايير قبول مبسطة (OKRs تقنية)**

* **تحذيرات Lint:** صفر في الفرع الرئيسي.
* **تغطية:** ≥ 60% الآن (خطة للوصول 80%).
* **Start time:** ≤ 2.5s (Android متوسط).
* **Crash-free:** ≥ 99.5% آخر 7 أيام.
* **حجم التطبيق:** ≤ 40MB.
* **أحداث التحليلات:** تطابق ≥ 95% مع الـBackend.

## **اختبار ذاتي سريع (نعم/لا)**

1. هل يمكن استبدال طبقة API دون لمس UI؟
2. هل أي Widget لا يتجاوز 200-300 سطر؟
3. هل الأخطاء تُترجم إلى Failures موحّدة برسائل ودّية؟
4. هل يمكنك تشغيل flutter test والحصول على تغطية؟
5. هل توجد بيئات Dev/Stage/Prod منفصلة؟
6. هل أي سرّ غير موجود في الكود؟
7. هل يمكن تعطيل/تمكين ميزة عن بعد؟
8. هل لديك Dashboard للأعطال والأداء؟
9. هل Funnel الشراء مضبوط ومُراجع أسبوعيًا؟
10. هل البيلد يتم بالكامل من CI بدون خطوات يدوية؟

النقاط الرئيسية

## **النقاط الرئيسية**

1. **واجهة المستخدم (UI Layer)**
   * الشاشات (Screens)
   * المكوّنات (Components/Widgets)
   * تعدد اللغات (Localization)
   * التصميم (Theme/Branding)
2. **إدارة الحالة (State Management)**
   * حالة المستخدم (Session/Auth State)
   * السلة (Cart State)
   * الطلبات (Orders State)
   * تفضيلات وإعدادات المستخدم
3. **طبقة البيانات (Data Layer)**
   * API Client (الاتصال بالـ Backend)
   * Local Cache / Database (SharedPrefs أو SQLite)
   * Models & DTOs (User, Product, Order…)
4. **الخدمات الداخلية (Core Services)**
   * **Authentication** (تسجيل الدخول/الخروج)
   * **Notifications** (FCM Push)
   * **Remote Config** (تشغيل/إيقاف ميزات)
   * **Analytics** (تتبع الأحداث والاستخدام)
   * **Deep Linking & Routing** (فتح شاشات عبر الروابط)
5. **الأمان (Security Layer)**
   * حماية الجلسات والرموز (Tokens)
   * App Check Integration
   * تشفير البيانات المحلية (عند الحاجة)
6. **الأداء والجودة (App Quality)**
   * Crashlytics (تتبع الأعطال)
   * Performance Monitoring (زمن الفتح/الشاشات الحرجة)
   * Unit Tests & UI Tests
7. **البنية المشتركة (App Architecture Pattern)**
   * MVVM أو Clean Architecture (View → ViewModel → Repository → Data Source)
   * فصل واضح بين الـ UI, Logic, Data

النمط المعماري داخل التطبيق ( MVVM)

# **1) النمط المعماري داخل التطبيق (Clean Architecture + MVVM)**

## **لماذا (الفائدة)**

* فصل صارم بين العرض والمنطق والبيانات → سهولة الصيانة والاختبار.
* قابلية التوسع وإضافة ميزات بدون كسر الشاشات.
* تقليل الأخطاء بتحديد اتجاه الاعتماديات (Dependencies) بوضوح.

## **ما المطلوب تنفيذه**

* **الطبقات:**
  + **Presentation:** View (Screens/Widgets) → ViewModel (State)
  + **Domain:** UseCases (قواعد العمل الصافية)
  + **Data:** Repository ← RemoteDataSource (API) + LocalDataSource (Cache/Storage)
* **إدارة الحالة (State):** تبنّي أسلوب موحّد (مثلاً في Flutter: **Riverpod** أو **Bloc**) مع حالات قياسية: idle / loading / success / error.
* **التنقل (Navigation):** تعريف خريطة صفحات وتوجيه موحّد (Routes + Deep Links) مع حارس دخول (Auth Guard).
* **الأخطاء (Error Handling):** نوع خطأ موحّد (Network / Validation / Auth / Unknown) ورسائل عربية جاهزة.
* **النتائج (Result Type):** كائن نتيجة موحّد (نجاح/فشل) يصل من الـUseCase للـViewModel.
* **قواعد التسمية:** snake\_case للحقول، أسماء ملفات/مجلدات ثابتة لكل طبقة.
* **تعقب الأحداث:** نقاط قياس قياسية ضمن الـViewModel (فتح شاشة، إضافة للسلة…) لإرسالها لـAnalytics لاحقًا.

## **Definition of Done (اعتماد)**

* هيكل مجلدات الطبقات موجود وثابت.
* شاشة نموذجية تعمل عبر المسار الكامل: View → ViewModel → UseCase → Repository → Remote/Local.
* حالات loading/success/error تظهر بصريًا بشكل موحّد.
* خريطة Routes + Deep Links + Auth Guard مُعتمدة.
* اختبارات وحدة لواحد على الأقل من UseCases وViewModels.
* توثيق موجز (Readme داخلي) يشرح تدفق البيانات ونقاط القياس.

## **تسليمات**

* قائمة الطبقات والمجلدات المطلوبة.
* قالب State موحّد جاهز (أسماء الحالات والخصائص).
* جدول رسائل الخطأ العربية القياسية (لكل نوع خطأ).
* جدول الأحداث القياسية التي سترسل للتحليلات (فتح شاشة/ضغط زر رئيسي).

## **أخطاء شائعة يجب تجنبها**

* استدعاء API مباشرة من الـView → ممنوع (يجب عبر UseCase/Repository).
* منطق تجاري داخل الـView → يُنقل إلى UseCase.
* تعدد أنماط إدارة الحالة بين الشاشات → اعتمد أسلوبًا واحدًا.

# **1) قائمة الطبقات والمجلدات المطلوبة**

هيكلة مرتبة (مقترحة لفلتر/كوتلن/سويفت بنفس الفكرة). لموبايل Flutter مثلًا:

lib/

core/ # أشياء مشتركة

errors/ # أنواع الأخطاء والمحوّل

network/ # http client, interceptors, headers

utils/ # أدوات عامة (formatters..)

constants/ # ثوابت (currency، مفاتيح..)

analytics/ # واجهة تتبع الأحداث

app\_check/ # تكامل App Check إن لزم

config/

env.dart # إعدادات البيئة (dev/stg/prod)

routes.dart # خريطة التوجيه + الحراس

theme.dart # الألوان والخطوط

localization/ # الترجمة (ar/en)

data/ # طبقة البيانات

models/ # DTOs (from/to JSON)

repositories/ # الواجهات + التنفيذ

datasources/

remote/ # استدعاءات API

local/ # cache/SharedPrefs/DB

mappers/ # التحويل DTO <-> Domain

domain/ # طبقة المجال

entities/ # نماذج المجال (UI-ready)

usecases/ # قواعد العمل

repositories/ # عقود الـRepo (Interfaces)

presentation/ # طبقة العرض

common\_widgets/ # مكوّنات UI قابلة لإعادة الاستخدام

state/ # قوالب الحالة العامة

features/ # كل ميزة في مجلد

home/

screens/

viewmodel/

catalog/

product\_detail/

cart/

checkout/

orders/

profile/

auth/

الفكرة: **View (screens)** → تتعامل فقط مع **ViewModel (state)** → ينادي **UseCases** → تمر عبر **Repository** → إلى **Remote/Local**.

# **2) قالب State موحّد (أسماء الحالات والخصائص)**

نستخدم نفس القالب في كل شاشة عشان تكون التجربة متسقة:

**أسماء الحالات:**

* idle — الواجهة جاهزة بدون بيانات أو قبل أول طلب.
* loading — جاري الجلب/التنفيذ.
* success — نجحت العملية وفيه بيانات.
* error — فشلت العملية وفيه معلومات عن الخطأ.

**الخصائص القياسية داخل الحالة:**

* data (اختياري) — البيانات المعروضة عند النجاح.
* error\_code — كود منطقي داخلي (مثل network\_offline, validation\_422, auth\_401…).
* error\_message — نص عربي جاهز للعرض للمستخدمة.
* last\_updated\_at — طابع زمني آخر تحديث (يفيد في الإنعاش/الـpull-to-refresh).
* is\_refreshing — هل التحديث الآن خلف الكواليس (SWr).

**قواعد استخدام:**

* أي طلب شبكة يبدأ بـ loading، ثم يتحول إلى success أو error.
* في القوائم: اعرض **skeleton** مع loading، و**Empty State** إذا success بدون بيانات، و**رسالة خطأ** مع زر إعادة المحاولة عند error.
* لا تُظهر رسائل تقنية؛ استخدم جدول الرسائل الجاهزة أدناه.

# **3) جدول رسائل الخطأ العربية القياسية**

| **error\_code** | **متى يُستخدم** | **الرسالة العربية (للواجهة)** | **ملاحظات واجهة** |
| --- | --- | --- | --- |
| network\_offline | عدم وجود إنترنت / فشل DNS | **اتصالك بالإنترنت غير متوفر.** تأكدي من الشبكة وحاولي مجددًا. | زر “إعادة المحاولة” |
| timeout | انقضاء مهلة الطلب | **الخدمة تتأخر بالاستجابة.** حاولي مرة أخرى بعد لحظات. | حاول تقليل المهلة في الخلفية |
| server\_5xx | خطأ من الخادم (500–599) | **حدث خطأ مؤقت في الخادم.** نعمل على إصلاحه. عاودي لاحقًا. | اعرض “أبلغي عن المشكلة” اختياري |
| not\_found\_404 | مورد غير موجود (منتج محذوف مثلًا) | **العنصر غير متاح الآن.** | أعِد التوجيه إلى شاشة مناسبة |
| validation\_422 | إدخال غير صحيح/قواعد عمل (مثال: كمية غير متاحة) | **رجاءً راجعي البيانات المدخلة.** | إن وُجدت تفاصيل، اعرضيها تحت الحقول |
| auth\_401 | غير مصدّق/انتهت الجلسة | **يرجى تسجيل الدخول للمتابعة.** | افتحي مودال تسجيل الدخول |
| forbidden\_403 | صلاحية غير كافية | **لا تملكين إذن تنفيذ هذا الإجراء.** |  |
| rate\_429 | طلبات كثيرة خلال فترة قصيرة | **طلبات متكررة.** حاولي بعد قليل. |  |
| conflict\_409 | تضارب حالة (مثل مخزون نفد أثناء الشراء) | **نفد المخزون لهذا المنتج.** حدّثي الصفحة واختاري بديلًا. | أعرضي بدائل/مقاسات |
| unknown | أي خطأ غير مصنّف | **حدث خطأ غير متوقع.** حاولي من جديد. | سجّلي الحدث للتحقيق |

**هدفه:** كل الشاشة تستخدم نفس الرسائل لنفس الحالات، فتكون التجربة موحّدة وواضحة.للمطور: حوِّل أكواد HTTP أو أكواد الخادم (ZH-XXXX) إلى error\_code أعلاه، وكلها تعرض نصًا موحدًا.

# **4) جدول الأحداث القياسية للتحليلات (GA4)**

**هدفه:** توحيد أسماء الأحداث والمعلمات الأساسية. كل حدث باسم **snake\_case**.

| **event\_name** | **متى يُطلق** | **أهم المعلمات (params)** |
| --- | --- | --- |
| app\_start | عند فتح التطبيق بنجاح | app\_version, platform |
| screen\_view | عند فتح أي شاشة | screen\_name, previous\_screen, section |
| view\_item\_list | عرض قائمة (فئة/بحث/شبكة) | list\_id, list\_name, items\_count |
| select\_item | ضغط عنصر من القائمة | item\_id, item\_name, list\_name, index |
| view\_item | فتح صفحة منتج | item\_id, item\_name, category, price, in\_stock |
| add\_to\_cart | إضافة للسلة | item\_id, variant\_id, size, color, quantity, price, currency |
| begin\_checkout | بدء إجراءات الدفع | items\_count, value, currency |
| add\_shipping\_info | إدخال/اختيار عنوان الشحن | city, region, value, currency |
| add\_payment\_info | اختيار طريقة الدفع | payment\_type (cod/prepaid), value, currency |
| purchase | إتمام الطلب | transaction\_id, value, currency, items[] (id, name, price, qty), shipping, coupon? |
| search | تنفيذ بحث | search\_term, results\_count |
| notification\_open | فتح إشعار | campaign\_id, type (promo/transactional) |
| login / sign\_up | دخول/تسجيل | method (phone/email) |

**قواعد عامة:**

* لا نرسل أي بيانات تعريف شخصية (رقم/إيميل).
* العملة الافتراضية YER.
* transaction\_id فريد ولا يتكرر.
* كل شاشة تسجل screen\_view باسم ثابت (مثل home, product\_detail, cart, checkout\_shipping, checkout\_payment, checkout\_review, orders\_list, profile).

واجهة المستخدم (UI/UX Layer)

# **2) واجهة المستخدم (UI/UX Layer)**

## **لماذا (الفائدة)**

* واجهة متسقة، قابلة لإعادة الاستخدام، سريعة، ومهيّأة للعربية (RTL) وللبيع.

## **ما المطلوب تنفيذه**

* **Design System:** ألوان، خطوط، أحجام، مسافات، ظلال؛ مكوّنات أساسية (Buttons, Cards, Badges, Inputs, AppBar, Skeleton).
* **شاشات أساسية:** Home, Categories, Product Detail, Search, Cart, Checkout, Orders, Profile.
* **حالات واجهة:** Loading Skeletons، Empty States، Error States موحّدة.
* **RTL & Localization:** كل الشاشات RTL جاهزة، نصوص عبر Localization.
* **Accessibility:** أحجام خط قابلة للتكبير، تباين كافٍ، عناصر لمس بقياس مناسب.

## **Definition of Done (اعتماد)**

* Theme ومكتبة Components جاهزة ومستخدمة في ≥ 80% من الشاشات.
* Skeleton/Empty/Error حالات موحّدة ومفعّلة.
* RTL يعمل في كل الشاشات بدون انعكاسات خاطئة.
* قياسات أولية للأداء (زمن فتح Home، سلاسة سكرول القوائم).

## **تسليمات**

* قائمة Components المعتمدة + كيفية استخدامها.
* لائحة الشاشات ومحتواها وعناصرها (بدون تصميم بصري مفصل).
* نصوص عربية قياسية للحالات الثلاث (فارغ/تحميل/خطأ).

## **أخطاء شائعة يجب تجنبها**

* بناء عناصر UI مكررة بدل استخدام المكونات المعتمدة.
* إهمال حالات الخطأ/الفراغ → يؤدي لتجربة سيئة.

# **1) قائمة Components المعتمدة + كيفية استخدامها**

الهدف: توحيد الواجهة، تسريع التطوير، وتقليل التكرار. (الأسماء إرشادية؛ المبرمج يطبّقها بالإطار المستخدم)

**A) التخطيط والتنقّل**

* **AppBar**: عنوان الشاشة + زر رجوع/بحث/سلة.  
  الاستخدام: يظهر في كل شاشة رئيسية؛ يبدّل الأزرار حسب السياق.
* **BottomNav**: 4–5 علامات (الرئيسية، الفئات، السلة، الطلبات/الحساب).  
  الاستخدام: ثابت في التطبيق، يبرز العنصر الحالي.
* **Tabs**: تبديل بين أقسام فرعية (مثل “فساتين / عبايات / نوم”).  
  الاستخدام: عند وجود تصنيفات فرعية كثيرة.

**B) عرض البيانات**

* **ProductCard** (شبكة): صورة، اسم، سعر، شارة خصم/نفاد.  
  Props: image, title, price, badge?, onTap.  
  الاستخدام: في قوائم المنتجات والاقتراحات.
* **ProductTile** (سطر): للصنّفات الأفقية/القوائم الطويلة.  
  Props: مثل أعلاه + subtitle?.
* **PriceTag**: يعرض السعر + السعر قبل الخصم إن وُجد.  
  Props: price, currency, originalPrice?.
* **VariantSelector**: اختيار المقاس/اللون مع إظهار المتاح.  
  Props: sizes[], colors[], selected, onChange.
* **ImageGallery**: سلايدر صور المنتج + تكبير.  
  Props: images[], onIndexChange?.
* **RatingStars** (اختياري): تقييمات نجوم مع عدد.  
  Props: value, count.

**C) القوائم والشرائح**

* **CategoryChip**: شارة فئة قابلة للنقر.  
  Props: label, icon?, selected, onTap.
* **Carousel**: شريط بانرات بالرئيسية.  
  Props: items[] (image, link), autoPlay?.
* **HorizontalList**: قائمة أفقية لمنتجات مقترحة.  
  Props: items[], title, onMore?.

**D) حقول ونماذج**

* **TextField**: عام + نسخ مخصصة (هاتف، اسم، مدينة).  
  Props: label, hint, type, errorText?.
* **PhoneInput**: إدخال الهاتف بصيغة E.164 مع كود الدولة.
* **Dropdown**: مدينة/منطقة/مقاس.
* **Stepper**: اختيار كمية.
* **CouponField**: إدخال كوبون + زر “تطبيق”.

**E) أزرار**

* **PrimaryButton**: CTA رئيسي (إضافة للسلة/إتمام).  
  Props: label, loading?, onPressed.
* **SecondaryButton**: ثانوي (استمرار التسوق).
* **IconButton**: للقلب/المفضلة/المشاركة.

**F) حالات واجهة/تغذية راجعة**

* **Skeleton**: هيكل تحميل للقوائم والبطاقات.
* **EmptyState**: أيقونة + عنوان + وصف + زر إجراء.  
  Props: icon, title, subtitle, actionLabel?, onAction?.
* **ErrorState**: رسالة خطأ موحّدة + زر إعادة المحاولة.
* **Toast/Snackbar**: تأكيدات سريعة (تمت الإضافة للسلة).
* **ModalBottomSheet**: لاختيار المقاس/اللون قبل الإضافة.

**G) سلة وشراء**

* **CartItemRow**: صورة مصغّرة، اسم، مقاس/لون، سعر، كمية، حذف.
* **SummaryCard**: ملخص الطلب (بضاعة، شحن، خصم، الإجمالي).
* **AddressCard**: عنوان تسليم + تحرير/حذف.
* **PaymentMethodTile**: COD / مسبق.
* **CheckoutStepHeader**: يوضح الخطوة الحالية (عنوان، دفع، مراجعة).

**H) شارات/وسوم**

* **Badge**: “جديد”، “نفد قريبًا”، “خصم %”.
* **Tag**: لون/مقاس متاح/غير متاح.

**إرشادات استخدام عامة**

* كل شاشة تستخدم **Components المعتمدة** فقط.
* خصائص (props) موحّدة لتقليل الفروق.
* الألوان/الخطوط من **Theme**، ولا يُحدد لون يدوي داخل المكوّن.

# **2) لائحة الشاشات ومحتواها وعناصرها (بدون تصميم بصري)**

الهدف: يعرِف المبرمج ما يجب أن تحتويه كل شاشة من عناصر ووظائف.

**A) الرئيسية (Home)**

* عناصر: AppBar + Carousel + أقسام مختصرة (Chip للفئات) + قوائم أفقية للمنتجات (الأكثر مبيعًا/وصل حديثًا).
* أفعال: فتح فئة، فتح منتج، عرض المزيد.

**B) الفئات (Categories)**

* عناصر: شبكة فئات رئيسية/فرعية (CategoryChip/Tile).
* أفعال: الدخول لقائمة المنتجات حسب الفئة.

**C) قائمة المنتجات (Product List)**

* عناصر: ProductCard Grid + فلترة (مقاس، لون، سعر) + فرز (الأحدث/الأقل سعرًا).
* أفعال: التمرير اللانهائي (pagination) + فتح التفاصيل.

**D) تفاصيل المنتج (Product Detail)**

* عناصر: ImageGallery، الاسم، السعر/الخصم، تقييم (اختياري)، وصف مختصر، **VariantSelector** (مقاس/لون)، **PrimaryButton**(أضف للسلة)، مشاركة/مفضلة، اقتراحات أسفل.
* أفعال: اختيار المقاس/اللون، إضافة للسلة (يفتح BottomSheet إذا لم يختر المقاس)، انتقال للسلة.

**E) البحث (Search)**

* عناصر: مربع بحث + سجل حديث + نتائج (قائمة/شبكة) + حالة “لا نتائج” برسالة.
* أفعال: اقتراحات تلقائية، تصفية أساسية.

**F) السلة (Cart)**

* عناصر: قائمة **CartItemRow** (صورة، عنوان، خصائص، سعر، كمية، حذف)، حقل كوبون، **SummaryCard**، زر “إتمام الشراء”.
* أفعال: تعديل كمية، تطبيق كوبون، حذف.

**G) الشراء – العنوان (Checkout – Shipping)**

* عناصر: اختيار/إضافة **AddressCard** (مدينة/منطقة/وصف)، ملاحظات اختيارية، متابعة للدفغ.
* أفعال: حفظ عنوان افتراضي.

**H) الشراء – الدفع (Checkout – Payment)**

* عناصر: **PaymentMethodTile** (COD / مسبق)، ملخص، متابعة للمراجعة.

**I) الشراء – المراجعة والتأكيد (Checkout – Review & Confirm)**

* عناصر: قائمة العناصر، الرسوم، الخصم، الإجمالي، العنوان، طريقة الدفع، سياسة الإرجاع المختصرة، زر “تأكيد الطلب”.
* أفعال: إرسال الطلب، الانتقال لشاشة نجاح مع رقم الطلب.

**J) الطلبات (Orders List)**

* عناصر: قائمة الطلبات (رقم، تاريخ، إجمالي، حالة بشارة لون)، بحث تاريخي بسيط.
* أفعال: فتح تفاصيل الطلب.

**K) تفاصيل الطلب (Order Detail)**

* عناصر: حالة الطلب بخط زمني (timeline)، العناصر، العنوان، الشحن (تتبّع إن وُجد)، المبلغ، سياسة/إجراءات.
* أفعال: نسخ رقم الطلب، تواصل مع الدعم.

**L) الملف الشخصي (Profile)**

* عناصر: بيانات أساسية (الاسم/الهاتف/البريد)، إدارة العناوين، الإعدادات (لغة/إشعارات)، تسجيل الخروج.

**M) المفضلة (Wishlist)**

* عناصر: شبكة منتجات محفوظة، زر إضافة للسلة أو إزالة.
* حالة فارغة بنص لطيف + CTA “ابدئي التسوق”.

**N) تسجيل الدخول (Auth)**

* شاشات:
  + **Phone OTP**: إدخال رقم، إرسال كود، إدخال كود، إعادة إرسال مع مؤقت.
  + **Email/Password** (اختياري): تسجيل/دخول، استعادة كلمة مرور.
* أفعال: ربط الحساب لاحقًا (Phone + Email).

**O) الإعدادات (Settings)**

* عناصر: اللغة (ع/En)، إدارة الإشعارات، عن التطبيق، سياسة الخصوصية.

# **3) نصوص عربية قياسية للحالات الثلاث (فارغ/تحميل/خطأ)**

الهدف: اتساق النصوص ونبرة زهراء (راقية، بسيطة، بدون مبالغة). استخدم مباشرة في الواجهة.

**A) حالات التحميل (Loading)**

* عام للشاشات:
  + العنوان: **جاري التحميل…**
  + الوصف: **لحظات ونجهّز لك المحتوى.**
* بطاقات المنتجات (Skeleton): بدون نصوص؛ فقط الهياكل.

**B) حالات فارغة (Empty States)**

* الرئيسية (لا توجد بيانات مبدئيًا):
  + العنوان: **مرحبًا بك في زهراء**
  + الوصف: **اكتشفي تشكيلاتنا المختارة لكِ.**
  + زر: **ابدئي التسوق**
* قائمة المنتجات (لا توجد عناصر بعد فلترة/فئة فارغة):
  + العنوان: **لا توجد منتجات مطابقة**
  + الوصف: **جرّبي تعديل الفلاتر أو تصفحي فئات قريبة.**
  + زر: **تعديل الفلاتر**
* البحث (نتيجة = 0):
  + العنوان: **لم نجد ما تبحثين عنه**
  + الوصف: **حوّلي الكلمة أو جرّبي فئة مختلفة.**
  + زر: **تصفح الفئات**
* السلة الفارغة:
  + العنوان: **سلتك فارغة**
  + الوصف: **ابدئي بإضافة القطع المفضلة لديكِ.**
  + زر: **تسوقي الآن**
* المفضلة الفارغة:
  + العنوان: **لا عناصر في المفضلة**
  + الوصف: **احفظي القطع التي تعجبك للرجوع إليها لاحقًا.**
  + زر: **استكشفي الجديد**
* الطلبات (لا طلبات بعد):
  + العنوان: **لا توجد طلبات بعد**
  + الوصف: **عند إتمام أول طلب سيظهر هنا.**
  + زر: **ابدئي التسوق**

**C) حالات الخطأ (Error States)**

* انقطاع الإنترنت / الشبكة:
  + العنوان: **اتصالك بالإنترنت غير متوفر**
  + الوصف: **تحقّقي من الشبكة ثم حاولي مجددًا.**
  + زر: **إعادة المحاولة**
* مهلة الطلب (Timeout):
  + العنوان: **الخدمة تتأخر بالاستجابة**
  + الوصف: **حاولي مرة أخرى بعد لحظات.**
  + زر: **إعادة المحاولة**
* خطأ خادم مؤقت (5xx):
  + العنوان: **حدث خطأ مؤقت**
  + الوصف: **نعمل على إصلاحه الآن. حاولي لاحقًا.**
  + زر: **تحديث الصفحة**
* غير مصدّق/انتهت الجلسة (401):
  + العنوان: **يرجى تسجيل الدخول**
  + الوصف: **لإكمال هذه العملية، سجّلي دخولك.**
  + زر: **تسجيل الدخول**
* عناصر غير متاحة/نفاد مخزون (409/404):
  + العنوان: **العنصر غير متاح الآن**
  + الوصف: **اختاري مقاسًا/لونًا آخر أو عودي لاحقًا.**
  + زر: **عرض بدائل**

**D) رسائل قصيرة (Toast/Snackbar)**

* إضافة للسلة: **تمت الإضافة إلى سلتك.**
* إزالة من السلة: **تمت الإزالة من السلة.**
* تطبيق كوبون ناجح: **تم تطبيق الكوبون.**
* كوبون غير صالح: **الكوبون غير صالح.**
* حفظ عنوان: **تم حفظ العنوان.**
* خطأ عام قصير: **حدث خطأ، حاولي مجددًا.**

**E) أزرار عامة (CTAs)**

* **تسوقي الآن** / **عرض المزيد** / **تعديل الفلاتر** / **إعادة المحاولة** / **تأكيد الطلب** / **متابعة** / **حفظ** / **إلغاء**

## **كيف يسلّمها المبرمج داخل المشروع؟**

* يبني **مكتبة Components** بالأسماء أعلاه ويستخدمها حصريًا.
* يملأ كل شاشة بالعناصر المحددة تحت “لائحة الشاشات”.
* يربط **حالات الواجهة** بالنصوص القياسية (Empty/Loading/Error) دون ابتكار نصوص جديدة.
* يضع النصوص في ملف Localization (ar/en) ويستدعيها كمفاتيح (مثال: empty.cart.title, error.network.title…).

طبقة البيانات (Repositories + API Client + Cache)

# **3) طبقة البيانات (Repositories + API Client + Cache)**

## **لماذا (الفائدة)**

* واجهة موحّدة للوصول للبيانات بغضّ النظر عن مصدرها (شبكة/محلي).
* تمكين العمل **أوفلاين جزئي** وتحسين سرعة التطبيق.

## **ما المطلوب تنفيذه**

* **Repository لكل مجال:** ProductsRepository, CartRepository, OrdersRepository, AuthRepository…
* **RemoteDataSource:** عميل REST موحّد، ترويسات ثابتة (Authorization, App-Check, Request-Id)، ترقيم صفحات، معالجة أخطاء معيارية.
* **LocalDataSource (Cache):**
  + Cache للـ Lists (الصفحة الأولى على الأقل) + تفاصيل المنتج.
  + تفضيلات خفيفة (اللغة/العملة/المستخدم) في تخزين محلي آمن.
* **سياسة التحديث:** **Stale-While-Revalidate**
  + اعرض الكاش فورًا، وجدّد من الشبكة بالخلفية، وحدّث الـViewModel عند وصول البيانات.
* **النماذج (Models):**
  + **Domain Models** لا تعتمد على JSON.
  + **DTOs** للتحويل من/إلى الـAPI فقط (Mapper واضح بينهما).
* **التعامل مع الأخطاء:** تحويل أكواد الشبكة إلى أخطاء مجال مفهومة (مثلاً ZH-VALID-422 → خطأ إدخال).
* **الأمان:** حفظ الرموز (Tokens) محليًا بشكل آمن، وتنظيفها عند تسجيل الخروج.

## **Definition of Done (اعتماد)**

* عميل API موحّد يعمل مع Timeouts وRetry محدود.
* Repositories فعّالة لشاشات: قائمة المنتجات، تفاصيل منتج، السلة، الطلب.
* Cache يعمل ويظهر فرق السرعة بين الفتح الأول والثاني.
* تحويل DTO ↔ Domain عبر Mapper مُختبر.
* تغطية أخطاء الشبكة شاملة (عدم وجود إنترنت/مهلة/422/401…)، والـUI تعرض رسائل عربية مناسبة.

## **تسليمات**

* جدول Endpoints الأساسية المطلوبة (أسماء فقط الآن، التفاصيل لاحقًا من مستند الـAPI).
* قائمة DTOs + Domain Models لكل مورد.
* سياسة الـCaching لكل شاشة (ما يُخزّن، مدة الصلاحية، متى يُحدّث).
* خريطة الأخطاء وتحويلها لرسائل واجهة.

## **أخطاء شائعة يجب تجنبها**

* استخدام نفس النموذج لـDomain وDTO → افصل بينهما.
* إشغال الـUI بانتظار الشبكة دائمًا → افتح بالكاش ثم حدّث.
* تجاهل الترقيم/الفلترة في القوائم → التزم بعقد موحّد.

## **1) جدول Endpoints الأساسية المطلوبة**

الغرض: المبرمج يعرف أي مسارات API سيبني عليها الشاشات. (نكتبها أسماء فقط، التفاصيل الدقيقة تكون في مستند الـAPI الرسمي).

| **الشاشة / الميزة** | **Endpoint الأساسي** | **الملاحظات** |
| --- | --- | --- |
| الصفحة الرئيسية / الفئات | GET /categories | جلب الفئات مع الصور |
| قائمة المنتجات | GET /products?category\_id=... | مع ترقيم صفحات |
| تفاصيل المنتج | GET /products/{product\_id} | يتضمن variants والمخزون |
| البحث | GET /search?q=... | مع عدد النتائج |
| السلة | GET /cart, POST /cart/items | إضافة/تعديل/حذف عناصر |
| القسائم | POST /coupons/validate | إرجاع قيمة وصلاحية |
| الدفع / الشراء | POST /checkout | حجز المخزون وإنشاء order |
| الطلبات | GET /orders, GET /orders/{id} | استرجاع تاريخ الطلب |
| الإشعارات (تسجيل جهاز) | POST /devices/register | لحفظ fcm\_token |

## **2) قائمة DTOs + Domain Models لكل مورد**

* **DTO (Data Transfer Object):** الشكل القادم من/إلى الـAPI (JSON).
* **Domain Model:** الكائن الداخلي الذي يستخدمه التطبيق (أبسط/أوضح).

**DTO:**{

"id": "123",

"name": "فستان تركي",

"price": 12000,

"currency": "YER",

"images": ["..."],

"variants": [

{"id": "v1", "size": "M", "color": "Black", "stock": 3}

]

}

**Domain Model:**class Product {

final String id;

final String name;

final int price;

final String currency;

final List<String> images;

final List<Variant> variants;

* }

## **3) سياسة الـCaching لكل شاشة**

تحدد **ماذا نخزن محليًا + مدة صلاحية التخزين + متى نحدّث**.

| **الشاشة** | **ما يُخزن** | **مدة الصلاحية** | **سياسة التحديث** |
| --- | --- | --- | --- |
| الصفحة الرئيسية | قائمة الفئات + منتجات أولية | 24 ساعة | جلب من الكاش أولًا ثم تحديث من الشبكة بالخلفية |
| تفاصيل المنتج | بيانات المنتج + الصور | 12 ساعة | جلب من الكاش ثم تحديث إذا تغير السعر/المخزون |
| السلة | عناصر السلة محليًا | دائم (حتى تسجيل خروج) | يُزامن عند كل دخول للشاشة |
| الطلبات | آخر 5 طلبات | 1 يوم | تحديث يدوي عبر سحب للتحديث |

## 

## **4) خريطة الأخطاء وتحويلها لرسائل واجهة**

نحدد كيف يتحول كود الخطأ من الـAPI إلى رسالة عربية مفهومة تعرض في التطبيق.

| **كود الـAPI** | **error\_code داخلي** | **الرسالة للواجهة** |
| --- | --- | --- |
| HTTP 401 / ZH-AUTH | auth\_401 | **يرجى تسجيل الدخول للمتابعة.** |
| HTTP 422 / ZH-VALID | validation\_422 | **رجاءً راجعي البيانات المدخلة.** |
| HTTP 404 | not\_found\_404 | **العنصر غير متاح الآن.** |
| HTTP 409 / ZH-INV | conflict\_409 | **نفد المخزون لهذا المنتج.** |
| HTTP 500+ | server\_5xx | **حدث خطأ مؤقت في الخادم.** |
| Timeout / NoNetwork | network\_offline | **اتصالك بالإنترنت غير متوفر.** |

💡**خارطة طريق واضحة**:

* يعرف أي Endpoints يحتاج يبني لها استدعاءات.
* عنده Models جاهزة للـData.
* يعرف كيف يدير الكاش لكل شاشة.
* عنده جدول يحوّل أي خطأ إلى رسالة واجهة موحدة.
* كل طلب request يحتوي على meta\_data حول معلومات الجهاز والجلسة

الخدمات الداخلية داخل التطبيق ( , Analytics)

# **4) الخدمات الداخلية داخل التطبيق (Auth, Notifications, Remote Config, Analytics)**

## **A) المصادقة (Authentication)**

**الفائدة:**

* ربط كل مستخدم بحساب موثوق.
* تمكين السلة، المفضلة، الطلبات.
* زيادة الأمان ومنع الوصول المجهول.

**المطلوب:**

* تسجيل الدخول برقم الهاتف + OTP (أساسي).
* دعم البريد/كلمة المرور (اختياري).
* توحيد الجلسة عبر Firebase Auth Token.
* عند انتهاء الجلسة → إعادة توجيه لشاشة الدخول.
* إمكانية ربط الحساب (هاتف + بريد).

**Definition of Done:**

* تسجيل/تسجيل دخول يعمل 100%.
* OTP يعمل مع إعادة الإرسال + مؤقت.
* تجديد الجلسة يتم تلقائيًا.
* رسالة موحّدة عند انتهاء الجلسة.

**تسليمات:**

* شاشة تسجيل دخول/OTP.
* Service خاص بالمصادقة.
* اختبارات دخول/خروج ناجح وخاطئ.

**أخطاء يجب تجنبها:**

* تخزين الرموز أو كلمات المرور مكشوفة.
* السماح باستخدام التطبيق بعد تسجيل الخروج.
* تجاهل إعادة المحاولة عند إرسال OTP.

## **B) الإشعارات (Push Notifications)**

**الفائدة:**

* إعادة تنشيط المستخدم (عروض، منتجات جديدة).
* إشعارات تشغيلية (نجاح الطلب، تحديث الشحن).

**المطلوب:**

* تسجيل fcm\_token مع بيانات الجهاز.
* دعم إشعارات تشغيلية + ترويجية.
* التعامل مع الإشعارات: foreground → In-App Banner / background → Notification Tray.
* Deep Link لفتح الشاشة المناسبة.

**Definition of Done:**

* تسجيل/حذف token يعمل.
* إشعار تجريبي يظهر في foreground + background.
* يفتح الشاشة الصحيحة عند الضغط.

**تسليمات:**

* كود تسجيل الجهاز مع API.
* مكوّن In-App Banner.
* جدول mapping بين نوع الإشعار والشاشة.

**أخطاء يجب تجنبها:**

* تكرار token لنفس الجهاز.
* عدم تحديث/حذف token عند الخروج.
* إرسال إشعارات بدون صلاحيات.

## **Remote Config (C**

**الفائدة:**

* تفعيل/إيقاف ميزات بدون تحديث التطبيق.
* تجارب A/B سريعة.

**المطلوب:**

* ربط مع Firebase Remote Config.
* إعداد قيم افتراضية محلية.
* تطبيق التغييرات فورًا أو عند إعادة فتح التطبيق.
* أمثلة: عرض Banner جديد، إخفاء ميزة الدفع المسبق.

**Definition of Done:**

* قيم افتراضية موجودة وتعمل بدون إنترنت.
* أي تغيير من Firebase يظهر خلال ≤ 1 دقيقة.
* يتم تسجيل التغييرات في Analytics.

**تسليمات:**

* ملف ConfigService.
* متغيرات واضحة (مثل: show\_cod\_banner, enable\_prepaid).
* أمثلة اختبارية: إخفاء/إظهار عنصر.

**أخطاء يجب تجنبها:**

* عدم وجود قيم افتراضية محلية.
* الحاجة لإعادة تنصيب التطبيق للتغيير.
* تجاهل تسجيل التغييرات في Analytics.

## **Analytics (D**

**الفائدة:**

* تتبع سلوك المستخدمين.
* قياس الحملات التسويقية.
* اكتشاف مشاكل الاستخدام مبكرًا.

**المطلوب:**

* ربط GA4 + Firebase Analytics.
* إطلاق أحداث قياسية (screen\_view, view\_item, add\_to\_cart, purchase).
* إطلاق أحداث مخصّصة (coupon\_applied, cod\_selected).
* User Properties (اللغة، المدينة، نوع الدفع المفضل).

**Definition of Done:**

* كل شاشة تطلق screen\_view.
* الأحداث الرئيسية تُسجّل مع القيم المالية.
* User Properties تظهر في لوحة GA4.

**تسليمات:**

* ملف AnalyticsService.
* توثيق الأحداث والمعلمات.
* اختبار داخلي عبر DebugView.

**أخطاء يجب تجنبها:**

* إرسال أحداث بدون معلمات أساسية.
* ازدواج الأحداث عند التنقل بين الشاشات.
* عدم استبعاد الأجهزة الداخلية من التتبع.

الأمان داخل التطبيق (Security Layer)

**5) الأمان داخل التطبيق (Security Layer)**

## **الفائدة**

* حماية بيانات العميل (الحساب، العناوين، الطلبات).
* منع إساءة استخدام الواجهات (API) وتقليل الفواتير الناتجة عن الهجمات.
* الالتزام بالممارسات الجيدة للخصوصية والثقة بالعلامة.

## **المطلوب من التنفيذ**

### **A) جلسات المستخدم والرموز (Tokens)**

* استخدام **Firebase ID Token** حصريًا في ترويسة Authorization: Bearer <token>.
* **تحديث تلقائي** للرمز قبل انتهاء صلاحيته؛ عند الفشل → الانتقال لصفحة الدخول.
* حفظ الرمز ومعلومات الجلسة في **تخزين آمن** (Keychain/iOS, Keystore/Android) وليس في SharedPrefs العادية.
* عند **تسجيل الخروج**: مسح كل الرموز والكاش الحساس (سلة/عناوين).

### **App Check (B (إثبات أصل التطبيق)**

* إرسال ترويسة X-App-Check مع كل طلبات الشبكة (إن فُعّل في الـBackend).
* معالجة أخطاء App Check برسائل ودّية (اطلبي التحديث/أعيدي المحاولة).

### **C) الاتصالات الشبكية**

* إلزام **HTTPS فقط**؛ تعطيل أي قبول لشهادات غير موثوقة.
* **(اختياري مفضّل)**: تفعيل **SSL Pinning** لزيادة الحماية من MITM.
* ضبط مهلات ومعاودة محاولات ذكية (Exponential Backoff) مع سقف محاولات.

### **D) الروابط العميقة والتنقّل (Deep Links)**

* السماح بروابط من **قائمة نطاقات موثوقة فقط**.
* التحقق من **المعلمات** قبل استخدامها (لا تعتمد قيم URL مباشرة في منطق حساس).
* أي رابط ناقص/مريب → شاشة آمنة برسالة خطأ عامة.

### **E) البيانات المحلية (Local Data)**

* تشفير ما يمكن تشفيره (العناوين، السلة إن كانت تتضمن أسعار تفصيلية).
* تنظيف الكاش عند تسجيل الخروج أو عند تبديل الحساب.
* منع النسخ الاحتياطي السحابي للبيانات الحساسة (Android: allowBackup=false، iOS: استثناء الملفات الحساسة من iCloud backup).

### **F) الخصوصية والـPII**

* **لا** نخزن أو نرسل أبدًا: رقم OTP، كلمات مرور، مفاتيح سرية.
* الحد الأدنى من الـPII في الذاكرة والتخزين.
* إخفاء/تقليل ظهور البيانات الحساسة في **السجلات (Logs)**؛ **حظر** طباعة التوكن أو الـheaders.

### **G) السلوك على الشاشة**

* (اختياري حسب الحاجة) منع لقطات الشاشة في شاشات حسّاسة (Android FLAG\_SECURE).
* إخفاء الحقول الحساسة عند الانتقال للخلفية (Blur View عند الـApp Switcher) — حسب السياسة.

### **H) صلاحيات النظام (Permissions)**

* طلب الحد الأدنى من الصلاحيات وبالزمن المناسب (Just-in-time).
* شروحات عربية واضحة لسبب طلب الصلاحية.

### **I) إدخال المستخدم (Validation)**

* التحقق محليًا من تنسيقات شائعة (الهاتف بصيغة E.164، الحقول المطلوبة).
* Sanitization للنصوص الحرّة قبل الإرسال (منع رموز تحكم/أطوال غير منطقية).

### **J) التعامل مع الأخطاء**

* رسائل **عامة** للمستخدمة (بدون تفاصيل تقنية).
* تعيين أكواد داخلية للأخطاء (mapping) كما تم تعريفه مسبقًا.
* تسجيل الأخطاء تقنيًا داخليًا بدون أي PII.

### **K) حماية إجراءات حسّاسة**

* قبل إجراءات مثل **حذف الحساب/تغيير رقم الهاتف**: إعادة تحقق (Re-auth) وواجهة تأكيد مزدوج.
* **Checkout/Payment**: إنشاء **Idempotency-Key** على الجهاز لكل عملية وإرساله مع الطلب (لتفادي التكرار عند ضعف الشبكة).

## **Definition of Done (اعتماد)**

* جميع طلبات الشبكة ترفق Authorization (+ X-App-Check عند التفعيل).
* الرموز مخزّنة في Keychain/Keystore، وتُمسح كليًا عند تسجيل الخروج.
* الاتصالات HTTPS صارمة؛ (إن فُعّل) SSL Pinning يعمل دون كسر التدفق.
* Deep Links تعمل فقط من نطاقات موثوقة، ومع فحص معلمات آمن.
* لا وجود لتوكنات/PII في السجلات، وتم تفعيل Redaction للّوجات.
* عند انتهاء الجلسة تظهر رسالة دخول موحّدة والتطبيق يحول المستخدم لسياق آمن.
* إجراءات حسّاسة تتطلب re-auth وتأكيدًا داخل الواجهة.
* Idempotency-Key يُرسل في مسارات الشراء/الدفع وتم اختبار تكرار الطلب.

## **تسليمات**

* **SecurityConfig**: إعدادات الشبكة (TLS فقط، مهلات، Backoff).
* **TokenStore**: طبقة تخزين آمن للرموز مع واجهة (get/set/clear).
* **AppCheckInterceptor**: إضافة ترويسة App Check تلقائيًا.
* **DeepLinkGuard**: طبقة تحقق لنطاقات ومعلمات الروابط.
* **LogRedactor**: أداة لإخفاء الحساسيّات في السجلات.
* **SecureCache**: واجهة لتخزين محلي مشفر لما يلزم (إن تقررنا تشفير عناصر معينة).
* وثيقة **Permissions Policy** (ما الذي نطلبه ولماذا ومتى).
* **اختبارات**:
  + انتهاء جلسة → إعادة توجيه صحيحة.
  + تسجيل خروج → مسح تام للرموز والكاش.
  + Deep Link غير موثوق → يُمنع بأمان.
  + (إن وُجد) SSL Pinning → فشل الشهادة المزوّرة يمنع الاتصال.
  + إرسال Idempotency-Key وتأكيد منع التكرار.

## **أخطاء يجب تجنبها**

* تخزين التوكن في SharedPreferences/NSUserDefaults بدون تشفير.
* طباعة التوكن/الهيدر في الـLogs أو إرسالها لتحليلات.
* قبول أي Deep Link بلا تحقق نطاق/معلمات.
* رسائل خطأ تفصح عن تفاصيل سيرفر/استثناءات.
* تجاهل مسح الكاش الحساس عند تسجيل الخروج.
* استدعاء Checkout/Payment **بدون** Idempotency-Key.

## **ملاحظات تشغيلية (للتنسيق مع الـBackend)**

* في الرد على 401/403: ارجاع كود واضح ليفهم العميل أنه يحتاج تسجيل دخول.
* تفعيل **Rate Limiting** و**WAF** في الخادم؛ العميل يطبّق Backoff ويحترم 429.
* اعتماد **سقف أحجام JSON** مع رسائل مفهومة إذا تم تجاوزه.
* توحيد أكواد الأخطاء (ZH-XXXX) مع جدول الـmapping في التطبيق.

## **1. SecurityConfig**

**الاستفادة:** يضمن أن كل الاتصالات مع الخادم آمنة (HTTPS فقط) مع مهلات وضبط محاولات إعادة الطلب.  
**الطريقة:** إعداد عميل الشبكة (HttpClient/Retrofit/Alamofire) ليقبل TLS فقط، يرفض الشهادات غير الموثوقة، مع مهلة (Timeout) محددة + سياسة Backoff عند الفشل.

**2. TokenStore**

**الاستفادة:** حماية رموز الدخول (Tokens) ومنع سرقتها أو قراءتها من تطبيقات أخرى.  
**الطريقة:** تخزين الرموز داخل Keychain (iOS) أو Keystore/EncryptedSharedPrefs (Android). مسحها تلقائيًا عند تسجيل الخروج.

## **3. AppCheckInterceptor**

**الاستفادة:** التأكد أن الطلبات تأتي من تطبيق “زهراء” الحقيقي فقط، وتقليل الهجمات الآلية.  
**الطريقة:** إضافة Interceptor في عميل الشبكة لإرسال Header:

X-App-Check: <token>

مع كل طلب، والتحقق من رد الخادم عند الفشل.

**4. DeepLinkGuard**

**الاستفادة:** منع أي رابط مزيف من فتح التطبيق أو إدخال قيم ضارة.  
**الطريقة:** تحديد قائمة نطاقات مسموحة (whitelist مثل zahraah.com) + فحص المعلمات قبل التنقل. روابط غير مطابقة → تعرض شاشة آمنة برسالة خطأ.

**5. LogRedactor**

**الاستفادة:** حماية السجلات (Logs) من تسرب بيانات حساسة مثل الرموز أو الأرقام.  
**الطريقة:** إنشاء أداة (Utility) تستبدل القيم الحساسة بـ \*\*\* عند الطباعة أو الإرسال لـCrashlytics.

## **6. SecureCache**

**الاستفادة:** حماية البيانات المحلية (مثل العناوين أو السلة) من الوصول غير المصرح به.  
**الطريقة:** استخدام تخزين مشفر (EncryptedSharedPrefs/SQLCipher) لتخزين البيانات الحساسة، مع تنظيفها عند تسجيل الخروج.

## **7. Permissions Policy**

**الاستفادة:** تقليل رفض التطبيق من المتاجر وزيادة ثقة المستخدم عبر طلب أقل صلاحيات وبشفافية.  
**الطريقة:** كتابة ملف Permissions.md يوضح:

* ما هي الصلاحية؟ (مثال: الكاميرا)
* لماذا نطلبها؟ (رفع صورة الملف الشخصي)
* متى نطلبها؟ (عند الحاجة فقط – Just-in-time).

**8. Idempotency in Checkout/Payment**

**الاستفادة:** منع تكرار الطلب أو الدفع عند ضعف الإنترنت أو إعادة المحاولة.  
**الطريقة:** إنشاء مفتاح فريد Idempotency-Key عند كل عملية Checkout/Payment وإرساله في Header، ويتحقق الخادم أن العملية لا تُنفذ مرتين.

الأداء والجودة (App Quality)

# **6) الأداء والجودة (App Quality)**

## **Crashlytics (A – تتبّع الأعطال**

**الاستفادة:**

* كشف الأعطال الحقيقية على أجهزة المستخدمين وترتيبها حسب التأثير.
* تقليل فقدان المبيعات بسبب توقف التطبيق أو انهياره.

**الطريقة:**

* تفعيل Crashlytics وربط رموز الإصدارات (version/build).
* تسمية الشاشات والأحداث داخل التقارير لسهولة الفرز.
* ربط المستخدم المجهول بمعرّف داخلي غير شخصي (user\_id المُولّد).
* تفعيل التقارير غير الفورية لمراعاة الخصوصية، مع إرسال عند الاتصال.

**معايير القبول (Definition of Done):**

* **Crash-free users ≥ 99.5%** أسبوعيًا.
* لا توجد **Fatal Issues** مفتوحة لأكثر من 72 ساعة في الإصدار الإنتاجي.
* كل Crash يحتوي **breadcrumbs** كافية (آخر 10 أحداث تنقّل/عمليات).
* تم توثيق **Runbook**: كيف نقرأ التقرير ونغلقه (Owner/ETA/Root Cause/Fix).

**تسليمات:**

* لوحة Crashlytics مُفلترة حسب الإصدار والبيئة.
* تسمية موحّدة للشاشات داخل التقارير.
* Runbook “إدارة الأعطال” (خطوات/مسؤوليات/أزمنة استجابة).

**أخطاء يجب تجنبها:**

* طباعة بيانات حساسة في السجلات.
* ترك إصدارات قديمة غير مُعطّلة في التقارير (يشوّه المؤشرات).
* تجاهل الأعطال الصامتة (Non-fatal) المرتبطة بالCheckout/Cart.

## **Performance Monitoring (B – مراقبة الأداء**

**الاستفادة:**

* ضمان سلاسة التجربة (خصوصًا صفحات القائمة، تفاصيل المنتج، والخطوات الحرجة للشراء).
* تقليل التسرّب في القمع الشرائي بسبب البطء.

**الطريقة:**

* تفعيل Performance Monitoring وقياس:
  1. **App Start** (وقت الإطلاق للواجهة الأولى).
  2. **Screen Load** للشاشات الحرجة: Home, Product Detail, Cart, Checkout (Shipping/Payment/Review).
  3. **Network Traces** لمسارات API الحرجة (products, checkout, orders).
* إضافة **Custom Traces** عند الحاجة (بحث، تحميل صور).
* اعتماد **Skeletons** و**Caching** لتقليل الإدراك بالبطء.

**معايير القبول (Definition of Done):**

* **App Start P95 ≤ 2.5s**.
* **Screen Load P95 ≤ 1.2s** للشاشات الحرجة على أجهزة متوسطة.
* **Network P95 ≤ 800ms** لنِداءات المنتجات، و≤ 1.2s لـ checkout (باستثناء مزود خارجي).
* معدل **ANR** شبه صفري، وإن وجد ≤ 0.1%.
* تنبيهات عند تجاوز الحدود (P95) خلال 1 ساعة من رصد الانحراف.

**تسليمات:**

* لوحة مؤشرات “Performance” تحوي App Start/Screen Load/Network P95.
* قائمة الشاشات الحرجة + أهداف الأداء لكل شاشة.
* خطة تحسين (Performance Budget) إذا تم تجاوز الحدود.

**أخطاء يجب تجنبها:**

* تتبّع عام بلا تسميات واضحة → يصعب التحليل.
* جمع صور ضخمة بدون ضغط أو تحكّم بالحجم.
* جلب بيانات كبيرة لشاشات لا تعرضها فورًا (غياب pagination).

## **C) الاختبارات – Unit Tests & UI Tests**

**الاستفادة:**

* منع رجوع الأعطال (Regression) عند كل إطلاق.
* تسريع التطوير بثقة أعلى في تغييرات الكود.

**الطريقة:**

* **Unit Tests (منطق):** لطبقة **Domain (UseCases)** و**Data (Repositories/Mappers)**.
* **UI Tests (واجهة):** مسارات حرجة (إضافة للسلة → Checkout COD → تأكيد الطلب).
* **Contract Tests:** تحقق من صيغ JSON الأساسية (DTO ↔ Domain) وتوافقها مع الـAPI.
* تشغيل الاختبارات ضمن CI لكل Merge.

**معايير القبول (Definition of Done):**

* تغطية Unit لطبقة **Domain ≥ 70%**، و**Data ≥ 50%**.
* سيناريوهات UI آلية تغطي:
  1. فتح التطبيق + عرض Home بدون أخطاء.
  2. بحث + فتح تفاصيل منتج + إضافة للسلة.
  3. Checkout COD كامل حتى شاشة النجاح.
* الاختبارات تمرّ **100%** قبل أي إصدار إنتاجي.

**تسليمات:**

* تقرير تغطية اختبارات يُنشر آليًا في CI.
* قائمة سيناريوهات UI موثّقة (Given/When/Then) للشاشات الحرجة.
* ملفات Fixtures للـDTOs الأساسية (منتج/سلة/طلب).

**أخطاء يجب تجنبها:**

* كتابة اختبارات UI هشة تعتمد على نصوص تتغير كثيرًا.
* إهمال اختبارات الأخطاء (401/422/Timeout).
* تجاهل تحديث Fixtures عند تغيّر عقد الـAPI.

## **D) مؤشرات الجودة (Quality KPIs) – للمتابعة الأسبوعية**

* **Crash-free users** (الهدف ≥ 99.5%).
* **App Start P95** (الهدف ≤ 2.5s).
* **Screen Load P95** للشاشات الحرجة (≤ 1.2s).
* **Network P95** لنِداءات المنتجات/الشراء.
* **ANR Rate** (≤ 0.1%).
* **نسبة نجاح سيناريوهات UI الآلية** (100% قبل الإطلاق).

## **E) التنبيهات (Alerts)**

* تنبيه فوري عند:
  + هبوط **Crash-free users** عن 99%.
  + تجاوز **App Start P95** هدفه لمدّة > 30 دقيقة.
  + ارتفاع أخطاء Checkout أو 5xx على مسارات الشراء.
* قناة تنبيه واحدة متفق عليها (Slack/Email) مع **Owner** لكل نوع تنبيه.

## **F) خطة الإصدار (Release Gates)**

لا يُنشر الإصدار إلى الإنتاج إلا إذا:

1. **الاختبارات الآلية** (Unit/UI) ناجحة 100%.
2. **Crash-free users للنسخة التجريبية ≥ 99.5%** خلال 24 ساعة.
3. مؤشرات **App Start/Screen Load P95** ضمن الحدود.
4. لا توجد **Fatal Issues مفتوحة** متعلقة بـ Checkout/Cart.

إدارة الحالة (State Management)

# **7) إدارة الحالة (State Management)**

## **الفائدة**

* تنظيم تدفق البيانات داخل التطبيق بحيث تكون **الشاشات دائمًا متزامنة** مع حالة المستخدم (سلة، طلبات، جلسة…).
* تقليل الأخطاء الناتجة عن فقدان البيانات أو عدم تحديث الواجهة.
* تحسين تجربة الاستخدام (تحديث فوري – بدون ارتباك).

## **المطلوب من التنفيذ**

### **A) حالة المستخدم (Session/Auth State)**

* حفظ بيانات الجلسة بعد تسجيل الدخول (User ID, Token, Profile).
* تحديث الحالة تلقائيًا عند انتهاء صلاحية التوكن → إعادة توجيه لشاشة تسجيل الدخول.
* تحميل بيانات المستخدم (Profile, Addresses) بمجرد نجاح تسجيل الدخول.

### **B) السلة (Cart State)**

* تتبع العناصر المضافة (product\_id, variant\_id, qty, price\_at\_add).
* تحديث الأسعار عند كل فتح للسلة أو عند الـCheckout (مع مقارنة السعر الحالي).
* مزامنة السلة مع الخادم عند تسجيل الدخول أو الخروج (User ↔ Guest Cart).
* إظهار حالة Loading/Empty/Error في الواجهة بشكل موحّد.

### **C) الطلبات (Orders State)**

* حفظ آخر الطلبات (Recent Orders) مع الحالة الحالية لكل طلب.
* تحديث الحالة عند وصول إشعار أو Refresh يدوي.
* إمكانية استعراض الطلبات حتى بدون إنترنت (من الكاش).

### **D) التفضيلات والإعدادات (Preferences/Settings State)**

* اللغة (ar/en).
* وضع الإشعارات (مفعّل/متوقف).
* طريقة الدفع المفضلة (COD/Prepaid).
* تفضيلات واجهة مثل آخر فلترة أو ترتيب منتجات.

## **معايير القبول (Definition of Done)**

* **Session State**: يتم تسجيل الدخول والخروج وتحديث الجلسة بشكل متسق في جميع الشاشات.
* **Cart State**: عند إضافة منتج يظهر فورًا في كل شاشة مرتبطة، وتحديث السعر عند التغيير.
* **Orders State**: شاشة الطلبات تعرض آخر حالة بدون الحاجة لإعادة الدخول.
* **Preferences State**: تغييرات الإعدادات تظهر مباشرة وتُحفظ للجلسة التالية.
* كل حالة مدعومة بـ **States قياسية**: idle, loading, success, error.

## **التسليمات**

1. **State Containers** (ViewModel/Bloc/Riverpod) لكل من: User, Cart, Orders, Preferences.
2. **تعريف الحالات (State Classes)** تشمل: idle, loading, success(data), error(message).
3. **مستند يوضح Events الأساسية** (AddToCart, RemoveFromCart, RefreshOrders…).
4. اختبارات Unit لـ Cart وOrders للتأكد أن الحالة تتغير بشكل صحيح.
5. آلية مزامنة بين الكاش المحلي والخادم.

## **أخطاء يجب تجنبها**

* تخزين الحالة في الواجهة مباشرة (Widget/Screen) → يسبب فقدان البيانات عند إعادة تحميل الشاشة.
* عدم تحديث السعر عند Checkout → يسبب تعارض مع الخادم.
* السماح بوجود أكثر من حالة متناقضة (مثال: المستخدم مسجّل دخول لكن الـSession State فارغ).
* تجاهل إدارة Empty/Error States → يخلق تجربة مربكة للمستخدم.

# **التسليمات – إدارة الحالة (State Management)**

1. **State Containers**
   * إنشاء حاويات حالة (ViewModel / Bloc / Riverpod Provider) لكل مجال رئيسي:
     + المستخدم (User State)
     + السلة (Cart State)
     + الطلبات (Orders State)
     + التفضيلات (Preferences State)
2. **تعريف الحالات (State Classes)**
   * كل حاوية تحتوي حالات قياسية:
     + idle (بدون بيانات بعد)
     + loading (جاري التنفيذ)
     + success(data) (تم التنفيذ مع بيانات)
     + error(message) (فشل مع رسالة)
3. **مستند Events الأساسية**
   * قائمة بالأحداث التي تغيّر الحالة، مثل:
     + AddToCart – إضافة منتج للسلة
     + RemoveFromCart – إزالة منتج من السلة
     + RefreshOrders – تحديث قائمة الطلبات
     + UpdatePreferences – تعديل الإعدادات
4. **اختبارات Unit**
   * بناء اختبارات لوحدة Cart وOrders للتحقق من:
     + إضافة/إزالة عناصر تغيّر الحالة بشكل صحيح.
     + تحديث الطلبات يظهر النتيجة الصحيحة.
5. **آلية مزامنة الكاش مع الخادم**
   * عند توفر إنترنت: تحديث الحالة من الخادم.
   * عند عدم توفر إنترنت: استخدام الكاش المحلي.
   * عند تسجيل الدخول أو الخروج: دمج السلة (Guest ↔ User) ومزامنة البيانات.

الملاحظات (البنية)

## **1) الدفع عند الاستلام (COD)**

**القاعدة**: اختلاف سياسة الدفع حسب المحافظة.  
**الفائدة**: تقليل إلغاء الطلبات وتحقيق ثقة أكبر.  
**التطبيق**:

* أوضاع الدفع:
  + cod\_standard: الدفع كاملًا عند الاستلام (صنعاء).
  + cod\_deposit\_required: دفع عربون + الباقي عند الاستلام (بقية المحافظات). لم يحدد بعد
  + prepaid\_optional: خيار الدفع المسبق عند الرغبة.
* واجهة المستخدم تعرض رسالة/Modal توضح سياسة الدفع حسب المنطقة.
* Analytics event: add\_payment\_info.payment\_type مع قيمة (cod\_standard / cod\_deposit / prepaid).

## **2) سياسة الصور والأداء (Image/CDN Policy)**

**القاعدة**: الصور يتم تحميلها عبر CDN مع تحجيم وضغط ديناميكي.  
**الفائدة**: سرعة تحميل أعلى + تقليل استهلاك البيانات + حجم تطبيق أصغر.  
**التطبيق**:

* PLP: w=360, q=75 (≤ 250KB).
* PDP: w=1080, q=80 (≤ 800KB).
* استخدام WebP/AVIF مع fallback.
* placeholder/skeleton عند التحميل.
* التحقق من حجم التطبيق ≤ 40MB عبر CI.

## **3) التحليلات (Analytics & Events)**

**القاعدة**: أحداث موحدة + أبعاد ثابتة.  
**الفائدة**: قياس funnel بدقة، ودعم قرارات الأعمال.  
**التطبيق**:

* الأحداث الأساسية: view\_item, add\_to\_cart, begin\_checkout, add\_shipping\_info, add\_payment\_info, purchase.
* الأبعاد الثابتة:
  + user\_id (مشفّر/مجهول).
  + app\_version.
  + region (المحافظة).
  + currency = YER.
  + channel (organic / ads / meta).
* مطابقة البيانات مع السيرفر ≥ 95%.

## **4) الإشعارات (Push Notifications)**

**القاعدة**: فصل الإشعارات الترويجية عن التشغيلية.  
**الفائدة**: تجربة مستخدم أوضح + تقليل الإزعاج.  
**التطبيق**:

* التصنيفات:
  + promo: عروض وتخفيضات.
  + transactional: تأكيد الطلب، تحديث حالة الشحن.
* هيكل الرسالة: {title, body, deeplink, campaign\_id, expire\_at}.
* دعم deeplink لشاشات المنتج/الطلب/العرض.
* قاعدة منع الإزعاج: لا رسائل promo خلال 48 ساعة بعد الشراء.

## **5) سيناريوهات الأخطاء التجارية (Commerce Errors)**

**القاعدة**: رسائل واضحة + حلول بديلة.  
**الفائدة**: تقليل فقدان العملاء عند الفشل.  
**التطبيق**:

* coupon\_invalid: الكوبون غير صالح.
* coupon\_expired: انتهت صلاحية الكوبون.
* stock\_conflict (409): نفاد المخزون، اقتراح مقاس/لون بديل.
* address\_unsupported: التوصيل غير متاح للمدينة.
* deposit\_required: المنطقة تتطلب عربون.
* UI يعرض زر بديل (تعديل العنوان/الكوبون/المقاس).

## **6) إدارة الجلسات (Session & Token Refresh)**

**القاعدة**: الحفاظ على أمان الجلسات واستمراريتها.  
**الفائدة**: منع طرد المستخدم بشكل مفاجئ + أمان أعلى.  
**التطبيق**:

* Token refresh عند انتهاء الصلاحية.
* في حال فشل التحديث → تسجيل خروج نظيف مع إشعار.
* منع تخزين tokens في SharedPreferences، والاكتفاء بـ Secure Storage.

## **7) اختبارات الأداء والميزانيات (Performance Budgets)**

**القاعدة**: تحديد حدود صارمة للأداء والحجم.  
**الفائدة**: تجربة أفضل على الأجهزة الضعيفة + سرعة تشغيل.  
**التطبيق**:

* cold start ≤ 2 ثانية (Android متوسط).
* بناء PDP ≤ 120ms.
* scroll jank < 0.5%.
* حجم التطبيق ≤ 40MB.
* CI يفشل إذا زاد حجم البيلد > 5% عن الإصدار السابق.

الملاحظات الخاصة بالشبكة (Networking)

# **الملاحظات الخاصة بالشبكة (Networking)**

## **1) إعادة المحاولة (Retry)**

* **الحاجة**: في السوق المحلي غالبًا الإنترنت ضعيف أو غير مستقر.
* **الإضافة المطلوبة**:
  + تفعيل **Exponential Backoff** (تأخير متزايد بين المحاولات).
  + عدد محاولات قصوى (مثال: 3 مرات).
  + استخدام Dio Interceptor لإعادة الطلبات الفاشلة تلقائيًا.
  + تسجيل السبب في Crashlytics أو Sentry عند الفشل النهائي.

## **2) مراقبة الاستهلاك (Network Monitoring)**

* **الحاجة**: لتقليل استهلاك البيانات عند العملاء.
* **الإضافة المطلوبة**:
  + قياس حجم الاستهلاك لكل عملية API.
  + عرض خيار **“وضع توفير البيانات”** (Data Saver Mode): تحميل صور منخفضة الجودة + تقليل طلبات الخلفية.
  + Analytics event: data\_saver\_enabled (true/false).

## **3) التعامل مع الانقطاع (Offline Mode)**

* **الحاجة**: الإنترنت قد ينقطع بشكل متكرر.
* **الإضافة المطلوبة**:
  + **Queue** للعمليات (إضافة للسلة، تعديل العنوان، المفضلة) تُرسل عند عودة الاتصال.
  + كاش محلي للصفحات المهمة (Catalog, Cart, Orders).
  + شاشة “أنت غير متصل” برسالة واضحة وزر إعادة المحاولة.

## **4) إدارة الجلسة (Session Handling)**

* **الحاجة**: الحفاظ على أمان الحساب مع انقطاع الشبكة أو تغيير الاتصال.
* **الإضافة المطلوبة**:
  + تحديث تلقائي للـAccess Token عند انتهاء صلاحيته.
  + إذا فشل التحديث → تسجيل خروج نظيف مع إشعار المستخدم.
  + إلغاء الطلبات المعلّقة (Cancel Tokens) عند فقد الاتصال.

## **5) الأخطاء الموحدة (Network Error Mapping)**

* **الحاجة**: توحيد تجربة المستخدم عند حدوث مشاكل.
* **الإضافة المطلوبة**:
  + **Timeout** → “الخدمة بطيئة، جرّب لاحقًا”.
  + **No Connection** → “تأكد من اتصال الإنترنت”.
  + **Server Error (5xx)** → “مشكلة في السيرفر، فريقنا يعمل على حلها”.
  + **Unauthorized (401)** → إعادة تسجيل الدخول.
  + **Conflict (409)** → معالجة حالة نفاد المخزون أو تحديث السعر.

## **6) الأمان (Security Layer)**

* **الحاجة**: ضمان أمان الطلبات.
* **الإضافة المطلوبة**:
  + دعم **TLS Pinning** (إذا أمكن).
  + التحقق من صحة الردود (schema validation).
  + منع تسجيل البيانات الحساسة في الـLogs.

طرق الاختبار (Flutter)

# **طرق الاختبار (Flutter)**

## **1) اختبار الوحدات Unit Tests**

**الهدف:** المنطق الصافي (Use-cases, Repositories, Validators).  
**الأدوات:** flutter\_test + mocktail/mockito.  
**ما نختبره:**

* تسعير، ضرائب/شحن، حساب العربون للمحافظات.
* منطق المخزون (Inventory Hold) والإلغاء بعد انتهاء المهلة.
* قواعد الكوبونات (انتهى/حد أدنى/استثناء فئات).
* محولات (DTO ↔︎ Entity) ومعالجة الأخطاء Result/Failure.  
  **قبول:** تغطية ≥ 80% للوحدات الحساسة (السلة، الدفع، المخزون).

## **2) اختبار الواجهات Widget Tests**

**الهدف:** الـUI بدون شبكة.  
**الأدوات:** flutter\_test, golden\_toolkit (للصور الذهبية).  
**ما نختبره:**

* PLP/قائمة المنتجات: عرض بطاقة المنتج، Placeholder للصورة، RTL.
* PDP: تبديل الألوان/المقاسات، رسالة نفاد المخزون، سعر الشحن الظاهر مبكرًا.
* السلة/الملخّص: تحديث الكميات، تطبيق كوبون، منع “متابعة الدفع” عند أخطاء.  
  **قبول:** ثبات العناصر الأساسية، عدم وجود استثناءات/overflow.

## **3) صور ذهبية Golden Tests (ثبات التصميم)**

**الهدف:** ثبات الشكل عبر الوقت وRTL.  
**الأدوات:** golden\_toolkit.  
**لقطات مطلوبة:** شاشة البداية، PLP، PDP، checkout، شاشة خطأ الشبكة، شاشة لا اتصال.  
**قبول:** اختلاف بكسلات ≤ 0.5%.

## **4) اختبارات تكامل Integration/E2E**

**الهدف:** رحلة عميل كاملة على جهاز حقيقي أو محاكي.  
**الأدوات:** integration\_test (+ **Patrol** أو **Maestro** لتسهيل السيناريوهات).  
**سيناريوهات أساسية:**

1. **تصفح → إضافة للسلة → Checkout (صنعاء)**
   * توقّع cod\_standard وعدم طلب عربون.
2. **Checkout (محافظة خارج صنعاء)**
   * إظهار Modal عربون → قبول → إتمام الطلب.
3. **كوبون غير صالح/منتهٍ**
   * رسالة خطأ صحيحة وعدم تغيير السعر.
4. **نفاد المخزون أثناء الإتمام**
   * ظهور Conflict 409 → اقتراح بديل/رجوع للسلة.
5. **تغيير العنوان قبل الدفع**
   * إعادة حساب الشحن تلقائيًا.
6. **Deeplink من إشعار**
   * يفتح المنتج/الطلب الصحيح.  
     **قبول:** نجاح الرحلة بنسبة 100% على أقلّ تقدير لجهازين (Android + iOS).

## **5) اختبارات الشبكة والعدم اتصال**

**الهدف:** ملاءمة التطبيق لبيئة إنترنت ضعيفة.  
**الأدوات:** محاكي الشبكة (Android Studio/Charles/Network Link Conditioner)، Interceptors.  
**السيناريوهات:**

* **Timeout**: إظهار رسالة “الخدمة بطيئة”.
* **No Connection**: شاشة “غير متصل” + زر محاولـة مجددًا.
* **Retry + Backoff**: فشل أول/نجاح ثاني.
* **Offline Queue**: إضافة للسلة/المفضلة/العنوان أوفلاين ثم تزامن عند عودة الشبكة.  
  **قبول:** عدم انهيار التطبيق، رسائل عربية واضحة، عدم فقدان تغييرات المستخدم.

## **6) اختبارات الأداء Performance**

**الهدف:** الالتزام بالميزانيات.  
**الأدوات:** flutter drive/integration\_test مع Timeline, DevTools (CPU/Memory), Firebase Performance (اختياري).  
**المقاييس:**

* **Cold start ≤ 2.0s** على جهاز 2GB RAM (Android متوسط).
* **بناء PDP ≤ 120ms**.
* **Jank < 0.5%** في قوائم طويلة.
* **حجم التطبيق ≤ 40MB** (فشل CI إذا زاد >5%).  
  **قبول:** ضمن الحدود المذكورة.

## **7) اختبار الصور/CDN**

**الهدف:** سرعة التحميل وتوفير البيانات.  
**الأدوات:** DevTools Network, Dio Logger.  
**ما نتحقق منه:**

* PLP: صور ≤ 250KB بـ w=360 q=75.
* PDP: الصورة الأولى ≤ 800KB بـ w=1080 q=80.
* WebP/AVIF تعمل مع fallback، Placeholder/Skeleton فعّال.  
  **قبول:** جميع القيود محترمة، عدم تحميل صور أصلية ضخمة.

## **8) اختبار التحليلات Analytics**

**الهدف:** دقة الـFunnel وربط الأبعاد الثابتة.  
**الأدوات:** DebugView (GA4/Firebase), Logcat, Stubs.  
**نقاط تحقق:**

* view\_item, add\_to\_cart, begin\_checkout, add\_shipping\_info(city), add\_payment\_info(type), purchase.
* **أبعاد ثابتة**: region, currency=YER, channel, app\_version, (user\_id مشفّر).
* **تطابق ≥ 95%** مع أرقام السيرفر أسبوعيًا.  
  **قبول:** عدم وجود Events مفقودة/مضاعفة.

## **9) اختبار الإشعارات والديب لينك**

**الهدف:** وصول صحيح بدون إزعاج.  
**الأدوات:** Firebase Cloud Messaging، سطر أوامر/لوحة FCM.  
**السيناريوهات:**

* transactional: إشعار حالة طلب يفتح شاشة الطلب.
* promo: يفتح صفحة العرض + احترام قاعدة “لا Promo خلال 48 ساعة بعد الشراء”.  
  **قبول:** deeplink صحيح، منع رسائل مكررة بعد الشراء.

## **10) الوصولية والـRTL Accessibility & RTL**

**الهدف:** تجربة ممتازة بالعربية وعلى قارئات الشاشة.  
**الأدوات:** Flutter Semantics, VoiceOver/TalkBack.  
**التحقق:**

* اتجاه RTL صحيح في كل الشاشات.
* تسميات Semantics للأزرار والصور.
* تباين النصوص مقروء (لا ألوان فاتحة على فاتح).  
  **قبول:** قابلية استخدام كاملة بدون عوائق.

## **11) الأمان Security**

**الهدف:** حماية الجلسات والبيانات.  
**الأدوات:** Secure Storage، فحص Logs، (اختياري TLS Pinning).  
**التحقق:**

* **Token refresh** يعمل؛ على الفشل → Logout نظيف.
* لا حفظ Tokens في SharedPreferences.
* لا طباعة بيانات حساسة في السجلات.  
  **قبول:** اجتياز جميع حالات الانتهاء/التجديد/الرفض.

## **12) الاستهلاك والطاقة Battery/Memory**

**الهدف:** عدم استنزاف الجهاز.  
**الأدوات:** Android Profiler, Xcode Instruments.  
**التحقق:**

* عدم تسريب ذاكرة (Leaks) في قوائم طويلة والتنقل المتكرر بين PLP↔︎PDP.
* استهلاك CPU طبيعي عند التمرير والتحميل.  
  **قبول:** بدون Leaks ملحوظة، CPU متوسط مستقر.

## **13) استعداد النشر Store-Readiness**

**الهدف:** جاهزية النسخة للمتاجر.  
**الأدوات:** flutter build, TestFlight, Firebase App Distribution.  
**التحقق:**

* الأذونات (Permissions) مبررة ومستخدمة فعليًا.
* crash-free sessions ≥ 99.5% على بيتا.
* لقطات صور/نصوص المتجر بالعربية، ASO أساسي.  
  **قبول:** معايير المتجر مكتملة + لا كراشات حرجة.

## **مصفوفة الأجهزة/الإصدارات (حد أدنى)**

* **Android:** 8.1, 10, 12 — أجهزة 2GB و4GB RAM (واحد ضعيف/واحد متوسط).
* **iOS:** 15 و16/17 — iPhone SE (ضعيف نسبيًا) + iPhone حديث.
* **شبكات:** 3G/ضعيف، 4G جيد، Wi-Fi ممتاز.

## **تكامل الاختبارات مع CI (مختصر)**

* تشغيل: flutter analyze && flutter test && flutter test --update-goldens (عند الحاجة) && flutter build appbundle --analyze-size.
* تشغيل E2E على **Firebase Test Lab** (Android) + **TestFlight** داخلي (iOS).
* فشل الـCI إذا:
  + تغطية الوحدات < 70% (أو حدّك المفضل).
  + حجم البيلد زاد > 5%.
  + أي سيناريو E2E أساسي يفشل.

## **سيناريوهات**

1. صنعاء → Checkout COD بدون عربون → نجاح شراء.
2. تعز/عدن → Checkout يطلب عربون → دفع عربون → نجاح شراء.
3. كوبون منتهي → رسالة خطأ → السعر لا يتغير.
4. نفاد مخزون أثناء الدفع → 409 → اقتراح بديل/رجوع للسلة.
5. تغيير المدينة في الملخص → إعادة حساب الشحن فورًا.
6. انقطاع الإنترنت في السلة → Offline Queue → عودة الشبكة → تزامن.
7. إشعار حالة طلب → يفتح شاشة الطلب الصحيحة (deeplink).
8. إشعار Promo بعد شراء خلال 24 ساعة → **لا يُرسل** (يحترم 48h).
9. انتهاء صلاحية الـToken أثناء الطلب → Refresh ناجح → تكملة؛ عند فشل التجديد → Logout نظيف برسالة.
10. PLP على شبكة ضعيفة → صور ≤ 250KB وتعمل الـPlaceholder.

اختبارات إضافية

# **اختبارات إضافية**

### **1) خريطة الأخطاء (Error Map Testing)**

* إنشاء جدول يربط كل كود HTTP أو كود داخلي (error\_code) برسالة واجهة موحّدة.
* كتابة اختبارات واجهة (Widget) لكل خطأ لضمان:
  + عرض نفس الرسالة عبر جميع الشاشات.
  + ظهور زر “إعادة المحاولة” أو بدائل عند الحاجة.

### **2) اختبارات الكاش والسياسات (Caching & TTLs)**

* إضافة اختبارات تضمن أن كل شاشة (Home, PLP, PDP, Cart) تحترم زمن الصلاحية (TTL) المحدد.
* التحقق أن الفتح الثاني للصفحة أسرع (يُعرض الكاش أولًا ثم يُحدَّث صامتًا).

### **3) اختبارات الروابط العميقة والإشعارات (Deep Links & Notifications)**

* إعداد اختبارات للتأكد أن كل رابط عميق أو إشعار يفتح الشاشة الصحيحة بالمعلمات المطلوبة.
* التحقق من السلوك عند فقدان أو خطأ في المعلمات (يجب أن يعيد التوجيه أو يظهر رسالة واضحة).

### **4) اختبارات الوصولية المتقدمة (Accessibility)**

* فحص أن جميع العناصر التفاعلية أكبر من **48dp**.
* فحص نسب تباين الألوان ضمن الحدود.
* التأكد من وجود تسميات (Semantics Labels) للعناصر الأساسية.

### **5) مراقبة ما بعد الإطلاق (Post-release Monitoring Tests)**

* التحقق من وصول التنبيهات عند:
  + ارتفاع معدل الأخطاء.
  + زيادة زمن بدء التشغيل.
  + انخفاض معدل التحويل في الدفع.
* إعداد اختبارات “تنبيهية” (Alert Tests) للتأكد أن الـMonitoring يرسل إشعارات عند تجاوز العتبات.

تحسينات

**- خريطة أخطاء موحّدة**إنشاء جدول إلزامي يربط بين أكواد HTTP / أكواد الخادم → error\_code → الرسالة المترجمة للمستخدم. التأكد أن كل شاشة تستخدم نفس واجهة Empty/Error مع زر إعادة المحاولة أو عرض بدائل.

**- خطة واضحة لـ TLS Pinning**تحديد استراتيجية عملية لتثبيت الشهادات: نوع الـPin (مثلاً SHA-256/SPKI)، آلية التحديث والتدوير، خطة fallback/rollback، واختبار حالات الفشل على بيئة Staging.

**- سياسة كاش لكل شاشة**توثيق مدد صلاحية (TTL) وقواعد التحديث لكل ميزة (الرئيسية، قوائم المنتجات، تفاصيل المنتج، السلة، الطلبات). ضمان تطبيق سياسة Stale-While-Revalidate بشكل متسق.

**- جدول روابط عميقة وإشعارات**إعداد جدول يوضح الربط بين كل Deep Link أو نوع إشعار والشاشة المناسبة مع المعلمات المطلوبة. تحديد السلوك عند فقدان/عدم صحة المعلمات.

**- ميزانيات أداء موسّعة**إضافة ميزانيات دقيقة تشمل:

* زمن الوصول للتفاعل (TTI) في الشاشة الرئيسية
* زمن تحميل أول صورة في قوائم المنتجات
* الحد الأقصى لأحجام الصور
* زمن استجابة أولي (TTFB) للـ APIs الحرجة

**- رفع التغطية الاختبارية ≥ 80% للوحدات الحرجة**زيادة تغطية الاختبارات تدريجيًا خاصة في طبقة Domain وRepositories إلى 80% أو أكثر. إضافة Golden Tests للمكوّنات الأساسية في الواجهة.

**- مراقبة ما بعد الإطلاق**تحديد عتبات تنبيه (Alerts) واتفاقيات خدمة (SLAs) مثل: زمن بدء التشغيل، معدل الأخطاء، معدل ANR، معدلات التحويل في الـCheckout. ربطها بلوحات مراقبة واضحة.

**- توحيد بروتوكول الترقيم (Pagination)**اعتماد صيغة واحدة (page/limit أو cursor) في جميع الـEndpoints لمنع التضارب.

**- تحسين الوصولية (Accessibility)**فرض قائمة تحقق للوصولية: دعم TalkBack/VoiceOver، الحد الأدنى لمساحة اللمس 48dp، توثيق نسب تباين الألوان.

**- حماية إضافية عبر Play Integrity / DeviceCheck**تفعيل آليات التحقق على مستوى المنصات للوصولات الحرجة لحماية الـAPIs من إساءة الاستخدام (اختياري لكنه مفيد).

**- تحسين CI وبناء iOS**إضافة Job مخصص لبناء iOS على macOS ضمن GitHub Actions. استخدام Caching للمكتبات لتقليل وقت التنفيذ.

**- تفصيلية (CODEOWNERS)**تخصيص مراجعين إلزاميين (Code Owners) للمسارات الحساسة (network، auth، checkout) لضمان مراجعة دقيقة قبل الدمج.

NOTE

**1. الحواجز البنيوية (Architectural Guards)**

\* يمنع الاستيراد المباشر بين Features.

\* التواصل فقط عبر Domain Interfaces.

\* التحقق الآلي عبر أدوات lint/CI: أي استيراد مخالف يفشل البناء.

**2. معايير Checkout / COD**

\* استخدام Idempotency-Key لكل طلب شراء.

\* Queue للطلبات عند انقطاع الشبكة لمنع التكرار.

\* معالجة فشل التسليم (رسائل عربية موحدة + إعادة المحاولة).

\* Analytics: حدث `add\_payment\_info(payment\_type=cod)` + حدث `cod\_failed\_reason` لتسجيل أسباب فشل الدفع.

**3. Budgets الأداء (Performance Budgets)**

\* زمن تشغيل أولي (Cold Start) ≤ 2.5 ثانية.

\* معدل Jank < 1% في صفحات (PLP/PDP).

\* حجم ملف AAB ≤ 40MB.

\* CI يحتوي مهمة فحص الحجم (`--analyze-size`) ويمنع الدمج عند التجاوز.

**4. Golden RTL Gates**

\* لقطات ذهبية إلزامية لكل مكون أساسي باللغتين (RTL/LTR).

\* تُختبر مع تكبير خط 1.3× و 1.6×.

\* شرط أساسي قبل الدمج في PR template.

**5. Observability ومعايير التشغيل**

\* معدل Crash-free ≥ 99.5%.

\* معدل فشل API < 2%.

\* لوحة متابعة أسبوعية لمسار الشراء (عرض → سلة → دفع → شراء).

**6. تعزيز الأمن (Security)**

\* تفعيل SSL Pinning للاتصالات.

\* استثناء البيانات الحساسة من النسخ الاحتياطي.

\* تنظيف شامل للـ Cache والبيانات الحساسة عند تسجيل الخروج.

**7. Attribution (First/Last Touch)**

\* تخزين خصائص المستخدم: `first\_source/medium/campaign` و `last\_source/medium/campaign`.

\* إرسال القيم مع حدث `purchase` لتقييم فعالية الحملات التسويقية بدقة.

إرشادات عند استخدام المكاتب (Libraries/Packages)

# **إرشادات عند استخدام المكاتب (Libraries/Packages)**

## **1. اختيار المكتبات**

* **الاعتماد على المكاتب الرسمية والمجتمعية الموثوقة فقط** (مثل: flutter\_bloc، riverpod، dio، go\_router).
* **تجنّب المكاتب الصغيرة أو غير المحدثة** (آخر تحديث قبل أكثر من 12 شهر = خطر أمني/توافق).
* تحقق من **عدد التنزيلات + معدل الصيانة** في pub.dev قبل الإضافة.
* **لا تُستخدم مكتبات خارجية** إذا كان يمكن إنجاز الوظيفة بكود بسيط داخليًا (لتقليل الاعتمادية).

## **2. إدارة الإصدارات**

* جميع المكاتب يجب أن تُثبت بنسخة محددة (pinning) داخل pubspec.yaml وليس باستخدام any.
* يتم **مراجعة التوافق** مع Flutter SDK قبل الترقية.
* لا يُسمح بتحديث مكتبة في فرع التطوير إلا عبر **PR مستقل** + مراجعة.

## **3. المعمارية والتقسيم**

* **لا استدعاء مباشر من الـUI لأي مكتبة خارجية**.
  + الشبكات: عبر طبقة Data فقط (مثال: dio داخل api\_service).
  + التخزين: عبر Repository/Cache layer (مثال: shared\_preferences أو hive).
* **تجريد (abstraction)**: كل مكتبة تُغلف داخل Service أو Helper class.
  + مثال: لا يُكتب FirebaseAnalytics.instance.logEvent في الكود مباشرة، بل داخل AnalyticsService.logEvent().

## **4. التوثيق والالتزام**

* عند إضافة مكتبة جديدة:
  + تحديث ملف ARCHITECTURE.md بذكر اسم المكتبة + الغرض منها + مكان استخدامها.
  + إضافة مثال صغير (usage snippet) داخل دليل المطورين.
* أي مكتبة لا تُوثّق لا يُسمح باستخدامها.

## **5. الأداء والوزن**

* يُمنع إضافة مكتبة تزيد حجم التطبيق بشكل كبير (خصوصًا مكتبات UI ضخمة).
* قبل اعتماد مكتبة، يجب اختبار أثرها على حجم الـAAB باستخدام flutter build appbundle --analyze-size.
* إذا كانت مكتبة توفر أكثر من وظيفة، ولا تحتاج إلا جزءًا منها → ابحث عن مكتبة أصغر أو اعمل implementation داخلي.

## **6. الأمان والخصوصية**

* لا استخدام مكتبات غير موثوقة تتعامل مع **البيانات الحساسة** (توكنات، معلومات مستخدم، بيانات دفع).
* عند استخدام Firebase/Analytics/Crashlytics → يجب تفعيل **إخفاء بيانات PII** (مثل البريد والهاتف) أو تخزينها بشكل مشفر.
* مكتبات التواصل مع السيرفر يجب أن تعمل فقط عبر **HTTPS + TLS** (ويُفضل SSL Pinning).

## **7. الصيانة والاختبارات**

* كل مكتبة جديدة يجب أن تكون **مغطاة باختبارات** أو Mock (مثلاً mock لـ Dio في اختبار الشبكات).
* لا يُسمح بربط المكتبة مباشرة في الاختبارات → استخدم واجهة مجردة (interface).

# **توجية**

* **المسموح**:
  + dio (شبكات) → يُستخدم داخل network\_service.dart فقط.
  + riverpod (state management) → عبر Providers، بدون خلط داخل Widgets.
  + go\_router (navigation) → داخل ملف Routing المركزي فقط.
* **الممنوع**:
  + استخدام http + dio معًا.
  + مناداة Firebase مباشرة من Widgets.
  + إدخال مكتبة لغرض تجريبي بدون توثيق.

نقاط إضافية

## **نقاط إضافية**

### **1. إدارة الأكواد (Code Style & Structure)**

* الالتزام بـ **Dart style guide** + استخدام Lints موحدة (pedantic أو flutter\_lints).
* ممنوع ترك TODO/print في الكود عند الدمج.
* أي كود تجريبي يجب أن يكون في فرع خاص أو ملف sandbox/.

### **2. التسمية (Naming Conventions)**

* أسماء الملفات: snake\_case.dart.
* أسماء الـClasses: PascalCase.
* أسماء المتغيرات: camelCase.
* يضاف **Suffix واضح** (مثال: HomeScreen, UserRepository, CartService).

### **3. الصور والموارد (Assets)**

* جميع الصور داخل assets/images/، والأيقونات داخل assets/icons/.
* لا يُستخدم رابط URL مباشر لصورة في الكود.
* يجب ضغط الصور (WebP أو SVG) لتقليل الحجم.

### **4. الـLocalization (الترجمة)**

* استخدام **arb**/intl للترجمات.
* يمنع كتابة النصوص مباشرة داخل Widgets.
* التأكد من دعم RTL و LTR دائمًا.

### **5. الأمن (Security Practices)**

* تخزين التوكنات/المفاتيح في Secure Storage فقط.
* عدم إدخال أي مفتاح API في الكود (يُستخدم .env أو Secret Manager).
* عند التعامل مع الطلبات، يجب وجود Timeout + Retry policy.

### **6. الـTesting (الاختبارات)**

* **Unit tests** لكل use-case و Repository.
* **Widget tests** للشاشات المهمة (Login, Checkout, Product Details).
* **Golden tests** لكل UI Component أساسي (مع RTL).

### **7. المراجعة والتوثيق (Reviews & Docs)**

* كل PR يجب أن يحتوي:
  + وصف للتغييرات.
  + صور/فيديو قصير إذا التغيير واجهة.
  + نتائج اختبارات (نجاح CI).
* تحديث ARCHITECTURE.md و CHANGELOG.md عند أي إضافة مهمة.

### **8. الأداء (Performance)**

* يمنع استخدام Widgets كثيفة بدون Virtualization (مثل ListView بدون builder).
* مراقبة إعادة البناء (rebuilds) عبر Flutter DevTools.
* استخدام const widgets حيثما أمكن.

### **9. الـAnalytics**

* توحيد جميع الأحداث (Events) عبر AnalyticsService.
* ممنوع تسجيل الأحداث مباشرة من الـUI.
* التأكد من وجود **Funnel كامل**: عرض منتج → إضافة للسلة → Checkout → Purchase → COD Confirm.

### **10. الـError Handling**

* ممنوع رمي Exception خام.
* استخدام Result/Failure objects.
* كل خطأ يجب أن يكون له رسالة موحدة للعميل + Log داخلي.

Codemagic CI/CD

# **Codemagic CI/CD – Flutter**

## **1) الهدف (Objectives)**

* أتمتة البناء والاختبارات والتوزيع لإصدارات **Android** و **iOS**.
* تقليل الأعطال ووقت الإطلاق عبر **بوابات قبول** واضحة (Release Gates).
* مسارات موحّدة: develop → staging → production مع تتبّع تغييرات واضح.

## **2) المتطلبات (Prereqs)**

* مستودع Git (GitHub/GitLab/Bitbucket) بفروع: develop, staging, main.
* **Flavors** في Flutter/Gradle/Xcode: dev, staging, prod.
  + Android applicationId:
    - com.zahraah.app.dev
    - com.zahraah.app.staging
    - com.zahraah.app
  + iOS **Bundle IDs**:
    - com.zahraah.app.dev
    - com.zahraah.app.staging
    - com.zahraah.app
* حسابات المتاجر: **Google Play Console** + **App Store Connect** (شركة).
* تفعيل **Crashlytics** و(اختياري لكن مفيد) **Firebase App Distribution**.

## **3) الأسرار ومتغيرات البيئة (Secrets / ENV)**

تُحفظ في Codemagic ضمن **Groups** (لا تُرفع إلى الريبو).

**android\_keystore**

* ANDROID\_KEYSTORE (Base64 لملف *.jks* أو ارفعه من Files)
* ANDROID\_KEYSTORE\_PASSWORD
* ANDROID\_KEY\_ALIAS
* ANDROID\_KEY\_ALIAS\_PASSWORD

**ios\_connect\_api** (App Store Connect API)

* ASC\_ISSUER\_ID
* ASC\_KEY\_ID
* ASC\_PRIVATE\_KEY (النص الخاص للمفتاح)

**play\_console** (Google Play Service Account)

* PLAY\_SERVICE\_ACCOUNT\_JSON (محتوى JSON لمستخدم الخدمة بصلاحية Release Manager)

**firebase\_dist** (Firebase App Distribution) – اختياري

* FIREBASE\_TOKEN
* FIREBASE\_ANDROID\_DEV\_APP\_ID / FIREBASE\_IOS\_DEV\_APP\_ID

**notifiers** (التنبيهات)

* SLACK\_WEBHOOK\_URL (اختياري)

## **4) سير العمل (Workflows)**

### **develop (داخلي – QA)**

* Trigger: أي Push/PR إلى develop.
* خطوات: flutter analyze + flutter test + بناء APK (debug أو release).
* توزيع: Firebase App Distribution لفريق QA (اختياري).

### **staging (اختبار متجر داخلي)**

* Trigger: دمج إلى staging أو Tag باسم staging-\*.
* بناء: **AAB** للأندرويد + **IPA** للـiOS (Release).
* نشر: Google Play **Internal track** + TestFlight تلقائي.

### **production (إطلاق)**

* Trigger: إنشاء **Tag يبدأ بـ v** (مثل v1.3.0).
* نشر: Google Play **Production** (طرح تدريجي 10%) + App Store (Submit to App Store، إطلاق يدوي).

## **5) بوابات القبول (Release Gates)**

* flutter analyze بدون أخطاء.
* نجاح flutter test --coverage (هدف أولي ≥ 20%).
* عدم وجود أعطال حرجة في **Crashlytics** للإصدار المراد إطلاقه.
* فحص أداء يدوي: TTI < 3s للشاشات الأساسية على أجهزة متوسطة.

## **6) تنبيهات (Alerts)**

* عند فشل البناء/النشر: إشعار Slack عبر SLACK\_WEBHOOK\_URL.
* عند نجاح إطلاق الإنتاج: إشعار بالنسخة والرابط.

## **7) أفضل ممارسات**

* **Conventional Commits** لتوليد changelog آلي.
* تفعيل **Gradle/CocoaPods cache** لتسريع البناء.
* **Staged Rollout** للأندرويد: 10% → 50% → 100%.
* أسرار **Secrets** دائمًا خارج الكود.

## **8) ملف جاهز: codemagic.yaml**

عدّل المسارات والمعرّفات حسب مشروع **Zahraah**.

workflows:

develop:

name: Dev – Internal QA

max\_build\_duration: 60

environment:

flutter: stable

vars:

APP\_FLAVOR: dev

groups:

- android\_keystore

- firebase\_dist

- notifiers

scripts:

- flutter pub get

- flutter analyze

- flutter test --coverage

- flutter build apk --flavor $APP\_FLAVOR --debug

# - |

# firebase appdistribution:distribute build/app/outputs/flutter-apk/app-dev-debug.apk \

# --app $FIREBASE\_ANDROID\_DEV\_APP\_ID \

# --groups "qa"

artifacts:

- build/app/outputs/\*\*/\*.apk

publishing:

scripts:

- |

if [ -n "$SLACK\_WEBHOOK\_URL" ]; then

curl -X POST -H 'Content-type: application/json' \

--data '{"text":"✅ Dev build finished: '$CM\_COMMIT'"}' $SLACK\_WEBHOOK\_URL

fi

staging:

name: Staging – Stores (Internal)

max\_build\_duration: 90

environment:

flutter: stable

vars:

APP\_FLAVOR: staging

groups:

- android\_keystore

- ios\_connect\_api

- play\_console

- notifiers

scripts:

- flutter pub get

- flutter analyze

- flutter test --coverage

- flutter build appbundle --flavor $APP\_FLAVOR --release

- flutter build ipa --flavor $APP\_FLAVOR --release

artifacts:

- build/app/outputs/\*\*/\*.aab

- build/ios/ipa/\*.ipa

publishing:

google\_play:

credentials: $PLAY\_SERVICE\_ACCOUNT\_JSON

track: internal

app\_store\_connect:

issuer\_id: $ASC\_ISSUER\_ID

key\_identifier: $ASC\_KEY\_ID

private\_key: $ASC\_PRIVATE\_KEY

submit\_to\_testflight: true

scripts:

- |

if [ -n "$SLACK\_WEBHOOK\_URL" ]; then

curl -X POST -H 'Content-type: application/json' \

--data '{"text":"🚀 Staging uploaded to Internal/TestFlight for '$CM\_BRANCH'"}' $SLACK\_WEBHOOK\_URL

fi

production:

name: Production – Stores

max\_build\_duration: 120

environment:

flutter: stable

vars:

APP\_FLAVOR: prod

groups:

- android\_keystore

- ios\_connect\_api

- play\_console

- notifiers

triggering:

events:

- tag

branch\_patterns:

- pattern: 'v\*'

include: true

source: true

scripts:

- flutter pub get

- flutter analyze

- flutter test --coverage

- flutter build appbundle --flavor $APP\_FLAVOR --release

- flutter build ipa --flavor $APP\_FLAVOR --release

artifacts:

- build/app/outputs/\*\*/\*.aab

- build/ios/ipa/\*.ipa

publishing:

google\_play:

credentials: $PLAY\_SERVICE\_ACCOUNT\_JSON

track: production

rollout:

fraction: 0.1

app\_store\_connect:

issuer\_id: $ASC\_ISSUER\_ID

key\_identifier: $ASC\_KEY\_ID

private\_key: $ASC\_PRIVATE\_KEY

submit\_to\_app\_store: true

submit\_to\_testflight: false

scripts:

- |

if [ -n "$SLACK\_WEBHOOK\_URL" ]; then

curl -X POST -H 'Content-type: application/json' \

--data '{"text":"🎉 Production released tag '$CM\_TAG'"}' $SLACK\_WEBHOOK\_URL

fi

## **9) سكربت (اختياري) لزيادة buildNumber تلقائيًا**

أضِفه في بداية قسم scripts: قبل البناء.

#!/usr/bin/env bash

set -e

CURRENT=$(grep '^version:' pubspec.yaml | awk '{print $2}')

BASE=${CURRENT%+\*}

BUILD=${CURRENT#\*+}

if [[ "$BUILD" == "$CURRENT" ]]; then BUILD=0; fi

NEXT=$((BUILD+1))

sed -i.bak "s/^version: .\*/version: $BASE+$NEXT/" pubspec.yaml

head -n 5 pubspec.yaml

## **10) قائمة ضبط سريعة (Checklist)**

1. اربط مستودع Git في Codemagic وفَعّل **Builds on push**.
2. أنشئ **Groups** وأضف الأسرار المذكورة.
3. تأكد من ضبط **Flavors** في Gradle/Xcode (IDs تتطابق مع المتاجر).
4. فعّل **Gradle/CocoaPods cache** من إعدادات Workflow.
5. جرّب develop → staging → اصنع Tag v1.0.0 لاختبار **production**.

## **11) مشاكل شائعة**

* **توقيع iOS يفشل**: تأكد من صلاحيات App Store Connect والقيم الصحيحة لـ ASC\_\*.
* **رفض Google Play**: زِد **Version Code** دائمًا؛ أو استخدم سكربت الرفع التلقائي.
* **عدم تطابق Bundle IDs/ applicationId** بين Flavors والمتاجر.
* **حجم IPA/AAB كبير**: فعّل --split-debug-info و --obfuscate (مع حفظ رموز التفكيك).

## **1) حدّ أدنى إلزامي “داخل” مستند بنية التطبيق (يبدأ من اليوم الأول)**

هذا جزء من المعايير الأساسية، لأنه يضمن التزام الكود بالمعايير تلقائيًا:

* **تشغيل تلقائي على كل PR**: analyze + format check + **Unit tests** + **تغطية ≥ 70%**.
* **بناء Debug** (Android/iOS) للتأكد من أن المشروع قابل للبناء دومًا.
* **بوابات دمج (Gates)**: فشل أي خطوة يمنع الدمج إلى main.
* **حماية الفروع** + مراجعتان للكود + CODEOWNERS.
* **إدارة أسرار آمنة** (API keys, keystore, signing certs) عبر متغيرات Codemagic.
* **Caching** لحزم Flutter/gradle لتقليل وقت البناء.
* **إشعارات فشل** للـSlack/البريد.

هذه البنود تُوثّق “سياسة CI” في ملف البِنية، وتُترجم عمليًا إلى ملف codemagic.yaml داخل المستودع.

## **2) توسيع لاحق كمرحلة إصدار (CD) — “ملف منفصل/ملحق عملي”**

هذا يُنفّذ بعد استقرار الأساس، وغالبًا نضع تفاصيله في **ملحق CI/CD** أو README مستقل:

* **توقيع تلقائي** وبناء **Release** لمنصّتَي Android & iOS.
* **توزيع مرحلي**: Play Console (Internal → Closed) + TestFlight، بنسبة 10% → 50% → 100%.
* **رفع الخرائط**: dSYM/Proguard لـ Crashlytics/Sentry.
* **رفع التغطية والتقارير**، وتوليد **Changelog** من الـtags/PRs.
* **مصفوفة بيئات** (dev/stage/prod) + تبديل مفاتيح --dart-define.
* **اختبارات تكامل/قبول** تلقائية قبل النشر، مع **Release QA Gates** (مثلاً نجاح Checkout ≥ 95%).
* **رجوع سريع** (Rollback): تعطيل ميزة عبر Feature Flag أو إرجاع الإصدار السابق.

### **الخلاصة**

* **له علاقة مباشرة بالبنية الأساسية**: الجزء “الحد الأدنى” من Codemagic يجب أن يُوثّق ويُفعّل من اليوم الأول كجزء من مستند البنية (CI كحارس جودة).
* **والتفاصيل التشغيلية العميقة** (التوقيع، الرفع للمتاجر، التوزيع المرحلي…) تُعامل كـ **مرحلة ثانية** داخل **ملحق CI/CD** أو في codemagic.yaml.

الإضافات

# **الإضافات**

## **1. المنصّات (Android / iOS)**

* **Android**
  + Notification permission (Android 13+)، إعداد قنوات إشعارات.
  + Backup rules، Network Security config، App Links.
  + targetSdk / minSdk وحدود حجم الـ AAB.
* **iOS**
  + Push capabilities (APS)، Background modes (الضرورية فقط).
  + App Transport Security (ATS) وسياسة الشبكة.
  + Privacy Manifest + App Tracking Transparency (ATT).
  + Universal Links وإعدادات الـ App Store.

## **2. إدارة الأسرار والأمان**

* إدارة الأسرار عبر GitHub Encrypted Secrets / CodeMagic variables.
* حظر التسرّب عبر secret-scanner step في CI.
* google-services.json / .plist لكل Flavor + حقن تلقائي في CI.
* سياسة تدوير المفاتيح (API Keys).
* دعم TLS Pinning للاتصالات الحرجة (اختياري).

## **3. التحليلات والمراقبة (Observability)**

* إلزام Crashlytics + Firebase Performance Monitoring.
* ربط الإصدار (release/channel/buildNumber) بالتحليلات.
* لوحة مؤشرات إلزامية:
  + Crash-free ≥ 99.5%
  + ANR < X%
  + Error Rate API < 2%
* ملف **analytics\_events.md** يدوّن جميع الأحداث والمعاملات ويُعتبر جزءًا من Definition of Done.

## **4. الجودة والأداء داخل CI/CD**

* تفعيل flutter build appbundle --analyze-size ورفع التقرير كـ artifact.
* فشل الـPR إذا زاد حجم الـAPK/AAB بنسبة > N%.
* Device Matrix لاختبارات Integration (شاشات/نسخ Android مختلفة).
* دعم اختبار iOS على macOS runner (عند توفره).
* تفعيل pre-commit hooks: (dart format, analyze).
* دليل Code Review: يحدد متى نرفض PR ومتى نطلب Goldens إضافية.

## **5. النشر عبر CodeMagic (CD Pipeline)**

* بيئات متعددة (dev/stage/prod).
* Build + Test + Sign + Deploy تلقائي إلى:
  + Firebase App Distribution
  + TestFlight
  + Play Console
* استخدام بيئات ومتغيرات آمنة (Environment Variables).
* تقارير ما بعد البناء: حجم التطبيق، اختبارات الأداء.

## **6. الترجمة والدعم العربي (RTL)**

* استخدام ملفات .arb موحّدة (snake\_case).
* دعم قواعد plural/select.
* إلزام Goldens RTL + تكبير خط 1.3× / 1.6× في CI.
* شرط فحص تلقائي للـRTL والـLocalization ضمن الـPR.

## **7. الشبكة والكاش**

* توحيد عقود Pagination / Sorting / Filtering.
* Idempotency keys للعمليات الحرجة (checkout).
* سياسة Offline Queue: ما يُخزّن محليًا ويُعاد إرساله عند عودة الإنترنت.

## **8. Definition of Done (موسع)**

أي Feature جديدة تعتبر “مكتملة” فقط إذا شملت:

1. الأحداث (Events) مربوطة ومُختبرة في DebugView.
2. Crashlytics يلتقط userId / session.
3. تقرير حجم التطبيق مرفق.
4. اختبار RTL + Golden مرفق.
5. استيفاء Checklist أصول المتجر (Store Assets Checklist).

ملاحظات متعددة

1. جدول قيم الكاش ومدة الـTTL لكل شاشة ولكل Endpoint (مع سياسة SWR بالأرقام).
2. خريطة الروابط العميقة Deep/Universal Links (التحقق من النطاقات + fallback + سيناريوهات الاختبار).
3. قنوات/فئات الإشعارات وسياسات الإرسال (TTL, collapse\_key, quiet hours, opt-in/opt-out).
4. خريطة أخطاء Backend→UI موحّدة (أكواد، رسالة عربية، إجراء المستخدم، مستوى الشدة).
5. ميزانيات أداء إلزامية (App start/Screen load/FPS/Memory/App size وحدود الصور).
6. سياسة الصور والأصول (صيَغ، أبعاد قصوى، الضغط، CDN، lazy-loading، placeholders).
7. أمان متقدّم داخل التطبيق (SSL pinning، كشف root/jailbreak، حجب لقطات على الشاشات الحساسة).
8. سياسة السجلات Logging (مستويات، إطفاء بالإنتاج، إزالة البيانات الحساسة، correlation IDs، الاحتفاظ).
9. Runbooks تشغيلية للحوادث (أعطال/شبكة/بطء/فشل الدفع) مع عتبات تنبيه وإجراءات فورية.
10. خطّة CI/CD لـ iOS (macOS runner، التوقيع، أرقام البناء، TestFlight، توزيع مرحلي).
11. مصفوفة المنصات والأذونات (min/target SDK، iOS min، Privacy Manifest/Play Data Safety).
12. التحليلات والهوية: حفظ **أول وآخر مصدر** (UTM) + ربط user\_id + مخطط أحداث موحّد.
13. Feature Flags/Remote Config Playbook (قائمة الميزات الحرجة، التفعيل/الإرجاع، حراسة المخاطر).
14. إدارة الأسرار Secrets (dart-define/CI secrets، التناوب Rotation، عدم التسريب في السجلات).
15. مرونة الشبكة (Retry/Exponential Backoff، Circuit Breaker، مصفوفة Timeouts بحسب العملية).
16. بوابات جودة الإصدار Release QA Gates (checklist، golden thresholds، smoke/automation، staged rollout
17. سياسة التبعيات الخارجية (تثبيت النسخ، تحديثات دورية، فحص أمن/ترخيص، بدائل عند الفشل).
18. سياسة البيانات المحلية (ما يُخزَّن، تشفير، مدة الاحتفاظ، إستراتيجية التنظيف/الترحيل).
19. الوصولية والـRTL الموسّعة (تباين، أحجام خط، TalkBack/VoiceOver، اختبارات pseudo-localization).
20. الامتثال والخصوصية داخل التطبيق (نصوص الموافقات Push/SMS/WhatsApp/Email، سياسة الخصوصية داخل التطبيق).
21. KPIs مستهدفة للواجهة (CTR البنرات، Add-to-Cart، Conversion، Checkout success، Retention) مع حدود تنبيه.
22. معايير التسمية Naming Conventions موحّدة (ملفات/مجلدات/Providers/UseCases/DTOs).
23. مصفوفة إعدادات البيئات (dev/stage/prod) في جدول واحد (baseUrls، featureFlags، logLevel…).
24. إدارة الإصدارات Versioning وChangelog إلزامي.
25. إستراتيجية Offline/Sync (حل تضارُب السلة والطلبات Last-write-wins أو دمج مُعرّف).
26. سياسة الشبكات الضعيفة (Prefetch، ضغط Gzip/Brotli، قيود الصور، fallbacks بلا إنترنت).
27. اختبار الأمان الدوري (قائمة MSTG-lite، نطاق Pen-Test، معالجة المخرجات).
28. خطط ترحيل/تراجع للبيانات (migrations للـLocal/Schema + خطة rollback مراقَبة).

# **ملحقات معيارية**

## **1) جدول الكاش وTTL (SWR)**

| **الشاشة/الخدمة** | **TTL** | **سياسة التحديث** |
| --- | --- | --- |
| Home/Banners | 10 دقائق | SWR فور فتح التطبيق |
| قائمة الأقسام | 24 ساعة | تحديث بالخلفية يوميًا |
| PLP (قائمة المنتجات) | 15 دقيقة | تحديث عند التمرير للأعلى |
| PDP (تفاصيل المنتج) | 60 دقيقة | تحديث عند الرجوع أو تغيير المقاس |
| البحث (اقتراحات) | 60 دقيقة | تحديث عند تغيّر المصطلح |
| السلة | لا كاش | حالة محلية فقط |
| الطلبات | 5 دقائق | تحديث عند الفتح |

## **2) الروابط العميقة (Deep/Universal Links)**

* النطاقات المعتمدة: app.zahraah.com (Android App Links + iOS Universal Links).
* خريطة مختصرة:
  + /p/:id → PDP
  + /c/:slug → PLP
  + /cart → السلة
  + /o/:id → تفاصيل الطلب
* حالات الاختبار: مُثبّت/غير مُثبّت، مُسجّل/غير مُسجّل.

## **3) الإشعارات (سياسات الإرسال)**

* قنوات: promotions, orders, system.
* TTL الرسالة: 4 ساعات؛ collapse\_key لكل قناة؛ **ساعات هادئة** 22:00–08:00.
* Foreground: In-App Banner مع Deep-Link صحيح.

## **4) خريطة أخطاء Backend→UI**

| **كود** | **الرسالة العربية** | **إجراء المستخدم** | **شدة** |
| --- | --- | --- | --- |
| NET\_TIMEOUT | الاتصال بطيء… | إعادة المحاولة | متوسطة |
| OOS | نفدت الكمية | اقتراح بدائل | منخفضة |
| AUTH\_EXPIRED | انتهت الجلسة | تسجيل الدخول من جديد | عالية |
| PAY\_FAIL | فش | اختيار وسيلة أخرى | عالية |

## **5) ميزانيات الأداء (حدود إلزامية)**

* **App start P95 ≤ 2.5s**، **Screen load P95 ≤ 1.2s** (الشاشات الحرجة).
* حجم التطبيق ≤ **45MB** (Android APK/IPA)، الصورة المفردة ≤ **180KB** (Hero)، **30KB** (Thumb).
* Crash-free users ≥ **99.5%**.

## **6) سياسة الصور والأصول**

* الصيغة: **WebP** (أساس)؛ fallback JPEG عند الحاجة.
* أبعاد قصوى: Product Hero 1080×1350، Thumb 320×320.
* عبر CDN مع ضغط تلقائي، Lazy-Loading وPlaceholders.

## **7) أمان متقدّم**

* **SSL Pinning** (تجديد سنوي)، حظر لقطات الشاشة في OTP/الدفع.
* كشف Root/Jailbreak مع مسار إيقاف آمن.

## **8) السجلات Logging**

* المستويات: debug/info/warn/error.
* إخفاء أي PII/رموز؛ **retention**: Dev 7 أيام، Prod أخطاء فقط.

## **9) Runbooks حوادث (مختصر)**

* أعطال: تحقق Crashlytics → عزل الإصدار → Hotfix.
* شبكة: تفقد Status/Backoff → بدائل/رسالة للمستخدم.
* بطء: مراجعة Performance Traces → تحسين الصور/الاستعلامات.
* دفع: فحص مزوّد/إعادة محاولة/تعطيل المسار مؤقّتًا.

## **10) CI/CD لنظام iOS**

* macOS runner + توقيع أوتوماتيكي + TestFlight.
* **Staged Rollout**: 10% → 50% → 100% خلال 48 ساعة.

## **11) مصفوفة المنصات والأذونات**

* Android: minSdk 24 / target 34، iOS: min iOS 13+.
* Privacy Manifest/Play Data Safety محدثان.

## **12) التحليلات والهوية**

* حفظ **أول وآخر UTM** محليًا وربطهما بـ user\_id في GA4.
* مخطط أحداث موحّد للقمع الشرائي + خصائص الجهاز/الإصدار.

## **13) Feature Flags / Remote Config**

* قائمة ميزات حرجة تحت أعلام (Checkout Guard, Free-Shipping Banner…).
* **Rollback فوري** عبر العلم + إفراغ كاش الميزة.

## **14) إدارة الأسرار**

* جميع المفاتيح عبر CI Secrets و--dart-define؛ **دوران كل 90 يومًا**.

## **15) مرونة الشبكة**

* Timeouts: اتصال 5s، قراءة 10s، إجمالي 30s.
* Retry 3 مرات + **Exponential Backoff**؛ **Circuit Breaker** بعد 5 فشل/60ث.

## **16) بوابات جودة الإصدار (Release Gates)**

* نجاح Checkout ≥ **95%**، نسبة الأخطاء ≤ **1%** من الجلسات.
* اختبارات تلقائية خضراء + Golden Tests للشاشات الأساسية.

## **17) سياسة التبعيات**

* قفل نسخ (lockfile)، تدقيق أمني وترخيص رُبع سنوي، إزالة المهجور.

## **18) سياسة البيانات المحلية**

* ما يُخزّن: جلسة/سلة/تفضيلات فقط. **تشفير** + تنظيف دوري (30 يومًا).

## **19) الوصوليّة والـRTL**

* تباين ≥ 4.5:1، دعم تكبير الخط، اختبار TalkBack/VoiceOver.

## **20) الموافقات والامتثال**

* مفاتيح Opt-in/Opt-out لـ Push/SMS/WhatsApp/Email داخل “الإعدادات”.
* سياسة خصوصية داخل التطبيق مرتبطة بكل نقطة جمع بيانات.

## **21) مؤشرات الواجهة (KPIs)**

* CTR البنرات ≥ **3%**، Add-to-Cart ≥ **8%**، Conversion ≥ **1.5%**.
* تنبيهات عند الهبوط 20% عن الهدف لأسبوعين.

## **22) التسمية Naming**

* المجلدات: feature\_name/…، الملفات camelCase، الأصناف PascalCase،
* اللاحقات: …Repository, …UseCase, …Dto, …Notifier.

## **23) إعدادات البيئات (dev/stage/prod)**

* جدول واحد: baseUrl, logLevel, featureFlags, analyticsId.

## **24) الإصدارات وChangelog**

* SemVer: MAJOR.MINOR.PATCH + **CHANGELOG.md** إلزامي مع كل إصدار.

## **25) Offline/Sync**

* السلة **محلية دائمة** + دمج حسب (SKU+المقاس+اللون) عند المزامنة.
* الطلبات: **Server-wins** دائمًا.

## **26) سياسة الشبكات الضعيفة**

* Prefetch للصور الحرجة، Gzip/Brotli، خفض جودة الصور على 3G،
* إرجاء طلبات الخلفية غير الحرجة.

## **27) اختبار أمني دوري**

* MSTG-lite شهريًا + **Pen-Test نصف سنوي** مع تتبّع Findings.

## **28) ترحيل/تراجع للبيانات**

* إصدارات للمخططات المحلية + **Migration scripts** + خطة Rollback.

حزمة تشغيلية

1. **عقد الـAPI الموحّد (OpenAPI/Swagger + Mock Server)**يثبّت كل Endpoint/Request/Response/Errors كمرجع واحد للعميل والخلفية.
2. **قاموس البيانات (Data Dictionary)**تعريف حقول الكيانات الأساسية (User, Product, Order, Cart…) مع القيود والتحقّقات وأنواع البيانات.
3. **مخطط التحليلات الموحّد (Event Schema v1)**أسماء الأحداث، المعاملات وأنواعها، ربط user\_id، ومنطق حفظ **أول/آخر UTM**.
4. **حزمة الاختبار والوصول**حسابات اختبار (dev/stage)، بيانات وهمية Seed، بطاقات دفع تجريبية، وأذونات الوصول للأدوات.
5. **تعريفات DoR/DoD + Checklists للكود وPR وCI**ما هو جاهز لبدء التطوير وما هو “مكتمل”، مع قواعد Git/PR والمراجعة والـlint.
6. **خريطة الملكية CODEOWNERS**من يملك كل ميزة/مجلد ومن يراجعها (مسؤولية واضحة، سرعة في القرارات).
7. **دليل الإصدار والتراجع**Release checklist لمتاجر iOS/Android، التوزيع المرحلي، خطوات Rollback، وبدائل Feature Flags.
8. **SLO/تنبيهات + أمان مختصر**SLOs (Crash-free ≥99.5%, App start P95 ≤2.5s…) + مصفوفة التنبيهات (من يتنبه ومتى) + فحص أمني MSTG-lite وجدول دوران الأسرار.

Redis Standards

# **Redis Standards**

## **0) النطاق**

* **الغرض:** تسريع القراءة، الحماية (rate limiting)، الأقفال الموزعة، والجلسات الخفيفة.
* **ليس مصدر حقيقة:** لا تُخزَّن بيانات الأعمال الحساسة كمصدر نهائي.

## **1) التوزيع والتشغيل (Topology)**

* بيئات منفصلة: dev, staging, prod.
* مثيلان (مستحسن):
  + **cache** (بدون AOF، سياسة إخلاء allkeys-lru).
  + **state** (AOF=on للأمان للجلسات/القيود/الأقفال).
* **التوفّر العالي:**
* Managed (ElastiCache/Redis Cloud/Upstash) أو Self-Hosted مع Sentinel (Master + Replica).

## **2) التسمية المعيارية للمفاتيح (Key Naming)**

{app}:{env}:{domain}:{entity}:{id}:{sub}

مثال: zahraah:prod:product:12345

* قوائم بفلاتر/ترتيب/صفحة ضمن المفتاح: zahraah:prod:category:{cid}:list:p{n}:sort:{s}

## **3) سياسة TTL**

* كل مفتاح **له TTL** مناسب لاستخدامه، مع **Jitter ±10–20%** لتجنب انتهاء جماعي.
* مرجع سريع:
  + المنتج: 5–15 دقيقة
  + قوائم الفئات: 1–5 دقائق
  + الصفحة الرئيسية: 30–60 ثانية
  + قواعد الشحن/التسعير: 5–30 دقيقة
  + الكوبونات: 2–10 دقائق

## **4) نمط Cache-Aside القياسي**

**قراءة:**

1. Get → إذا Hit أعد القيمة.
2. إذا Miss: استعلم MySQL → Set مع TTL → أعد القيمة.

**كتابة:**

* نفّذ المعاملة في MySQL.
* بعد **نجاح commit**: إبطل (DEL) المفاتيح ذات الصلة، وانشر حدث cache:invalidate (Pub/Sub) مع قائمة المفاتيح/الأنماط.
* احمِ من **Cache Stampede** عبر Lock (SET NX + TTL قصير).

## **5) خرائط الإبطال (تُنسَّق مع فريق DB)**

* **product/variant:**
  + DEL zahraah:prod:product:{product\_id}
  + DEL zahraah:prod:category:{cid}:list:\* ذات الصلة
* **category:**
  + DEL zahraah:prod:category:{cid}:list:\*
  + DEL zahraah:prod:home
* **pricing/shipping:**
  + DEL zahraah:prod:shipping:city:\*
* **coupon:**
  + DEL zahraah:prod:coupon:{code}
* **inventory:**
  + DEL zahraah:prod:product:{product\_id} (+ قوائم ذات صلة)

المصدر الموازي موجود في ملف «معايير قاعدة البيانات» — حافظوا على تطابق التسمية.

## **6) القيود والحماية**

* **Rate Limiting (OTP/Login):** مفاتيح نافذة منزلقة (Sliding Window) لكل رقم أو IP.
* **Locks:**
  + عمليات حساسة (كوبونات، إنشاء الطلب) عبر SET key value NX EX {ttl\_s}.
  + إعادة المحاولة مع backoff بسيط عند فشل الحصول على القفل.
* **Sessions/Tokens:** إن لم تكن JWT كاملة، خزن الحد الأدنى بTTL مناسب.

## **7) الإعداد والتهيئة (Config)**

* قراءة الإعداد من متغيرات البيئة: REDIS\_URL, REDIS\_TLS, REDIS\_DB\_CACHE, REDIS\_DB\_STATE.
* مهلة أوامر افتراضية 50–200ms، وإلغاء (timeout) عند التجاوز.
* اقتبس القيم الثنائية/JSON بسلاسل UTF-8.

## **8) المراقبة (Observability)**

* **مؤشرات أساسية:** Hit Ratio (>0.9 هدف للكاش الشائع)، Latency، Evictions، Memory Fragmentation.
* **Slowlog:** مفعّل وتحليل دوري.
* **لوحة مراقبة:** أعلى المفاتيح وصولًا/حجمًا + أخطاء الاتصال.

## **9) الأمن**

* شبكة خاصة فقط (VPC/VNet) + TLS عند العبور بين الشبكات.
* كلمات مرور قوية + ACL، وتعطيل أو تقييد أوامر حساسة (FLUSHALL, CONFIG).

## **10) أمثلة شبه-كود (Pseudocode)**

**قراءة منتج:**

val = redis.get(key)

if (!val) {

val = db.fetchProduct(id)

redis.set(key, val, TTL+Jitter)

}

return val

**تحديث منتج:**

transaction {

db.updateProduct(p)

}

redis.del(productKey)

redis.publish('cache:invalidate', [productKey, relatedCategoryListPattern])

**قفل كوبون:**

if (redis.set(lockKey, reqId, NX, EX=20)) {

// safely redeem

try { redeem(); }

finally { if (redis.get(lockKey)==reqId) redis.del(lockKey) }

} else {

throw BusyError

}

## **11) معايير القبول (App)**

1. التزام التسمية والـTTL والجِتر.
2. مسارات الكتابة تُبطِل المفاتيح بعد commit فقط.
3. حماية Stampede بالقفل.
4. مراقبة Hit Ratio/Evictions/Latency متاحة للفريق.
5. اختبارات تكامل تُثبت الإبطال لقوائم وفرديات المنتج والكوبونات والشحن.

## **12) خطة طوارئ مختصرة (Runbook)**

* ارتفاع evictions: راجع TTL/السعة أو وسّع الذاكرة.
* Hit Ratio منخفض: تأكد من التسمية/التخزين؛ راجع عدم وجود Miss متكرر بسبب Jitter مفرط.
* Latency مرتفع: تحقق من الشبكة، CPU، Slowlog، وحجم القيم.

استخدام الأدوات AI

# **خريطة استخدام الأدوات:**

| **الأداة** | **الهدف** | **أين تعمل** | **متى نستخدمها** | **خطوات مختصرة** | **المخرج** |
| --- | --- | --- | --- | --- | --- |
| **Android Studio + إضافات Flutter/Dart** | بيئة التطوير الأساسية | محلي | طوال التطوير | فتح المشروع، تشغيل/تصحيح | بناء وتشغيل ثابت |
| **GitHub Copilot** | إكمال وتوليد كود | محلي داخل IDE | أثناء الكتابة | قبول/رفض الاقتراحات، توليد دوال/اختبارات | كود أسرع وأقل أخطاء |
| **Copilot Code Review** | مراجعة PR ذكية | GitHub | عند فتح PR | يضيف تعليقات تلقائيًا | ملاحظات قابلة للتنفيذ |
| **Continue.dev** (وكيل داخل IDE) | مهام على مستوى المشروع | محلي | عند refactor/ترقية | وصف المهمة → تنفيذ تغييرات متعددة | تغييرات متناسقة عبر الملفات |
| **lefthook** | منع الكود السيّئ قبل الرفع | محلي (git hooks) | قبل كل commit | يشغّل format/analyze/test تلقائيًا | رفض الكومِت إن فشل فحص |
| **very\_good\_analysis** | قواعد lint صارمة | محلي + CI | دائمًا | يطبّق قواعد Dart/Flutter | نمط كود موحّد |
| **Semgrep** | فحص أمني/جودة ثابت | CI | كل PR | يشغّل قواعد أمان/جودة | تنبيهات مشاكل مبكرًا |
| **Codemagic CI/CD** | فحص وبناء تلقائي | CI | كل PR + عند الإصدار | analyze/test/build (+ تغطية) | Gates تمنع الدمج إن فشل |
| **Renovate** | تحديث تبعيات pubspec | GitHub | أسبوعيًا/عند توفر تحديث | يفتح PR تلقائي | تبعيات محدّثة آمنًا |

* **أي كود لا يمر عبر lefthook وCodemagic وCopilot Code Review لا يُدمَج.**

# **2) عمل مطوّر (Timeline سريع)**

1. **ابدأ:**
   * **git pull → افتح Android Studio.**
   * **شغّل الجهاز/المحاكي.**
   * **استخدم Copilot لكتابة الدوال/الواجهات، وContinue للمهام الكبيرة (إعادة هيكلة/ترقية).**
2. **قبل الـcommit:**
   * **احفظ التغييرات → git add . → git commit**
   * **lefthook يشغّل: dart format → dart analyze → flutter test**
   * **إن فشل أي فحص: صحّح الخطأ ثم أعد المحاولة.**
3. **ادفع الكود:**
   * **git push وافتح Pull Request.**

# **3) ماذا يحدث عند فتح PR؟**

1. **Copilot Code Review يعلّق تلقائيًا باقتراحات (أمن/أسلوب/منطق).**
2. **Codemagic يشغّل Pipeline:**
   * **Format check → Analyze → Tests (≥ 70% تغطية) → Build Debug.**
   * **Semgrep يفحص أمان/جودة.**
3. **إن فشل أي جزء → أصلِح وادفع تحديثًا.**
4. **بعد نجاح CI بالكامل + موافقتين مراجعة (CODEOWNERS) → يسمح بالدمج إلى main.**

# **4) كيف نستخدم الأدوات عند الإصدار؟**

1. **أنشئ Release PR بسجل تغييرات واضح.**
2. **شغّل Codemagic Release pipeline: توقيع/بناء (Android/iOS) + رفع خرائط الرموز لـCrashlytics.**
3. **Staged rollout: 10% → 50% → 100% خلال 48 ساعة.**
4. **راقب Crash-free وPerformance؛ إن ظهرت مشكلة:**
   * **عطل الميزة عبر Feature Flag أو ارجع للإصدار السابق.**
   * **افتح تذكرة تحسين واجعل Continue/Copilot يساعدان في الإصلاح.**

# **5) صيانة أسبوعية (روتين ثابت)**

* **Renovate PRs: راجع وادمج تحديثات التبعيات بعد مرورها على CI.**
* **Semgrep Rules: حدّث القواعد إن لزم.**
* **تقارير CI: وقت البناء، حالات الفشل المتكررة، تغطية الاختبارات.**
* **تنظيف فروع قديمة + مراجعة الاقتراحات المتكررة من Copilot لتحويلها إلى Components مشتركة.**

# **6) كيف نستفيد من الذكاء الاصطناعي بأمان وفعالية؟**

* **اكتب طلبات دقيقة: “حوّل هذا الـWidget إلى Stateless وادعم RTL وتست قبل الدمج”.**
* **اجعل Copilot يولّد اختبارات ووثائق موجزة لكل Use Case.**
* **استخدم Continue عندما تريد تغييرات عابرة للملفات (إعادة تسمية شاملة، توحيد أسلوب، ترقية API).**
* **لا ترسل أسرار/مفاتيح—أي شيء حساس يُدار عبر CI Secrets فقط.**
* **دوّن القرارات المتكررة (من اقتراحات AI) في Guidelines لتصبح معيارًا.**

## **خلاصة التنفيذ:**

* **Copilot: اكتب/حسّن الكود لحظيًا داخل IDE.**
* **Copilot Code Review: تعليقات ذكية تلقائية على كل PR.**
* **Continue: نفّذ Refactor/ترقيات واسعة بسرعة وأمان.**
* **lefthook + very\_good\_analysis: يمنعان الكومِت الرديء قبل أن يصل للمستودع.**
* **Semgrep: يلتقط مشاكل أمن/جودة مبكرًا.**
* **Codemagic: يفرض Gates (analyze/test/build) ويمنع الدمج إن فشل شيء.**
* **Renovate: يضمن تبعيات حديثة بلا كسر مفاجئ.**
* **FlutterFlow: للنماذج الأولية فقط—صدّر الكود وراجعه قبل الدمج.**

ملاحظات & تحسينات بنية التطبيق

# **ملاحظات & تحسينات بنية التطبيق**

## **0) (Overview)**

**التدفق العام:**Flutter App (UI/State/Cache) → API (Auth/Cart/Orders/Payments/Notifications) → DB/PIM/Services  
**الطرف الثالث:** Firebase/Crashlytics, GA4/BigQuery, Meta Pixel, Google Ads, Sentry (أو بديل)، Codemagic (CI/CD), TestFlight/Play Console  
**المراقبة:** Logs + Metrics + Traces + Alerts  
**الإصدارات:** Feature Flags → QA/Staging → Beta → Prod (تدرّج Rollout%)  
**هرم الاختبارات:** Unit → Widget/Golden → Integration/E2E → Post-release Monitoring

## **1) جداول قياسية لإضافتها (Templates)**

### **1.1 Error Catalog**

| **module** | **http\_status** | **error\_code** | **user\_msg\_ar** | **user\_msg\_en** | **severity** | **retry\_action** | **telemetry\_event** | **SLO** | **fallback\_ui** | **test\_case\_id** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| auth | 401 | AUTH\_EXPIRED | انتهت صلاحية الجلسة… | Session expired… | high | show\_login | auth\_error | < 0.5% | modal\_login | TC-ERR-001 |

**معايير:** توحيد صياغة الرسائل، backoff + jitter، عدم الاعتماد على اللون فقط لبيان الفشل.

### **1.2 Cache/TTL Matrix (stale-while-revalidate)**

| **screen/data** | **source** | **TTL** | **invalidate\_on** | **offline\_behavior** | **TTR\_from\_cache** | **TTU** | **test\_ids** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PLP | /products?cat=… | 10m | price\_change, sale\_start | show\_cached + banner\_offline | <300ms | <2s | TC-CACHE-PLP-\* |
| PDP | /product/:id | 5m | stock\_change | show\_cached\_basic | <300ms | <1.5s | TC-CACHE-PDP-\* |
| Cart | /cart | 0 (live) | add/remove/qty | hard-failover msg | — | <800ms | TC-CACHE-CART-\* |

### **1.3 Deep Link & Notifications Matrix**

| **type** | **route** | **required\_params** | **optional** | **fallback** | **analytics\_event** | **expiry** | **test\_case\_id** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PDP | /p/:id | id | ref, coupon | open\_home | dl\_open\_pdp | 24h | TC-DL-001 |
| Search | /s | q | sort | open\_search\_empty | dl\_open\_search | — | TC-DL-010 |

### **1.4 Accessibility Checklist (WCAG AA)**

| **screen** | **min\_touch\_target(≥48dp)** | **contrast(AA)** | **semantics/labels** | **dynamic\_text(120%)** | **dark/light** | **keyboard\_nav** |
| --- | --- | --- | --- | --- | --- | --- |
| Checkout | ✔ | ✔ | ✔ | ✔ | ✔ | N/A |

### **1.5 Alerting & SLOs**

| **metric** | **threshold** | **window** | **channel** | **owner** | **runbook** | **auto\_stop\_rollout** |
| --- | --- | --- | --- | --- | --- | --- |
| Crash-free sessions | <98.5% | 1h | Slack/Email | Mobile Lead | RB-CRASH | ✔ |
| Checkout conversion Δ | −15% vs prev | 2h | Slack/Email | PM + Eng | RB-CONV | ✔ |
| Start-up P95 | >3.5s | 1h | Slack | Mobile Lead | RB-STARTUP | ✔ |

### **1.6 Device/OS Test Matrix**

| **os** | **version** | **device class** | **screen** | **locale** | **theme** | **network** |
| --- | --- | --- | --- | --- | --- | --- |
| iOS | 16/17/18 | mid/high | small/large | ar/en | dark/light | 3G/poor/wifi |
| Android | 10–14 | low/mid/high | small/large | ar/en | dark/light | 3G/poor/wifi |

### **1.7 Performance Budgets**

| **scenario** | **P50** | **P95** | **notes** | **test\_case\_id** |
| --- | --- | --- | --- | --- |
| Cold start to first paint | 1.5s | 3.5s | pre-warm cache, defer non-critical | TC-PERF-001 |
| PLP initial render | 0.9s | 2.0s | skeleton + SWR | TC-PERF-PLP |
| PDP ready to interact | 1.0s | 2.2s | image lazy-loading | TC-PERF-PDP |

### **1.8 CI/CD Pipeline (Codemagic)**

| **stage** | **purpose** | **gates/outputs** |
| --- | --- | --- |
| Lint/Format | التزام بالأسلوب وجودة الكود | فشل عند أخطاء lint |
| Unit/Widget/Golden | جودة منطق وواجهة | تغطية ≥80% للوحدات الحرجة |
| Integration/E2E | رحلات حرجة (login, add2cart, pay) | نجاح 100% |
| Build & Size Budgets | التحقق من الحجم | لو تخطى الميزانية → فشل |
| Sign & Upload | توقيع وإرسال للمتاجر | سجلات إصدار، build id |
| Release Gate | تحقق SLO & تنبيهات | إيقاف تلقائي عند كسر الحدود |

### **1.9 Security & Privacy Controls**

| **area** | **control** | **status** | **notes** |
| --- | --- | --- | --- |
| Tokens | secure storage (Keychain/Keystore) |  | منع logs |
| API | rate limiting, idempotency keys |  | حماية إعادة الطلب |
| PII | masking in logs |  | التزام Privacy |
| Screenshots | منع على شاشات حساسة (اختياري) |  | Android/iOS flags |
| Permissions | rationale + deny paths |  | UX بدائل عند الرفض |

### **1.10 Analytics Event Schema (GA4/Firebase)**

| **event** | **params** | **when** | **owner** | **test\_id** |
| --- | --- | --- | --- | --- |
| view\_item\_list | list\_id, cat\_id | فتح PLP | PM/Mobile | TC-AN-PLP |
| view\_item | item\_id, price | فتح PDP | PM/Mobile | TC-AN-PDP |
| add\_to\_cart | item\_id, qty, price | زر الإضافة | PM/Mobile | TC-AN-ATC |
| begin\_checkout | items\_value | بدء الدفع | PM/Mobile | TC-AN-CHK |
| purchase | tr\_id, value, items | نجاح الدفع | PM/Mobile | TC-AN-PUR |

### **1.11 Release & Store Ops**

| **item** | **owner** | **checklist** |
| --- | --- | --- |
| App Store assets | Design | لقطات/أيقونات/Privacy labels |
| Versioning (SemVer) | Eng | bump + changelog |
| Beta (TestFlight/Play) | QA | دعاة داخليون + feedback |
| Review responses | Support | قوالب ردود رسمية |

### **1.12 Runbooks/Playbooks**

| **runbook** | **trigger** | **steps (مختصر)** | **owner** |
| --- | --- | --- | --- |
| RB-ROLLBACK | SLO مكسور | freeze rollout → revert → verify | Eng Lead |
| RB-INCIDENT | spike errors | form war-room → assign → comms | On-call |
| RB-HOTFIX | blocker bug | branch hotfix → QA fast → release | Eng |

## **2) معايير تفصيلية لكل بند (قابلة للتنفيذ)**

### **2.1 Error Map**

* **التنفيذ:** مكوّن UI موحّد للأخطاء + طبقة ترجمة + سياسة retry/backoff.
* **اختبارات:** Golden لثبات النص/التصميم، محاكاة مهلات/انقطاع/401/409/422.
* **مقاييس:** معدل ظهور كل error\_code، نسبة نجاح إعادة المحاولة.

### **2.2 Cache/TTL (SWR**

* **التنفيذ:** in-memory + disk + API cache، invalidation عند تغيّر سعر/مخزون/عروض.
* **اختبارات:** offline-first لعرض الكاش، قياس TTR/TTU، صحة invalidation.

### **2.3 Deep Links & Notifications**

* **التنفيذ:** مصفوفة روابط + فحص المعلمات + fallback للصفحة المناسبة + events.
* **اختبارات:** E2E من إشعار حتى الشاشة المستهدفة + صلاحية الروابط المنتهية.

### **2.4 Accessibility**

* **التنفيذ:** semantics/labels، حجم لمس 48dp، تباين AA، دعم تكبير الخطّ.
* **اختبارات:** أدوات الوصول + لقطات Golden للوضعين الداكن/الفاتح.

### **2.5 Post-release Monitoring**

* **التنفيذ:** Crashlytics/Sentry + لوحات قياس + تنبيهات SLO + إيقاف rollout تلقائي.
* **اختبارات:** firing تنبيه تجريبي، مقارنة cohort إصدار جديد بالقديم.

### **2.6 Architecture/Environments**

* **التنفيذ:** Dev/QA/Pre-Prod/Prod، secrets عبر Vault، فصل التحليلات على البيئات.
* **اختبارات:** ضمان تحميل مفاتيح البيئة الصحيحة، منع مزج بيانات.

### **2.7 Code Quality & Workflow**

* **التنفيذ:** Lint/Format، نمط حالة موحّد (Bloc/Riverpod)، PR rules، SemVer.
* **اختبارات:** فشل تلقائي عند lint errors/low coverage.

### **2.8 CI/CD (Codemagic)**

* **التنفيذ:** stages أعلاه + artifacts (JUnit/Coverage/Size) + gates.
* **اختبارات:** pipeline dry-run، فشل متعمد لقياس الحماية.

### **2.9 Testing Pyramid**

* **التنفيذ:** وحدات، ودجت/Golden، تكامل/E2E لرحلات الشراء.
* **اختبارات:** Matrix الأجهزة/الشبكات/اللغات/الثيمات.

### **2.10 Performance Budgets**

* **التنفيذ:** قياس start-up/PLP/PDP/Checkout، lazy-load، defer non-critical.
* **اختبارات:** قياس P50/P95 تلقائي داخل CI، فشل عند تجاوز الحدود.

### **2.11 Security & Privacy**

* **التنفيذ:** secure storage، masking logs، rate-limit، idempotency، سياسات أذونات.
* **اختبارات:** pentest خفيف للعميل، التحقق من منع لقطات على الشاشات الحساسة (اختياري).

### **2.12 Analytics & Measurement**

* **التنفيذ:** مخطط موحّد للأحداث/المعلمات، ربط BigQuery/لوحات.
* **اختبارات:** مطابقة عدد الأحداث مع عدد الرحلات، عدم وجود events يتيمة.

### **2.13 Product/UX (Degrade/Offline/Image Policy)**

* **التنفيذ:** سلوك واضح دون اتصال، placeholders/shimmer، سياسة صور (أبعاد/صيَغ/WebP).
* **اختبارات:** عرض صحيح عند غياب الشبكة/الصور/المخزون.

### **2.14 Release & Store**

* **التنفيذ:** أصول المتاجر، Privacy labels، ردود المراجعات.
* **اختبارات:** قائمة تحقق قبل الإرسال (pre-submission checklist).

### **2.15 Runbooks**

* **التنفيذ:** وثائق rollback/incident/hotfix، قنوات اتصال محدّدة.
* **اختبارات:** مناورة (drill) قصيرة ربع سنوية.

## **3) معايير القبول (Definition of Done)**

* **Error Map:** 100% الشاشات تستخدم نفس مكوّن الخطأ + Golden ثابت.
* **Cache/TTL:** PLP/PDP TTR\_from\_cache < 300ms، TTU ≤ 2s (شبكة جيدة).
* **Deep Links:** 0% فشل في روابط PDP/PLP/Search الأساسية + أحداث تتبّع صحيحة.
* **Accessibility:** اجتياز AA للعناصر الحرجة + لا انكسار عند 120% خط.
* **Monitoring:** تنبيهات فعّالة + auto-stop rollout عند كسر SLO.
* **CI/CD:** pipeline كامل مع gates فعّالة وتغطية ≥80% للوحدات الحساسة.
* **Performance:** احترام ميزانيات P50/P95 الموضوعة.
* **Security/Privacy:** عدم تسريب PII في السجلات + rate-limit/idempotency مفعل.
* **Analytics:** الأحداث الرئيسية تعمل من الفتح حتى الشراء، بلا ازدواجية.

## **4) خطة إغلاق الفجوات (Sprint واحد مقترح)**

1. إضافة الجداول 1.1–1.12 للمستند كـ baseline.
2. تفعيل Codemagic مع المراحل والبوابات والأثر المرئي (Coverage/Size/Reports).
3. إعداد Crashlytics/Sentry + تنبيهات SLO للقنوات المناسبة.
4. بناء مجموعة اختبارات: unit/widget/golden/integration + Matrix الأجهزة.
5. نشر Runbooks مختصرة (rollback/incident/hotfix) وتجربة مناورة واحدة.

## **5) ملاحظات تنسيق/إدارة**

* اربط كل صف “test\_case\_id” بحالة اختبار في نظام QA (مثلاً TC-AN-PDP).
* عيّن “owner” واضح لكل عنصر (PM/Mobile/Backend/QA).
* حافظ على نسخة عربية/إنجليزية للرسائل الظاهرة للمستخدم.

التدفق العام:

**التدفق العام:**

المستخدم

│

▼

Flutter App (UI/State/Cache/Networking)

│ HTTP/GRPC + Auth + Retry/Backoff + DL/Push

▼

API Gateway (Rate limit, AuthZ, Observability)

▼

Micro/Services (Catalog, Cart, Checkout, Payments, Orders, CMS, RMA, Loyalty)

▼

Datastores (SQL/NoSQL, Search, Cache) — CDN/Media — Queue/Events

▼

Observability (Logs/Metrics/Traces), CI/CD, Feature Flags, Analytics

**أهداف التصميم:** قابلية التوسّع، الاستقرار، الأداء (P50/P95)، تجربة متسقة، أمان وخصوصية، سهولة التطوير والصيانة.

# **1) تصميم الواجهة في (App Architecture)**

## **1.1 هيكلة المشروع (Modularization)**

* **Feature-first**: features/<feature>/presentation|domain|data + طبقات مشتركة core/.
* **Layered داخل كل ميزة**:
  + presentation: Widgets, Pages, go\_router, ViewModels/Bloc.
  + domain: UseCases, Entities, Repositories (interfaces).
  + data: Repositories (impl), DataSources (remote/local), DTOs, Mappers.
* **core/**: ثوابت، أخطاء، Network, DI, Theming, Localization, Analytics.

## **1.2 إدارة الحالة (State Management)**

* توصية: **Bloc/Cubit** أو **Riverpod** (مع ProviderScope)، مع فصل العرض عن المنطق.
* عقود واضحة بين View ↔ ViewModel/Bloc ↔ UseCase.
* Events/States قابلة للاختبار + عدم تمرير BuildContext داخل الطبقات المنطقية.

## **1.3 التنقّل (Navigation)**

* **go\_router** مع أسماء مسارات ثابتة، ودعم deep-link وstate restoration.
* رسم **Graph** للرحلات الحرجة (Onboarding → Auth → PLP → PDP → Cart → Checkout → Orders).

## **1.4 DI/Config**

* **get\_it** (أو Riverpod DI) لحقن الاعتماديات.
* ملفات **flavors**: dev, staging, prod (مع Bundle IDs مختلفة).

## **1.5 الثيم/الوصولية/التعريب**

* Theme موحّد (Light/Dark)، مقاسات تفاعلية ≥48dp، تباين WCAG AA.
* intl للتعريب (AR/EN)، RTL مدعوم، تنسيقات أرقام/عملات.

# **2) طبقة البيانات داخل التطبيق (Data Layer)**

## **2.1 الشبكات (Networking)**

* **Dio** + Interceptors: Auth (Bearer/OTP), Retry with exponential backoff + jitter, Logging (مراعاة الخصوصية).
* **Idempotency** لعمليات الشراء/الدفع، **Circuit Breaker** عند الأعطال المتكررة.

## **2.2 التخزين المحلي والكاش (Offline/SWR)**

* محركات: **Isar/Drift/Hive**؛ اختيار واحد حسب الاحتياج.
* سياسة **stale-while-revalidate**: عرض من الكاش ثم تحديث صامت.
* **TTL Matrix** لكل نوع بيانات (PLP/PDP/Cart/Configs).

## **2.3 DTOs/Mappers/Errors**

* فصل DTOs عن Entities، توحيد أخطاء المجال: NetworkError | AuthError | BusinessError | Unknown.

# **3) تجربة الأخطاء والمرونة (Resilience & Error UX)**

* **Error Catalog** موحّد (HTTP/Business) مع رسائل AR/EN، وإجراءات retry/fallback.
* مكوّن UI موحّد للأخطاء + **Golden tests** لثبات التصاميم.
* سيناريوهات: ضعف الشبكة/مهلة/401/409/422/نفاد مخزون/رفض دفع.

**قالب جدول Error Catalog**

| **module** | **http\_status** | **error\_code** | **user\_msg\_ar** | **user\_msg\_en** | **severity** | **retry** | **telemetry** | **fallback\_ui** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| checkout | 402 | PAY\_3DS\_FAIL | فشل التحقق البنكي… | Bank 3DS failed… | high | retry\_once | pay\_3ds\_fail | info\_banner |

# **4) الأداء (Performance)**

* **ميزانيات**: Cold Start P50/P95، PLP render، PDP ready، Checkout step.
* تتبّع **jank** (raster/build times)، استخدام Isolates للأعمال الثقيلة.
* صور: cached\_network\_image + precacheImage + WebP/AVIF + أحجام مناسبة.
* قوائم كبيرة: ListView.builder, Slivers, تجنّب العمل الثقيل في build.

**قالب Performance Budgets**

| السيناريو | P50 | P95 | ملاحظات |
| --- | --- | --- | --- |
| Cold start (to first paint) | 1.5s | 3.5s | defer non-critical, prewarm |
| PLP initial render | 0.9s | 2.0s | skeleton + SWR |
| PDP ready to interact | 1.0s | 2.2s | lazy images |

# **5) الأمان والخصوصية (Security & Privacy)**

* **Secure Storage** للتوكينات (Keychain/Keystore) + Masking للـ logs.
* **TLS** محدث؛ (اختياري) Certificate Pinning بحذر.
* كشف Root/Jailbreak (تحذير/تقليل صلاحيات)، Play Integrity/App Attest (عند الحاجة).
* سياسات لقطات الشاشة للشاشات الحساسة (اختياري)، ATT على iOS عند التتبع.

# **6) التحليلات والقياس (Analytics & Event Schema)**

* مخطط أحداث موحّد **GA4/Firebase**: open → view\_item\_list → view\_item → add\_to\_cart → begin\_checkout → add\_payment\_info → purchase.
* تعيين user\_id = معرف العميل، ربط الأجهزة/الشرائح.

**قالب Event Schema**

| event | params | when |
| --- | --- | --- |
| view\_item | item\_id, price, cat\_id | عند فتح PDP |
| add\_to\_cart | item\_id, qty, price | عند إضافة للسلة |
| purchase | tr\_id, value, items | نجاح الدفع |

# **7) الإشعارات والروابط العميقة (Push & Deep Links)**

* **FCM**: إدارة tokens، قنوات Android، foreground/background/terminated.
* **Dynamic Links/OneLink**: روابط مؤجّلة لفتح PDP/PLP/Cart/Order.

**قالب Deep Link Matrix**

| type | route | required\_params | fallback | analytics |
| --- | --- | --- | --- | --- |
| PDP | /p/:id | id | /home | dl\_open\_pdp |

# **8) الرايات والتجارب (Feature Flags & A/B)**

* **Remote Config** لمفاتيح الميزات الحساسة + تجارب A/B.
* توثيق دورة حياة العلم (create → roll → measure → retire).

# **9) إعداد البيئات (Flavors & Config)**

* Flavors: dev/staging/prod، مفاتيح/Endpoints/Analytics منفصلة.
* رمز/أيقونة/اسم للتطبيق لكل بيئة، ومنع خلط حسابات الإنتاج.

# **10) CI/CD وخط النشر (Codemagic مثال)**

**المراحل:** Lint/Format → Unit/Widget/Golden → Integration/E2E → Build & Size → Sign & Upload → Release Gate.

* **بوابات قبول**: تغطية اختبارية، ميزانيات الأداء، لا أعطال حرجة، موافقة QA.
* تقارير JUnit/Coverage/Screenshots artifacts، طرح تدريجي (rollout%).

# **11) هرم الاختبارات (Testing Pyramid)**

* **Unit**: المرافق/الفالديشن/UseCases.
* **Widget/Golden**: ثبات الواجهة وحالات الخطأ.
* **Integration/E2E**: الرحلات الحرجة (Auth, PLP→PDP→Cart→Pay→Order).
* **Contract tests** مع API، **Network mocking**، **Screenshot tests** (AR/EN, Light/Dark)، **A11y tests**.

**Matrix الأجهزة/الشبكات**

| os | versions | devices | locales | theme | network |
| --- | --- | --- | --- | --- | --- |
| iOS | 16–18 | small/large | ar/en | dark/light | 3G/poor/wifi |
| Android | 10–14 | low/mid/high | ar/en | dark/light | 3G/poor/wifi |

# **12) المراقبة والتشغيل (Observability & Ops)**

* Crashlytics/Sentry + Logs + Performance + Dashboards.
* **SLO/Alerts**: Crash-free sessions، Start-up P95، Checkout Conversion.
* **Runbooks**: Incident, Rollback, Hotfix.

**قالب SLO/Alerting**

| metric | threshold | window | owner | channel | auto\_stop\_rollout |
| --- | --- | --- | --- | --- | --- |
| Crash-free | <98.5% | 1h | Mobile Lead | Slack/Email | ✔ |
| Checkout Δ | −15% vs prev | 2h | PM + Eng | Slack | ✔ |

# **13) المتجر والإصدار (Store & Release)**

* أصول المتاجر (لقطات/أيقونات/Privacy labels)، in\_app\_review، in\_app\_update (Android).
* سياسة دعم الإصدارات (N-1/N-2)، ردود المراجعات، خطّة الطرح التدريجي.

# **14) قوالب جاهزة للاستخدام**

## **14.1 بنية مجلدات (مثال)**

lib/

core/

config/ (env, flavors)

errors/

network/ (dio, interceptors)

di/

theme/

l10n/

analytics/

features/

catalog/

presentation/

domain/

data/

product/

cart/

checkout/

orders/

account/

app.dart

main\_dev.dart

main\_staging.dart

main\_prod.dart

## **14.2 قالب README داخلي**

* نظرة عامة
* إعداد البيئة (flavors, keys)
* تشغيل الاختبارات
* أسلوب الكود (lint/format)
* كيفية إضافة ميزة جديدة (skeleton)
* مراقبة/لوحات

# **15) مسار التنفيذ (Roadmap مختصر)**

1. Foundation: Network/DI/Flavors/Theme/L10n/Error+Cache policies.
2. Flows: PLP → PDP → Cart → Checkout → Orders → RMA/Wallet.
3. 3P: Firebase (Core+Analytics+Crash+Perf+RC+FCM) → Attribution → Payments → Maps/Media.
4. Hardening: Performance, A11y, Security, Tests, SLO/Alerts.
5. Launch: Store assets, Beta, Rollout, Support.

هذا الدليل يُستخدم كمرجع سريع + قائمة تشغيل (Playbook). عند البدء بكل ميزة، اربطها بجدول: **المدخلات/المخرجات، أخطاء متوقعة، أحداث تحليلية، حالات اختبار، ميزانيات الأداء**.

# **16) قائمة تحقق نهائية (Definition of Done)**

**الغرض:** تحديد معايير قبول دقيقة وقابلة للقياس قبل الانتقال من مرحلة إلى التي تليها (Foundation → Flows → 3P → Hardening → Launch).

## **- بوابات القبول العامة (Gates)**

## **- أدلة الإثبات المطلوب إرفاقها (Artifacts)**

* **تقارير اختبار**: JUnit + تغطية + لقطات Golden/Screenshot tests.
* **لوحات قياس**: Firebase DebugView للقناة التجريبية + لوحة Crashlytics + Firebase Performance.
* **سجلات E2E**: فيديو قصير (اختياري) لكل رحلة حرجة على iOS/Android.
* **نتائج أداء**: Export لمقاييس P50/P95 لكل سيناريو (Start/PLP/PDP/Checkout).
* **أمن/خصوصية**: لقطات تكوين secure storage، فحص repo لعدم وجود أسرار، سياسة ATT/Permissions.
* **جاهزية المتجر**: رابط داخلي لـ TestFlight/Play Internal + قائمة تحقق ما قبل الإرسال.

## **15.3 DoD تفصيلي لكل وحدة رئيسية**

### **- الشبكات/التهيئة (Networking & Config)**

### **- الهوية والجلسة (Auth/Session)**

### **- الكتالوج/PLP**

### **- صفحة المنتج/الوسائط (PDP/Media)**

### **- السلة/الكوبونات (Cart/Coupons)**

### **- الدفع (Checkout/Payments)**

### **- الطلبات/المرتجعات/المحفظة (Orders/RMA/Wallet)**

### **- الإشعارات والروابط العميقة (Push & Deep Links)**

### **- التحليلات والتجارب (Analytics & A/B)**

### **- الأداء والذاكرة (Performance & Memory)**

### **- الوصولية والتعريب (A11y & i18n)**

### **- الأمان والخصوصية (Security & Privacy)**

### **- CI/CD والإصدار (CI/CD & Release)**

## **15.4 قالب توقيع الاعتماد (Sign‑off Template)**

| **البند** | **المالك** | **الدور** | **القرار** | **التاريخ** | **ملاحظات** |
| --- | --- | --- | --- | --- | --- |
| DoD – المرحلة الحالية | Eng Lead | تقنية | ✅/❌ | YYYY‑MM‑DD |  |
| DoD – QA/UAT | QA Lead | جودة | ✅/❌ | YYYY‑MM‑DD |  |
| DoD – Product | PM | منتج | ✅/❌ | YYYY‑MM‑DD |  |

## **15.5 ملاحق تفصيلية (تكملة البنود)**

هذا الملحق يملأ أي عناوين كانت ظاهرًا دون تفاصيل، ويُستخدم كمرجع تنفيذ مباشر.

### **- Runbooks/Playbooks (مُفصّلة)**

* **RB‑INCIDENT (حوادث Sev1–Sev3):**
  1. تشكيل غرفة طوارئ (PM+Eng+Ops) خلال ≤10 دقائق.
  2. تجميد النشر وبدء قياس الأثر (المناطق/الإصدارات/الأجهزة).
  3. عزل السبب (API vs App vs 3P) عبر اللوحات والسجلات.
  4. تخفيف الأثر (Feature Flag OFF / Kill‑Switch / Banner تنبيهي داخل التطبيق).
  5. توثيق Post‑mortem خلال 48 ساعة مع إجراءات منع التكرار.
* **RB‑ROLLBACK:**
  1. إيقاف Rollout% الحالي. 2) إعادة تفعيل الإصدار السابق (N‑1). 3) التحقق من SLOs. 4) فتح تذكرة Root‑Cause.
* **RB‑HOTFIX:**
  1. فرع hotfix من الإصدار الحالي. 2) اختبارات سريعة (Unit/Widget/Smoke). 3) رفع Beta مغلقة. 4) نشر تدريجي مع مراقبة.

### **- Store & Release (اكتمال المتجر)**

* **App Store/Play:** لقطات محلية AR/EN، أيقونات، Privacy labels، وصف مختصر/طويل، كلمات مفتاحية.
* **Beta:** TestFlight/Play Internal مع تعليقات داخلية، صفحة ملاحظات الإصدار.
* **طرح تدريجي:** 5% → 25% → 50% → 100% مع عتبات إيقاف تلقائي (Crash‑free <98.5% أو −15% تحويل).
* **الرد على المراجعات:** نماذج ردود، تحويل الشكاوى إلى تذاكر دعم.

### **- Feature Flags Lifecycle**

* تعريف العلم: المالك، نطاق التأثير، خطة رجوع.
* تجزئة الجمهور: نسبة/خصائص (دولة/نسخة/لغة).
* قياس: أحداث نجاح/فشل، نافذة قياس، معايير إغلاق.
* تقاعد العلم: تنظيف الكود خلال ≤2 سبوع بعد القرار.

### **- Security & Privacy (تفصيلي)**

* **Tokens:** تخزين عبر Keychain/Keystore (flutter\_secure\_storage)، منع النسخ الاحتياطي، دوران توكن.
* **Logs:** حذف/إخفاء PII، تعطيل logging في الإنتاج، حدود حجم السجل.
* **Permissions/ATT:** شروحات، مسارات بديلة عند الرفض، طلب ATT فقط عند الحاجة.
* **Network:** TLS حديث، (اختياري) Certificate Pinning بحذر، مهلات معقولة.
* **Integrity:** Play Integrity/App Attest للعمليات الحرجة (دفع/ولاء).

### **- Flavors & Config**

* **flavors:** dev/staging/prod مع Bundle Ids مختلفة، اسم/أيقونة مميزة.
* **Config:** مفاتيح/Endpoints منفصلة، Analytics stream لكل بيئة، منع مزج بيانات الإنتاج.

### **- Data Layer (خيارات محلية)**

* اختيار **Isar** للكاش السريع وأوفلاين، أو **Drift** لو جداول/علاقات معقدة.
* سياسة TTL/Invalidation لكل كيان (PLP/PDP/Cart/Config).
* ترقية المخطط (migrations) موثقة ومختبرة.

### **- Networking Interceptors**

* **Auth:** إدراج Bearer/OTP، تجديد تلقائي عند 401 (مرة واحدة).
* **Retry:** Exponential backoff + jitter، حظر إعادة محاولات غير idempotent.
* **Idempotency:** Header/Key لخطوات الدفع.
* **Logging:** إخفاء الحقول الحساسة، تعطيل أجسام الاستجابات في الإنتاج.

### **- Analytics Mapping (GA4)**

* view\_item\_list(cat\_id, list\_id) عند PLP.
* view\_item(item\_id, price, cat\_id) عند PDP.
* add\_to\_cart(item\_id, qty, price) عند الإضافة.
* begin\_checkout(value) عند بدء الدفع.
* purchase(tr\_id, value, items[]) عند نجاح الدفع.

### **- Deep Links Matrix (أمثلة مكتملة)**

| type | route | required | optional | fallback | analytics |
| --- | --- | --- | --- | --- | --- |
| PDP | /p/:id | id | ref | /home | dl\_open\_pdp |
| PLP | /c/:id | id | sort | /home | dl\_open\_plp |
| Search | /s | q | sort | /s?q= | dl\_open\_search |
| Cart | /cart | — | coupon | /home | dl\_open\_cart |
| Order | /o/:id | id | — | /orders | dl\_open\_order |

### **- Performance Budgets (أرقام نهائية)**

* **Cold Start:** P50 ≤ 1.5s، P95 ≤ 3.5s.
* **PLP:** P50 ≤ 0.9s، P95 ≤ 2.0s.
* **PDP:** P50 ≤ 1.0s، P95 ≤ 2.2s.
* **Checkout Step:** P95 ≤ 1.5s لشاشة.

### **- Testing Matrices**

* **تغطية:** وحدات ≥80% للطبقات الحرجة؛ Golden للشاشات الستة الأساسية.
* **أجهزة:** iOS (16–18) small/large، Android (10–14) low/mid/high، AR/EN، داكن/فاتح، شبكات 3G/ضعيفة/Wi‑Fi.

### **- Alerting & SLOs**

* **Crash‑free:** <98.5% (1h) → تنبيه Critical + إيقاف طرح.
* **Start‑up P95:** >3.5s (1h) → تنبيه Warning.
* **Checkout Conversion Δ:** −15% مقابل الإصدار السابق (2h) → Critical.

هرم الاختبارات

# **هرم الاختبارات**

Pipeline في CI/CD:  
**Lint → Unit → Widget/Golden → Integration/E2E (على أجهزة) → Build/Sign → Release → Post-release Monitoring**.  
كل طبقة تمنع دخول أخطاء للتي بعدها، ومعها عتبات قبول (Gates) واضحة.

# **1) Unit Tests — اختبارات الوحدات**

**الهدف:** تأكيد منطق الأعمال سريعًا ومعزولًا عن الـUI والشبكة.  
**نغطي:** الدوال والخدمات وUseCases/Repositories (بتبديل المصادر بمُحاكيات).  
**أمثلة:** حساب السعر بعد الكوبون، شروط صلاحية الكوبون، تنسيقات العناوين، مُولِّد مفاتيح idempotency.  
**أدوات:** flutter\_test, mocktail/Mockito, fake\_async.  
**بيانات الاختبار:** Fixtures صغيرة وثابتة، لا اتصالات حقيقية.  
**بوابة القبول:** نجاح 100%، وتغطية ≥ **80%** للمنطق الحرج.  
**مقاييس:** زمن الاختبار < 200ms/اختبار، عدم flakiness.

# **2) Widget / Golden — واجهة معزولة + لقطات ثابتة**

**الهدف:** ضمان سلوك الواجهة وعرض الحالات (loading/empty/error) وثبات الشكل بصريًا.  
**Widget نغطي:** عناصر مثل بطاقة منتج، شريط الإضافة للسلة، نماذج الإدخال…  
**Golden (لقطات):** مقارنة Pixel-perfect لصفحات/مكوّنات أساسية (Error screen، بطاقة PLP، شاشة PDP).  
**أدوات:** flutter\_test (pumpWidget)، ولقطات matchesGoldenFile (يمكن استخدام golden\_toolkit).  
**بيئة:** خطوط/مقاسات ثابتة، صور وهمية (placeholders)، تعطيل الرسوم المتحركة.  
**بوابة القبول:** 100% نجاح لواجهات أساسية + عدم تغيّر ملفات Golden دون مبرر.  
**مقاييس:** تغطية حالات الحالة (states) لكل Widget أساسي.

# **3) Integration / E2E — رحلة كاملة على جهاز**

**الهدف:** اختبار “رحلات حرجة” كأنها مستخدم حقيقي، من داخل التطبيق حتى الـAPI.  
**نغطي:** Auth → PLP → PDP → AddToCart → Coupon → Address → Shipping → Payment(3DS) → Order.  
**أدوات:** حزمة Flutter **integration\_test** (الرسمية) + تشغيل على **محاكٍ/جهاز فعلي** (iOS/Android). يمكن إضافة أدوات مثل Patrol/Appium عند الحاجة.  
**البيئة والبيانات:**

* حسابات اختبار، منتجات مهيّأة (مخزون/أسعار معروفة)، قسائم صحيحة/خاطئة.
* شبكة: ملفات تعريف 3G/ضعيفة، لغتان (AR/EN)، وضع داكن/فاتح.  
  **بوابة القبول:** نجاح 100% لرحلة واحدة على الأقل لكل منصة + عدم flakiness.  
  **مقاييس:** زمن كل خطوة (P95) ضمن الميزانية (مثال: Checkout step ≤ 1.5s)، معدل النجاح عبر الجهاز/الإصدار.

# **4) Post-release Monitoring — مراقبة ما بعد الإطلاق**

**الهدف:** اصطياد ما يفلت من الاختبارات على أجهزة المستخدمين الحقيقية.  
**الأدوات:** **Crashlytics/Sentry** (أعطال)، **Firebase Performance** (بدء/شبكة/إطارات)، **GA4** (الفانِل)، تنبيهات.  
**ما نراقب:**

* Crash-free sessions (%)، Start-up P95، مهلة/أخطاء الشبكة، Funnel التحويل (من open إلى purchase).
* مقارنات cohort للإصدار الجديد مقابل السابق، وإيقاف طرح تلقائي عند كسر العتبات.  
  **بوابة القبول (للطرح):**
* Crash-free ≥ **98.5%**، وعدم هبوط **Checkout Conversion** أكثر من −15% مقارنة بالإصدار السابق.  
  **إجراءات:** Runbooks (Incident/Hotfix/Rollback)، تذاكر جذور المشكلة، تحسينات لاحقة.

## **جدول سريع (لكل طبقة)**

| **الطبقة** | **لماذا؟** | **يغطي** | **أدوات** | **بيانات** | **بوابة القبول** | **مقاييس أساسية** |
| --- | --- | --- | --- | --- | --- | --- |
| Unit | حماية المنطق | UseCases/Repos | flutter\_test + mocktail | Fixtures | نجاح 100% + تغطية ≥80% | زمن <200ms/اختبار |
| Widget/Golden | ثبات الواجهة | Widgets/States/Visual | flutter\_test + golden | صور/خطوط ثابتة | 100% للشاشات الأساسية | صفر تغيّر بصري غير مبرر |
| Integration/E2E | صحة الرحلات | تدفقات الشراء | integration\_test | حسابات/منتجات/قسائم | نجاح كامل على iOS/Android | P95 ضمن الميزانية |
| Post-release | صحة على أرض الواقع | Crash/Perf/Funnel | Crashlytics/Perf/GA4 | بيانات حية | SLOs محققة | Crash-free ≥98.5% |

**ملاحظة///**اجعل أي PR يمرّ بـ Unit + Widget/Golden. نفّذ Integration/E2E ليليًا وعلى كل **Release Candidate**. وبعد الإطلاق، راقب SLOs تلقائيًا مع **Auto-rollback/Stop Rollout** إذا انكسرت العتبات.

(لإغلاق الفجوات)

# **(مهمة لإغلاق الفجوات)**

1. **خريطة المعماريات والبيئات**

* توثيق بيئات: Dev/QA-Staging/Pre-Prod/Prod، ومفاتيح/Secrets عبر Vault، وفصل بيانات التحليلات بين البيئات.
* سياسة **Feature Flags** (تمكين/تعطيل/تجريب) وربطها بالاختبارات.

1. **معايير الكود والجودة**

* Lint/Formatting (effective\_dart)، تغطية اختبارية مستهدفة (≥80% للوحدات الحساسة)، نمط إدارة الحالة (Bloc/Provider/Riverpod) موثّق.
* اتفاقية **Versioning/SemVer**، و**GitFlow/Trunk-Based** مع قواعد Pull Requests.

1. **CI/CD Codemagic**

* Jobs منفصلة: Lint → Tests (unit/widget/integration) → Build → Static Analysis (licenses, size budgets) → Sign & Upload.
* **Release gates**: نجاح الاختبارات + عتبات المراقبة + موافقة QA قبل النشر.
* توزيعات Beta (TestFlight/Play Internal) مع ملاحظات إصدار ومعايير قبول.

1. **الاختبارات المعيارية (Test Pyramid)**

* **Unit** (النماذج/المرافق/الفاليديتورز)، **Widget/Golden** (ثبات UI)، **Integration/E2E** (رحلات الشراء والدفع/تسجيل الدخول/المرتجعات).
* **Matrix الأجهزة/أنظمة التشغيل**: iOS/Android، شاشات صغيرة/كبيرة، شبكات 3G/ضعيفة، لغتين (AR/EN)، وضع داكن/فاتح.

1. **الأداء (Performance Budgets)**

* أهداف P50/P95 لـ: App Start، Frame Render، Time-to-First-Content، زمن تحميل PLP/PDP، زمن تفعيل القسائم.
* تتبّع jank (raster/build times) وتحليل memory/leaks.

1. **الأمان والخصوصية**

* تخزين آمن للتوكينات/المفاتيح، حماية API (Rate limiting/Retry/Idempotency)، قيود لقطات الشاشة عند البيانات الحساسة، تشويش/إخفاء بيانات السجلات، سياسة كلمات المرور/OTP، مكافحة الاحتيال للدفع وCOD.
* صفحة خصوصية وأذونات (ATT/iOS، runtime permissions/Android) ومعالجة الرفض.

1. **التحليلات والقياس**

* مخطط Events/Params موحّد (GA4/Firebase) للـ funnel: open → view\_item\_list → view\_item → add\_to\_cart → begin\_checkout → add\_payment\_info → purchase.
* ربط BigQuery وداشبود (Metabase/Looker) + ضمان تطابق أحداث الروابط العميقة/الإشعارات.

1. **التجربة والمنتج**

* **Offline/Degrade** واضح للسلة، إشارات حالة المخزون المباشرة، fallback للصور (Image policy)، دعم تعدد اللغات، سياسات الخطأ النصي الموحّد للواجهة (tone & style)، الوضع الداكن.
* سياسة المحتوى للصور (أحجام/نِسَب/قصّ/WebP/تحسين)، وقواعد Placeholder وShimmer.

1. **إدارة الإطلاق والمتجر**

* مواد المتجر (لقطات/أيقونات/Privacy labels)، Checklists للمتجرين، توقيع تلقائي، تتبّع مراجعات المستخدمين وخطة الرد.

1. **Runbooks/Playbooks**

* كتيّبات: **Incident response**، **Rollback**، **Hotfix**، **Maintenance windows**، **اتفاقيات التواصل** بين فرق Dev/QA/Support/Ops.

# **توصيات تنفيذية (مختصرة وقابلة للتطبيق فورًا)**

**A) جداول معيارية:**

* **Error Catalog**:  
  module | http\_status | error\_code | user\_msg\_ar | user\_msg\_en | severity | retry | telemetry | SLO | fallback\_ui
* **Cache/TTL Matrix**:  
  screen/data | source | TTL | invalidate\_on | offline\_behavior | test\_ids
* **Deep Link Matrix**:  
  type | route | required\_params | optional | fallback | analytics\_event | test\_case\_id
* **Accessibility Checklist**:  
  screen | min-touch-target | contrast-ratio | semantics-labels | dynamic-type | keyboard-nav
* **Alerting/SLOs**:  
  metric | threshold | window | channel | owner | runbook | auto-rollout-stop?
* **Device/OS Test Matrix**:  
  os | version | device class | screen size | locale | theme | network profile

**B) معايير قبول (Acceptance) لكل بند:**

* **Error Map**: 100% من شاشات الأخطاء تستخدم نفس المكوّن + Golden tests لرسائل الخطأ.
* **Cache/TTLs**: PLP/PDP/Cart تحقّق TTR-from-cache < 300ms وTTU < 2s (شبكة جيدة).
* **Deep Links/Notifications**: 0% فشل في E2E على الروابط الأساسية، مع تتبّع حدث فتح صحيح.
* **Accessibility**: اجتياز WCAG-AA للعناصر الحرجة + لا انكسار تخطيطات عند 120% نص.
* **Post-release**: تنبيهات فعالة على Crash-free < 98.5% أو Checkout Conversion −15% مقارنة بالإصدار السابق مع إيقاف rollout تلقائي.

**C) سدّ الفجوات بسرعة (Sprint واحد):**

1. إضافة الجداول الستة أعلاه للملف واعتمادها كـ baseline.
2. تجهيز قوالب اختبارات: unit/widget/golden/integration + مصفوفة الأجهزة.
3. إعداد Codemagic pipelines ببوابات قبول، ورفع التقارير (JUnit/coverage) كأثرٍ مرئي.
4. تفعيل Crashlytics + Sentry/Logs + GA4 events الموحّدة وربط تنبيهات Slack/Email.
5. كتابة Runbook للـ rollback وIncident response.