**Prismatica Android — Launch Plan**

**1. Overview & Value**

Prismatica is an offline-first Android app for tracking devices/items with fast CRUD operations and simple notes. Users create a local account (signup/login), manage a grid of items, and—optionally—enable SMS alerts for simple triggers such as low inventory. Data is stored in a persistent on-device SQLite database; the app functions fully without network access. Target users include makers, students, and hobbyists who need a lightweight way to maintain small inventories or device lists on their phone.

**Core goals**

* Make basic inventory/event/weight tracking dead simple.
* Persist data reliably across app restarts and phone reboots.
* Keep permissions minimal (SMS is optional and gated by runtime consent).
* Favor responsive UI with immediate feedback.

**2. Store Listing (what users will see)**

**App name:** Prismatica  
**Icon concept:** A simple prism/sensor glyph in a purple accent, with a subtle grid motif to suggest inventory.

**Short description (≤80 chars):**  
Track devices and notes offline. Optional SMS alerts.

**Full description (key points):**

* **Offline first:** All data is stored locally via SQLite.
* **Fast CRUD:** Add, edit, delete, and view items in a tidy grid.
* **Notes:** Attach short notes per item.
* **Optional SMS alerts:** Enable runtime permission to send self-alerts (e.g., low quantity).
* **Privacy-respecting:** No analytics and no cloud sync by default.

**Feature bullets:**

* Sign up / log in locally (no network needed)
* Device/items grid with add, edit, delete
* Notes per item
* Data persists across restarts
* Optional SMS test screen to verify permissions and sending

**Screenshots to include:**

1. Login/Signup, 2) Devices grid (with notes on at least one card), 3) Add/Edit dialog, 4) SMS Notifications screen (Denied and Granted states).

**3. Platforms, Versions, and Devices**

* **Min SDK:** 26 (Android 8.0).
* **Target SDK:** 36 (per module config).
* **Form factors:** Phones (portrait primary).
* **Tested devices:** Android Emulator “Medium Phone API 36.”
* **Why these versions:** Min 26 allows modern APIs; target 36 keeps the app current and compliant with Play requirements.

**4. Permissions & Privacy**

**Requested permissions:**

* SEND\_SMS — runtime only; **not required** to use the app. The app declares  
  <uses-feature android:name="android.hardware.telephony" android:required="false" />  
  so devices without telephony can still install/run.

**Justification:**

* Used solely to let a user send themselves alerts. If the user denies permission, the app continues to function with no loss of core features.

**Data handling & privacy:**

* All content (users/items/notes) is stored locally in SQLite on the device.
* No network calls, cloud storage, or analytics.
* No collection of personal data beyond the username/password the user creates locally.
* SMS is user-initiated and optional.

**5. Monetization**

**Course release:** Free, no ads.  
**Future options (not in this release):** One-time purchase for advanced features (export/CSV, cloud backup), or an ad-free Pro tier if ads are ever introduced (not planned now).

**6. QA Plan & Pre-Launch Checklist**

**Manual test cases (pass on min/target emulators):**

* **Auth:** Sign up new user → relaunch → log in. Invalid creds show a clear toast.
* **CRUD:** Add item (with notes) → item appears; Edit (update qty + notes) → persists; Delete → disappears; relaunch → state preserved.
* **Orientation:** Basic rotation sanity (UI remains stable or state restored).
* **SMS (Denied):** Request → Deny → status shows DENIED; “Send Test SMS” blocked; app still works.
* **SMS (Granted):** Request → Allow → Send test; show success toast/feedback (or receive on second emulator).
* **Permissions audit:** No unexpected permission prompts; app runs without SMS permission.
* **R import sanity:** Only com.prismatica.iotInterface.R; no crashes from resource lookups.

**Pre-launch checklist:**

* VersionCode/VersionName updated in build.gradle.
* App icon & adaptive icon exported.
* All user-visible strings moved to res/values/strings.xml.
* DB schema stable (DB\_VERSION managed; no destructive upgrades).
* Build release AAB/APK with a keystore; verify install.
* Prepare listing text and screenshots.

**7. Release & Rollout**

**Packaging:** Build a signed **Android App Bundle** (Play preferred) or release APK.  
**Track:** Start with internal testing, then a small closed testing track, then staged production rollout (25% → 50% → 100%).  
**Crash/ANR monitoring:** Use Play Console vitals; fix any issues before widening rollout.

**8. Maintenance & Roadmap**

**Migrations:**

* Maintain DB\_VERSION; perform additive migrations (e.g., ALTER TABLE ... ADD COLUMN) to avoid data loss.
* Keep CRUD queries using column names (getColumnIndexOrThrow) to remain resilient to order changes.

**Bug triage:**

* Repro from user steps → logcat review → unit test or instrumentation test for repo methods → patch → bump versionCode.

**Potential future features:**

* CSV export/import; basic backup/restore; thresholds per item to auto-suggest SMS; light theming; tablet layout.

**Support & documentation:**

* README with quick start, SMS test instructions, and emulator tips (use a second emulator number like 5556 for SMS demo).