Step-by-Step Implementation Plan (with tech choices)

0) Baseline Decisions

- **Target OS: Android AOSP 13/14** (stable, modern APIs, good kiosk support).
- **Languages: Kotlin** (apps) + **C++ (NDK)** for PDF engine & ink performance.
- **UI Toolkit: Jetpack Compose** for most screens; **custom SurfaceView** for ultra-low-latency ink.
- **Build systems:**
  - **Apps:** Gradle (KTS).
  - **Platform:** Soong/Make (AOSP).
- **Repo layout (monorepo):**
- /aosp/ # AOSP source + device/vendor/overlay
- /apps/reader/ # PDF/EPUB app
- /apps/notes/ # Ink/notes app
- /apps/files/ # File Manager
- /apps/dictionary/ # Offline dictionary service+UI
- /apps/admin/ # Device Owner (DPC) + kiosk policies
- /apps/btshare/ # Bluetooth PDF share provider
- /shared/libs/pdf/ # Pdfium/MuPDF JNI bindings
- /shared/libs/ink/ # Stroke model, smoothing, storage

1) AOSP Image + Kiosk Foundation

1.1 Build a Custom AOSP Image

- Create device & vendor trees; include your core apps as **priv-app**.
- Remove/disable:
  - **Play Store, Browser, Email, Contacts, Dialer**, package installer UI.
  - **Settings pages** for Network/Wi-Fi (via **config overlays** + DPC restrictions).
- Pre-grant runtime permissions for your apps (priv-app permissions XML).

1.2 Device Owner (DPC) & Lock Task

- **App:** /apps/admin (Kotlin)
- **Core APIs:** DevicePolicyManager, Activity.startLockTask()
- **On first boot:** Provision **Device Owner** using Android Managed Provisioning (QR/NFC).
- **Enforce kiosk:**
- val dpm = getSystemService(DevicePolicyManager::class.java)
- val admin = ComponentName(this, AdminReceiver::class.java)
- dpm.setLockTaskPackages(admin, arrayOf(
- "com.weproz.reader",
- "com.weproz.notes",
- "com.weproz.files",

- "com.weproz.dictionary",
- "com.weproz.btshare",
- "com.weproz.settingslite"
-))
- dpm.setStatusBarDisabled(admin, true)
- dpm.addUserRestriction(admin, UserManager.DISALLOW_ADD_USER)
- dpm.addUserRestriction(admin, UserManager.DISALLOW_INSTALL_APPS)
- dpm.addUserRestriction(admin, UserManager.DISALLOW_INSTALL_UNKNOWN_SOURCES)
- dpm.addUserRestriction(admin, UserManager.DISALLOW_DEBUGGING_FEATURES)
- // Kiosk entry
- startLockTask()

1.3 **No Wi-Fi** Policy (Bluetooth allowed)

- In DPC:
- dpm.addUserRestriction(admin, UserManager.DISALLOW_CONFIG_WIFI)
- dpm.addUserRestriction(admin, UserManager.DISALLOW_CONFIG_MOBILE_NETWORKS)
- dpm.addUserRestriction(admin, UserManager.DISALLOW_CONFIG_VPN)
- AOSP **overlay**: hide Wi-Fi UI tiles/panels; optionally **disable network stack** service.
- (Optional hard kill) **iptables/eBPF** policy that drops all non-Bluetooth traffic at boot.

2) Reader App (PDF/EPUB)

2.1 PDF Technology

- **Engine: Pdfium** (preferred) or **MuPDF** via **NDK** JNI bindings for speed & memory.
- **Alternative (faster start):** AndroidPdfViewer (Pdfium-based) + custom patches.
- **Key APIs:** ParcelFileDescriptor, native rendering to Bitmap in worker threads.
- **Features to implement:** zoom, page jump, bookmarks, highlights (ink overlay), search, ToC, night/sepia, margins/spacing.

2.2 EPUB Technology

- **Render:** WebView + **epub.js** (offline assets).
- **Bridge:** addJavascriptInterface() to sync bookmarks/highlights/search with app DB.
- **Reflow:** enable custom CSS for font size/spacing/themes.

2.3 Storage

- **Metadata & annotations: Room (SQLite)**.
- **Documents:** app sandbox + **encrypted** export on request.

3) Notes App (Ink/Handwriting)

3.1 Input & Rendering

- **View:** Custom **SurfaceView** (hardware accelerated) for low latency.
- **Capture:** MotionEvent (TOOL_TYPE_STYLUS, pressure, tilt).

- **Model:** Vector strokes (sampled points), **Bezier**/Catmull-Rom smoothing, **pressure → width** curve.
- **Renderer:** OpenGL ES or Canvas (start with Canvas, upgrade to GL if needed).
- **Latency tactics:** Choreographer, off-UI thread raster, double buffering, partial invalidation.

3.2 Tools

- Pen, highlighter (blend mode multiply/alpha), eraser (stroke-level + pixel scrub), lasso (select/move/scale).
- **Templates:** ruled/dotted/grid/blank as vector layer; **infinite canvas** with tiled coordinates.

3.3 Export

- **PDF:** render vector ink to PDF canvas (PdfDocument) with downscaled backgrounds.
- **Image:** PNG/WebP with size limits; **target ≤5 MB** per 20 pages.

3.4 Storage

- **Notes & strokes:** Room/SQLite (binary blobs for strokes or protobuf).
- **Encryption:** per-notebook **AES-256-GCM** using **Android Keystore** key.

4) Offline Dictionary (Hindi & English)

4.1 Tech

- **Data:** Prebuilt dictionary packs (e.g., **StarDict** format or custom TSV → SQLite).
- **DB: SQLite** with **FTS5** tables for instant lookup; morphology tables for word forms.
- **Service:** Foreground-bound service that answers lookups from Reader/Notes via **AIDL** or **bound service**.

4.2 Integration

- Reader & Notes select text → send to Dictionary service → popup panel with definition/synonyms/pronunciation.

5) File Manager

- **Scoped to "Documents/E-Tablet"** root for simplicity.
- **Ops:** folder create/rename/copy/move/delete; bulk actions.
- **Encryption at rest:** user vault (EncryptedFile / SQLCipher if needed).
- **USB support:** SAF (ACTION_OPEN_DOCUMENT_TREE) for **pendrive** access.

6) Bluetooth-Only PDF Sharing

6.1 Custom Share Provider (system app)

- **App:** /apps/btshare

- **Intent filter:** accept **only** ACTION_SEND with type="application/pdf".
- **Send flow:** target **system Bluetooth** package explicitly.
- fun sharePdf(uri: Uri) {
- val i = Intent(Intent.ACTION_SEND).apply {
- type = "application/pdf"
- putExtra(Intent.EXTRA_STREAM, uri)
- // Restrict to system Bluetooth
- setPackage("com.android.bluetooth")
- addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION)
- }
- startActivity(i)
- }
- **Block others:** In AOSP, remove share targets; in DPC, **intent resolution** policy so only com.weproz.btshare is exported.

6.2 Receive side

- Use stock **Bluetooth OPP** (Object Push) receiver (system).
- Optionally add a **receiver app** to auto-import PDFs into library.

7) Security & Privacy

7.1 Device Security

- **Unlock:** PIN/password/gesture; auto-lock after idle.
- **Owner restrictions:** block USB debugging, app installs, users/profiles.

7.2 Data Security

- **FS encryption:** AOSP FBE.
- **App encryption: AES-256-GCM** via **Android Keystore**:
- val keyGen = KeyGenerator.getInstance(KeyProperties.KEY_ALGORITHM_AES, "AndroidKeyStore")
- keyGen.init(
- KeyGenParameterSpec.Builder("notes_key",
- PURPOSE_ENCRYPT or PURPOSE_DECRYPT)
- .setBlockModes(KeyProperties.BLOCK_MODE_GCM)
- .setEncryptionPaddings(KeyProperties.ENCRYPTION_PADDING_NONE)
- .build()
-)
- val key = keyGen.generateKey()
- **Prefs:** Jetpack **DataStore** (proto) for robust, transactional settings.

7.3 Privacy

- No analytics SDKs.
- Local privacy policy HTML stored offline.

8) Accessibility & Localization

- **TalkBack:** content descriptions, focus order, semantic actions.
- **Font scaling:** sp units; respect Configuration.fontScale.
- **High contrast & dark mode** themes.
- **Multilingual UI:** English + regional languages; embed **Noto** fonts for crisp rendering.

9) Performance Targets (and how)

- **Launch < 2s:** splash → lazy load heavy engines.
- **PDF page turn ≤50 ms:** pool bitmaps; prefetch next/prev pages.
- **Ink latency ≤20–40 ms:** SurfaceView + off-thread raster; avoid allocations per frame.
- **Battery:** throttle background work with **WorkManager**; do on-demand parsing/caching.

10) Provisioning & Manufacturing

10.1 Device Owner Provisioning

- Use **QR provisioning** at first boot with **Managed Provisioning**:
  - Whitelisted packages (your app IDs)
  - Disable Wi-Fi config
  - Set home/launcher to your **Reader** (or minimalist launcher)

10.2 Boot Customization

- **Boot animation/logo** overlays (admin-customizable) per your SRS.
- **Wallpaper preload** (admin-only change).

10.3 Updates (offline)

- **Recovery OTA via USB (pendrive)**: signed package; admin menu triggers update mode.

11) CI/CD & Quality

- **CI:** GitLab/Jenkins runners with Gradle caches; artifact signing.
- **Static analysis: Detekt**, **ktlint**, **Android Lint**; **clang-tidy** for NDK.
- **Tests:**
  - Unit (JUnit + MockK)
  - Instrumented (Espresso/UI Automator)
  - Soak tests for battery & memory.
- **Performance:** Macrobenchmark; perfetto traces on target tablet.

12) Acceptance Checklist (tie to SRS)

- Kiosk lock: cannot exit apps; status bar off; settings restricted.
- Wi-Fi: not configurable; device never connects.
- Reader: open 300-page PDF ≤2s; highlight/bookmark persist; search works offline.

- Notes: stroke latency ≤40ms; undo/redo; export 20p ≤5MB.
- Dictionary: Hindi/English lookup ≤300ms offline.
- Bluetooth share: **PDF only**; non-PDF blocked with message.
- Encryption: files unreadable off-device; key in Keystore.
- Accessibility: TalkBack navigable, fonts scale.

Tech Choices Summary (at a glance)

Area	Tech / Library
OS	**Android AOSP 13/14**
Kiosk	**DevicePolicyManager**, Lock Task, overlays
Apps	**Kotlin**, Jetpack **Compose** (+ SurfaceView for ink)
Native	**C++ (NDK)** for Pdfium/MuPDF & high-perf ink
PDF	**Pdfium** (JNI) or MuPDF
EPUB	**WebView + epub.js**
Ink	Custom SurfaceView, Bezier smoothing, OpenGL/Canvas
DB	**Room/SQLite**, **FTS5** for dictionary
Crypto	**Android Keystore**, **AES-256-GCM**, **EncryptedFile**
Sharing	**Bluetooth OPP** via explicit **ACTION_SEND** to com.android.bluetooth
Settings	**DataStore (Proto)**
Testing	JUnit, Espresso, Macrobenchmark, Perfetto
CI/CD	GitLab/Jenkins + Gradle; AOSP Soong builds

Suggested Development Timeline (8–9 weeks)

- **Week 1:** AOSP device image, overlays, remove bloat, DPC scaffold.
- **Week 2:** Device Owner provisioning, kiosk + restrictions, hide Wi-Fi.

- **Week 3–4:** Reader (PDF/EPUB engines, navigation, search, highlights).
- **Week 4–5:** Notes (ink pipeline, tools, export).
- **Week 5:** Dictionary (DB + service + UI).
- **Week 6:** File Manager + Bluetooth PDF share provider.
- **Week 7:** Encryption, DP customization, accessibility, localization.
- **Week 8:** Optimization, acceptance tests, manufacturing provisioning, USB OTA.