Handz V1 PRD Bundle

CH38 — Development Milestones & Release Checklist

This chapter defines the build milestones, definition-of-done gates, and the complete release checklist to ship Handz V1 to the Apple App Store with minimal guesswork. It is intentionally operational: it tells you exactly what must exist, what must be verified, and what artifacts must be produced.

Doc ID:	HZ-V1-CH38_Development_Milestones_Release_Checklist_R1
Revision:	R1 (2026-01-02)
Status:	Draft (Ready for review)
Depends on:	CH00 (Manifest), CH35 (QA Acceptance Tests), CH25 (Monetization), CH07 (Auth), CH08 (Plan State
Related:	CH04 (Navigation), CH05 (Screen Inventory), CH06 (Design System), CH12–CH22 (Core Product), C
Supersedes:	—
Owned Decisions:	Milestone definitions, readiness gates, build order, App Store submission artifacts, release rollback pl
Open Questions / Placeholders:	Exact target dates, exact App Store metadata copy, final pricing validation checkpoints, legal policy U

1. Scope of this chapter

CH38 is not a feature spec. It is the operational plan that turns the feature specs (CH01–CH37) into a shippable iOS app. It defines: (a) milestone stages and what must be complete at each stage, (b) the build order, (c) a release checklist that includes App Store compliance, and (d) rollback and post-launch monitoring.

2. Milestone model

Milestones are gates. A gate is passed only when all acceptance criteria for the gate are met. If a gate fails, the project returns to the previous gate and fixes the failing criteria (do not pile new features on top of a failing gate).

Stage	Purpose	Exit criteria (summary)
M0 — Repo & Tooling Ready	A buildable skeleton exists.	App compiles on device; navigation shell; CI runs; crash reporting wire
M1 — Core Data Ready	Moves/Flows data model exists.	Auth + plan states; create/edit moves; create/edit flows (draft); offline d
M2 — Flow Builder Ready	Users can build real flows.	Pan/zoom; branching up to 10; reorder; sequence details; save/load; fr
M3 — Practice Ready (Paywalled)	Practice sessions function end-to-end.	Setup; active session; interruptions; logging; credits; view-only for inbox
M4 — Sharing + Inbox Ready	Growth loop works without contradictions.	Unlisted links; inbox cap; import conflict rules; save-to-library gating.
M5 — Monetization + Paywalls Ready	Store end-to-end works.	Trial; $9.99; restore purchases; entitlement sync; upsell surfaces; recei
M6 — App Store Ready (RC)	Submission package ready.	Privacy/compliance; performance; accessibility baseline; QA pass; mar

M7 — Launch + Monitoring	Ship + observe.	Production rollout plan; dashboards; incident playbook; hotfix pipeline.

3. Build order by dependency

This is the recommended build order for a first-time implementation using vibe coding. Each step references the owning PRD chapter(s). Do not skip gates; skipping creates hidden contradictions that surface later during App Review or monetization testing.

- **Step 0 — Bundle setup:** Attach CH00 + CH38 to the build environment. Create a single source-of-truth folder for PRD PDFs (e.g., **/prd**).
- **Step 1 — App shell + navigation:** Implement IA + navigation routes + tab bar + modal patterns. (See: CH04, CH05, CH06)
- **Step 2 — Auth + plan states:** Apple/Google/Email sign-in; Guest limitations; entitlement model (Guest/Free/Trial/Pro). (See: CH07, CH08, CH25)
- **Step 3 — Core data models:** Moves, flows, nodes/edges, sequences, paths, gameplans. Implement CRUD with local-first drafts + sync. (See: CH09–CH14, CH28, CH29)
- **Step 4 — Flow builder:** Pan/zoom, sideways layout, reorder, branching up to 10, merges, dangling paths allowed, replace-any-node, sequence editor entry. (See: CH12–CH14)
- **Step 5 — Library + detail views:** Folders; flow-only search; flow detail; export entry points; saved-flow cap logic. (See: CH15–CH16, CH08)
- **Step 6 — Sharing + inbox:** Unlisted link creation; inbox receiving; inbox cap; view-only for free inbox items; save-to-library gating; import conflict resolution UI. (See: CH17–CH19)
- **Step 7 — Practice mode:** Setup, active session, interruptions, logging, history. Enforce: paywall + monthly credits usable only on saved flows. (See: CH20–CH22, CH25)
- **Step 8 — Mastery + maintenance:** Gameplan selection; maintenance scheduling; overload prevention; notifications; trust mechanics. (See: CH23–CH24, CH27)
- **Step 9 — Safety/abuse + warnings:** Warning ladder; soft caps; messaging; enforcement surfaces. (See: CH30)
- **Step 10 — Error states + offline:** Global empty states; non-blocking vs blocking; offline drafts; sync conflicts; retry rules. (See: CH28, CH31)
- **Step 11 — Monetization hardening:** StoreKit trial; purchase; restore; receipt validation; paywall placements; downgrade behavior. (See: CH25)
- **Step 12 — QA + release prep:** Execute CH35 tests; accessibility baseline; export/deletion; privacy. (See: CH32–CH36)

4. Milestone gates (detailed)

Each gate below includes: required artifacts, verification steps, and a strict pass/fail checklist.

M0 — Repo & Tooling Ready

Goal: the project can be built, run, and iterated quickly without hidden setup work.

- **Repo structure**: /app (source), /docs (developer notes), /prd (PDFs), /scripts (build helpers).
- **Environment**: iOS simulator build succeeds; physical device build succeeds.

- **Navigation shell**: Tab scaffolding exists (even if screens are placeholders).
- **Remote config toggle**: a single kill-switch flag exists to disable Practice Mode server-side in case of incidents (see CH27/CH25 cross-ref).
- **Crash reporting**: crash logs captured (tool choice owned outside PRD; must exist).
- **Analytics stub**: event logger wrapper exists with no-op in dev mode (see CH33).
- **App icon + launch screen placeholders**: use the approved Handz logo assets; final polish later in M6.

M1 — Core Data Ready

Goal: a signed-in user can create/edit core objects reliably (moves + flows) and the data persists correctly under real-world conditions.

- **Auth**: Apple + Google + Email sign-in works; session persists across app relaunch. (See: CH07)
- **Plan states**: Guest/Free/Trial/Pro state is always determinable and exposed via a single helper (no duplicated gating logic). (See: CH08)
- **Guest restrictions**: Guests can browse but cannot save flows even locally; before building any flow, the app clearly states saving requires an account. (See: CH07/CH08)
- **Moves**: default move pack loads; user can add custom moves; aliases/families/variants framework exists (even if not fully populated). (See: CH09–CH11)
- **Flows**: user can create a draft flow; edit title; add/remove nodes; discard draft; save to library (account required). (See: CH12–CH16)
- **Free caps enforced**: Free users cannot exceed 2 saved flows; UI explains the cap and offers upgrade pathways. (See: CH08, CH25)
- **Offline behavior defined**: if network drops mid-edit, changes are preserved locally and sync later; user sees clear status. (See: CH28)

M2 — Flow Builder Ready

Goal: users can build complex decision trees without the UI fighting them, and without corrupting data.

- **Pan/zoom** works smoothly; sideways layout supports both top-to-bottom and left-to-right mental models. (See: CH12)
- **Branching**: any move node supports up to 10 outgoing branches; branches can branch; merges can have multiple incoming paths; dangling paths allowed. (See: CH12)
- **Reordering**: user can reorder nodes/paths; replace root; replace any move; preserve edges when sensible; prompt when destructive. (See: CH12/CH13)
- **Sequence detail**: optional sequence node exists; summary node option if chosen; details editor saves reliably. (See: CH14)
- **Validation**: prevent impossible states (e.g., edge without target); if state becomes invalid, show repair UI rather than silent failure. (See: CH31)
- **Performance**: stress test with a large flow (e.g., 75+ nodes, many branches) without crashing; frame drops acceptable but interaction remains usable.
- **Autosave**: drafts autosave; undo/redo policy (if present) is consistent and tested.

M3 — Practice Ready (Paywalled)

Goal: practice is reliable, motivating, and correctly gated by plan state.

- **Setup**: user selects paths across flows, orders them, sets timers and assumed reps. (See: CH20)
- **Eligibility**: Practice requires Pro/Trial OR uses monthly credits; credits usable only on saved flows (not inbox items). (See: CH25)
- **Active session**: timer, pause, early end (follow workout-tracker norms), save as interrupted when appropriate. (See: CH21)
- **Logging**: actual duration recorded; per-path completion counts recorded; streak logic uses local time and one completed set counts as a day. (See: CH22)
- **History**: user can view past sessions; interruptions are distinguishable from completes. (See: CH22)
- **Free restrictions**: Free users cannot practice flows that are only in the inbox; they can view only. (See: CH18)

M4 — Sharing + Inbox Ready

Goal: sharing drives growth without allowing free users to bypass the business model.

- **Unlisted share links**: links can be created/revoked; view-only experience works; link lifecycle rules enforced. (See: CH17)
- **Inbox**: free inbox cap is 10; imports above cap show a clear block and CTA. (See: CH18)
- **Save-to-library flow**: saving an import requires account + respects saved-flow cap; when cap is hit, user must delete or upgrade. (See: CH08, CH18, CH25)
- **Import conflicts**: missing moves/custom payloads trigger conflict resolution; receiving user chooses how to handle custom data (keep in-template only vs add to library). (See: CH19)
- **Contradiction check**: importing many flows cannot grant unlimited practice for free (practice remains paywalled and/or credits-limited). (See: CH25)

M5 — Monetization + Paywalls Ready

Goal: purchases are robust enough for App Review and real money, including restore and downgrade behavior.

- **StoreKit**: $9.99/month subscription, 7-day trial, managed via StoreKit subscriptions. (See: CH25 lock in CH00)
- **Restore purchases**: works reliably; user can recover Pro after reinstall/new device.
- **Entitlements**: trial behaves as Pro; gating is consistent across app (single source of truth). (See: CH08)
- **Paywalls**: placements implemented per CH25; upsell copy matches CH26 claims pages and includes results-may-vary language. (See: CH25, CH26)
- **Downgrade**: when Pro ends, user keeps data but loses gated functionality; UX clearly explains what changes. (See: CH08, CH25)
- **Receipt validation**: minimal validation approach exists; failures degrade gracefully (no user lockout due to transient network).

M6 — App Store Ready (Release Candidate)

Goal: pass App Review and deliver a stable first impression on Day 1.

- **QA pass**: execute CH35 scripts on latest iOS version(s); all critical tests pass; known issues logged with severity. (See: CH35)
- **Performance**: cold start acceptable; no frequent crashes; memory stable on large flows.
- **Accessibility baseline**: tap targets, dynamic type where feasible, VoiceOver labels for critical controls. (See: CH32)

- **Privacy + compliance**: privacy policy links; account deletion; data export where promised; tracking disclosures match analytics. (See: CH33, CH34)
- **App Store assets**: icon (1024px), screenshots, preview video (optional), subtitle, keywords, description, support URL.
- **Legal**: Terms + Privacy; user-generated content rules (share links) documented; reporting route if required.
- **Support**: in-app 'Contact support' path or email; troubleshooting doc (CH36) published.
- **Safety**: warning ladder implemented; abuse limits enforced with soft warnings. (See: CH30)

M7 — Launch + Monitoring

Goal: launch is controlled, measurable, and recoverable.

- **Monitoring dashboard**: crashes, purchase failures, share-link errors, sync conflicts, onboarding funnel metrics. (See: CH33)
- **Incident playbook**: triage steps, severity levels, rollback plan (kill-switch for Practice), and hotfix release procedure. (See: CH36)
- **Release notes**: v1.0 notes prepared; known limitations stated (e.g., iOS-only, portrait-only). (See: CH02 locks)
- **Post-launch cadence**: patch window (48–72h) reserved; backlog grooming for v1.0.1.

5. Apple App Store submission checklist (step-by-step)

This checklist is written to be executed in order. Treat it like a flight checklist: if you cannot check an item, stop and fix the underlying issue.

5.1 Pre-submission artifacts

- **Bundle identifiers** are finalized (App name, bundle id, versioning scheme).
- **App icon** is final (1024x1024) and matches iOS icon set; no alpha channel; readable at small size.
- **Screenshot set** exists for required device sizes; each screenshot corresponds to a real in-app screen and matches the current UI.
- **Support URL** is live and describes: account, restoring purchases, sharing, practice, common errors.
- **Privacy policy URL** is live and matches the app behavior (data collected, retention, deletion).
- **Test account** credentials exist for App Review if any gated content cannot be accessed without sign-in.
- **In-app purchases** (subscription + trial) are created in App Store Connect and linked to the build.

5.2 App Store Connect metadata (fill-in list)

- **App name**: Handz
- **Subtitle**: (locked in CH00/branding chapter; update here when finalized)
- **Keywords**: set of 100 characters; avoid competitor trademarks; include striking-relevant terms + 'combo' + 'drills' + 'gameplan'.
- **Description**: first 3 lines must explain benefit (faster decisions, smarter sparring); then features; then safety + results-may-vary.
- **Promotional text**: short value prop (optional).
- **Category**: Health & Fitness (or Sports) — finalize before submission.
- **Age rating**: set per content; include UGC (shared flows) considerations.
- **Privacy nutrition label**: must match actual SDK usage; if analytics/crash tools collect identifiers, declare it.
- **App review notes**: explain Guest limitations; how to access demo flows; how paywall works; provide test account.

5.3 Build validation before upload

- **Version numbers**: increment build number; ensure marketing version matches intended release (e.g., 1.0.0).
- **Clean install test**: install on a fresh device/simulator; confirm no stuck onboarding loops; confirm guest restrictions messaging.
- **Purchase test**: test trial start, cancel, renew; test restore; test offline purchase edge-cases (defer gracefully).
- **Core flow tests**: create move; build flow with branches; save; share link; import to inbox; attempt to practice (gated).
- **Limits tests**: free saved flows capped at 2; inbox capped at 10; video storage capped at 2GB; soft warnings trigger as designed.
- **Deletion/export**: delete account path reachable; data export works (if included).
- **Crash sweep**: run through major flows; check crash logs are empty.

5.4 Submission + review workflow

- **Upload build** via Xcode/Transporter; confirm processing passes.
- **Attach build** to the app version and to the subscription IAP if required.
- **Complete compliance questions** (encryption, tracking, etc.).
- **Submit for review**; record submission timestamp and build number.
- **During review**: if rejected, capture rejection reason verbatim, map to owning chapter, patch, and resubmit with clear notes.

6. Release readiness: regression set (minimum)

This is the minimum regression set to run for every release candidate. It is not a replacement for the full CH35 test suite; it is the fast pass to catch catastrophic regressions.

- **Auth**: sign in with Apple; sign out; sign back in; session persists after relaunch.
- **Guest**: guest can browse demo; cannot save; sees account CTA before building.
- **Flows**: create a flow; add 3 nodes; add 2 branches; add a merge; save; reopen; edit; delete.
- **Caps**: free user can save 2 flows; third save triggers upgrade path; cannot bypass by inbox imports.
- **Sharing**: generate an unlisted link; open it on another device; import to inbox; confirm inbox count increments; cap at 10 enforced.
- **Practice**: Pro user starts practice; pauses; resumes; ends early; sees summary; history updated; streak updates correctly.
- **Paywall**: free user taps Practice → sees paywall; trial start works; entitlement updates instantly.
- **Offline**: turn on airplane mode during flow edit; confirm local draft persists; on reconnect, sync completes without data loss.
- **Crash**: no crashes in above flows; if any occur, fix before shipping.

7. Rollback & hotfix plan

Handz V1 should be recoverable from bad releases without relying on users to update immediately.

- **Remote kill-switch**: ability to disable Practice Mode entry points (and optionally sharing) if severe issues are detected.
- **Server-side gating**: plan state decisions must be server-authoritative where possible to prevent entitlement mismatches.
- **Hotfix pipeline**: ability to cut a patch build within 24 hours; a predefined minimal regression set (Section 6) must be executed before upload.
- **Data safety**: never run destructive migrations automatically on client start; use backwards-compatible schema evolution for V1.x.
- **User messaging**: in-app banner for known issues + mitigation steps (if needed).

8. Risks & mitigation (V1-specific)

- **Complex flow builder bugs**: risk of corrupted graphs. Mitigation: validate graph invariants on every save; provide repair UI; autosave versions.
- **Monetization regressions**: StoreKit edge-cases and App Review rejection. Mitigation: run purchase tests on every RC; keep paywall copy consistent; implement restore prominently.
- **UGC safety**: imported flows containing unexpected custom data. Mitigation: sanitize inputs; cap payload sizes; use inbox cap; soft warnings ladder.
- **Offline sync conflicts**: duplicated gating logic leads to inconsistent states. Mitigation: single plan-state helper; deterministic conflict resolution rules.
- **Performance**: large graphs cause lag. Mitigation: virtualization, memoization, and a max-render strategy; warn user when graph gets heavy.

9. Replit build prompt for this chapter (PROMPT 38)

Use this prompt inside Replit (or any vibe-coding environment) **after** the core product prompts have produced a runnable app. This prompt adds the release-readiness tooling and the in-repo operational docs so the build can be shipped with fewer surprises.

```
You are building the iOS-only (portrait) app "Handz" using the attached PRD bundle.
First, read CH00 (Master Index Manifest) and CH38 (Development Milestones & Release
Checklist).
Then do the following WITHOUT asking me questions unless a step is impossible:

A) Create a /docs/release folder with:
- RELEASE_CHECKLIST.md (copy the CH38 Section 5 checklist verbatim, formatted as
checkboxes)
- REGRESSION_MINIMUM.md (copy CH38 Section 6 verbatim)
- ROLLBACK_PLAYBOOK.md (copy CH38 Section 7 verbatim)
- APP_STORE_METADATA_TEMPLATE.md (copy CH38 Section 5.2 fill-in list with placeholders)

B) Create /scripts with:
- verify_limits.ts (or equivalent language): a small script that validates the configured
hard locks and caps at runtime:
- Free saved flows cap = 2
- Free inbox cap = 10
- Video storage cap = 2GB
- Practice credits cannot be used on inbox flows
The script should fail CI if these values differ from the configured constants.

- smoke_test_checklist.md: a command-line runnable checklist (or a simple JSON) that
lists the minimum regression steps.

C) Add an in-app hidden "Release Readiness" debug panel (dev builds only) that:
- Shows current plan state (Guest/Free/Trial/Pro)
- Shows caps (flows/inbox/video storage) and current counts
- Runs a lightweight graph invariant check on the currently open flow (no orphan edges,
no cycles if disallowed, etc.)
- Provides a "copy diagnostics" button that copies device/app version + recent errors to
clipboard.

D) Add CI (if not present) that:
- Builds the iOS project
- Runs unit tests (at least for gating/limits helpers)
- Runs verify_limits.* so caps cannot silently drift.

E) Output a short "Milestone status" report mapping current repo state to CH38 gates
M0–M7, listing what is done and what is missing.
```

10. Troubleshooting & diagnostics (operational)

This section is designed for non-engineers to identify what is broken, where it likely lives, and what to try first. It does not replace debugging, but it reduces guesswork.

- **App won't launch / crashes immediately**: check M0 prerequisites; disable optional modules via feature flags; inspect last crash log; confirm no missing environment variables (e.g., API keys).
- **Users report 'lost flows'**: verify offline/sync model (CH28); check whether user was a Guest (cannot save); verify account state; inspect conflict resolution logs.

- **Free users can practice without paying**: regression in plan-state helper (CH08/CH25). Run verify_limits script; confirm practice entry points always call gating helper.
- **Paywall purchase fails**: confirm StoreKit product IDs match App Store Connect; confirm app build is signed correctly; test restore; check receipt parsing; provide 'Try again' path.
- **Share links open but import fails**: check link payload size limits; check move-family ID mapping; confirm inbox cap; check server endpoint/serialization compatibility.
- **Flow builder behaves weird on large flows**: performance mitigation (virtualization); validate graph invariants; check pan/zoom gesture conflicts; confirm hit targets.
- **App Review rejection**: map rejection text to checklist item (privacy, IAP, account deletion). Patch the owning chapter area, update review notes, resubmit.

11. Change management (keeping the PRD editable)

Because Handz will evolve, this PRD bundle must remain versioned and modular. To avoid 'locked in' architecture mistakes:

- Every chapter PDF is versioned (R1, R2, ...). Changes must include: what changed, why, and downstream chapters impacted.
- All constants that represent locked decisions (caps, pricing, gating) live in one file in code and are referenced everywhere else.
- Whenever a cap or pricing changes, update: CH00 locks, CH25 monetization, and CH38 verify_limits script + release checklist.
- Additive updates (new features) should be appended as new chapters where possible rather than rewriting old ones, unless the old spec is now wrong.

Appendix A — Gate sign-off template

Use this template each time a milestone gate is passed. It creates a paper trail for what was tested and prevents 'we thought it worked' failures.

- **Gate**: M__
- **Date**: ____
- **Build number**: ____
- **Tester(s)**: ____
- **What was tested**: link to checklist or run log
- **Known issues**: list + severity + owner
- **Decision**: PASS / FAIL (circle one)
- **Notes**: ____