Handz V1 PRD Bundle

CH02 — Non-Goals, Assumptions, and Constraints

Doc ID: HZ-V1-CH02_Non_Goals_Assumptions_Constraints_R1

Revision: R1 (2026-01-02)

Status: Draft (Review)

Depends on: CH00 (Master Index & Manifest), CH01 (Product Definition & V1 Scope)

Related: CH04 (Navigation Map), CH07 (Auth & Account), CH08 (Entitlements), CH12 (Flow Builder), CH20–CH22 (Practice), CH25 (Monetization), CH28–CH31 (Offline/Sync, Data Limits, Safety, Errors)

Supersedes: None (first revision)

Owned Decisions (locked by this chapter): V1 platform/form-factor constraints; V1 non-goals list; explicit assumptions; constraints catalog; enforcement expectations.

Read-me first: This chapter prevents "scope creep by accident." It defines what V1 explicitly does *not* do, what we assume to be true, and what constraints must be enforced in UI, logic, and release readiness. If something contradicts this chapter, update this chapter (new revision) and cross-reference the change in CH00.

1. Purpose of CH02

CH02 exists to remove guesswork for builders and to reduce bugs created by ambiguous expectations. It defines:

- Non-goals: features, platforms, and behaviors explicitly out of scope for V1.
- Assumptions: what we presume about users, devices, and usage so we can make consistent product decisions.
- Constraints: hard boundaries that must be enforced (not just documented).

Where a constraint depends on another chapter's detailed behavior, this chapter references it rather than re-defining it.

2. V1 Non-Goals (Explicitly Out of Scope)

Non-goals are not "maybe later" in an undefined way; they are intentionally excluded so V1 can ship. If you implement a non-goal, you are building V2+ by accident.

2.1 Platforms & form factors

- **No Android build** in V1. (iOS-only for first ship.)
- **No landscape orientation**. (Portrait-only.)

- **No iPad/tablet optimization**. iPad may run in compatibility mode, but no tablet layouts are designed or supported.
- **No web app** as a product surface (no PWA). Share links may render a web view-only page later (owned by CH17), but V1 does not ship a full web client.

2.2 Social/Community features

- No public community feeds, comments, likes, or follower graph.
- No gym codes, team spaces, or multi-user shared workspaces.
- No public searchable user profiles or public libraries.
- No in-app messaging / DMs.
- No community moderation tooling (reporting queues, admin panels).

2.3 AI / automation features

- No automatic move extraction from video, no pose detection, no auto-tagging from camera footage.
- No opponent video analysis or scouting automation (tape breakdown automation is out of scope).
- No auto-generated flows/gameplans from prompts in V1 (manual creation and structured templates only).

2.4 Advanced content authoring

- No embedded technique encyclopedia. V1 ships **without** authoritative technique descriptions to avoid disagreements across gyms/styles.
- No required sequence nodes between moves (optional later). V1 can support optional transition detail in the sequence editor (owned by CH14), but sequences are not mandatory.
- No custom scripting language for conditions ("if opponent does X") beyond plain-language labels.

2.5 High-end collaboration & export

- No real-time collaborative editing (no "two people editing same flow" live).
- No full library sharing in V1 (sharing focuses on unlisted links for flows/gameplans, owned by CH17–CH19).
- No complex export formats (e.g., video montage export, Notion sync, Google Drive integration).

2.6 Hardware / wearable integrations

- No Apple Watch app.
- No sensor integrations (heart rate straps, accelerometers for strike detection).

3. V1 Assumptions (What We Presume is True)

Assumptions reduce ambiguity. If an assumption proves false in real usage, document it as an issue and revise CH02.

3.1 User & usage assumptions

- Most users are strikers (boxing/kickboxing/Muay Thai/karate/TKD hybrids). Flows can mix arts, but the product positioning is striking-first.
- Users want both extremes: quick simple combo notes *and* obsessive detail (progressive disclosure).
- Typical sessions: users may spend up to ~2 hours building/gameplanning (long editing sessions are normal).
- Practice sessions may happen with gloves on, so V1 should avoid requiring frequent precise tapping mid-session.
- Users are willing to sign up to save work; guest mode is for exploration, not for long-term storage.

3.2 Vocabulary assumptions

- Many users do not know what a "flow" is on first launch. Onboarding must explain the concept in plain language (owned by CH15 Copy Pack).
- Move naming varies by gym/style; aliases matter and small differences matter to some users.
- Users may want both names present (e.g., teep and push kick) and will self-select what matches their training.

3.3 Device & environment assumptions

- Users have an iPhone capable of running the minimum supported iOS version (placeholder; see §6).
- Users have intermittent internet. Core authoring should not catastrophically fail on brief disconnects (owned by CH28).
- Users may be in gyms with poor reception; practice mode should not require network calls during an active session (owned by CH20–CH22 and CH28).
- Users may deny camera or photo library permissions; V1 must gracefully degrade (links still work; upload is optional and Pro-only).

3.4 Business assumptions

- Monetization uses StoreKit subscriptions (iOS-first).
- Free plan exists to drive adoption, but meaningful value is paywalled via Practice + higher caps (owned by CH25).
- Unlisted sharing is a growth loop. Share links are intended to spread flows without making the app a social network (owned by CH17).

4. V1 Constraints Catalog (Must Be Enforced)

Constraints are not aspirational. Builders must implement them as runtime logic and UI behavior. Where the enforcement detail lives in another chapter, this section references it.

4.1 Platform & layout constraints (hard)

- **iOS-only.** All UI/logic assumes iOS navigation patterns and StoreKit monetization.
- **Portrait-only.** The app must not rotate into landscape. If iPad rotates, app should remain portrait or letterboxed (implementation choice, but V1 does not design landscape layouts).
- **One-handed use bias.** Primary actions should live in reachable zones when possible (bottom actions, large hit targets).

4.2 Account & guest constraints (hard)

- **Saving flows requires an account.** Guests cannot save flows locally or in cloud. If a guest attempts to start building, they must be warned up front (owned by CH07 and CH15).
- **Guest capabilities are limited.** Guests can browse demo content and interact with the UI to understand the product, but anything that creates durable value (saving, practicing saved flows, exporting, etc.) requires sign-up.
- **Login methods:** Apple, Google, and Email sign-up must be supported in V1 (owned by CH07).

4.3 Entitlements & caps (hard)

These are global locks already recorded in CH00. This chapter restates them only as constraints, not as full monetization spec.

- Plan states: Guest / Free / Pro / Trial (Trial behaves as Pro).
- Practice is paywalled. Free receives monthly practice credits usable only on saved flows.
- Free saved flows cap: 2.
- Free inbox cap: 10; free can view but cannot practice inbox items.
- Branching: up to 10 outgoing branches from a move.
- Uploads: Pro-only; total cap 2GB; uploads are private-only and not shared; links are shareable.

4.4 Content model constraints (hard)

- **No authoritative technique descriptions shipped.** Default moves ship as names + tags/families; users add their own notes/videos if desired (owned by CH09–CH11).
- **Aliases can be separate canon moves** when community uses both terms and nuance may matter (e.g., teep vs push kick).
- **Moves can belong to multiple tags**; flows can mix arts (owned by CH09–CH11).

4.5 Flow builder interaction constraints (hard)

- Builder must support sideways flow with pan/zoom to handle large branching trees.
- A single move can branch to up to 10 next moves; branches can branch recursively; merges can have multiple incoming paths.

- Dangling paths are allowed (a node may exist without a next step).
- Root move must be replaceable; any node should be replaceable (owned by CH12–CH13).

4.6 Practice constraints (hard)

- Practice must support selecting paths from different flowcharts and ordering them before the session.
- Practice uses **actual duration** logging and supports interruptions saved as interrupted.
- Practice must be usable without touching the phone constantly; controls are minimal and glove-friendly (owned by CH20–CH22).

4.7 Sharing constraints (hard)

- Sharing is unlisted-only in V1 (no public directory).
- If a flow includes uploaded (local) videos, receiving users must be explicitly told those uploads do not travel with the share; only links are shareable (owned by CH17–CH19).
- Importing must not accidentally overwrite a recipient's canonical move definitions without explicit choice (owned by CH19).

4.8 Data, privacy, and safety constraints (hard)

- Data must be exportable and account deletable in an App Store compliant way (owned by CH34).
- Safety/abuse escalation uses the warning ladder (owned by CH30).
- Share links must be revocable; anti-abuse thresholds apply even to Pro (owned by CH17 and CH30).
- No health/medical claims in product UX without controlled language and disclaimers (owned by CH26).

5. Enforcement Expectations (How Builders Should Implement Constraints)

This section defines what "enforced" means. The goal is to prevent accidental loopholes that would undermine monetization, safety, or user trust.

5.1 Constraint enforcement patterns

- **Centralized config:** Define constraints and caps in one place (constants + feature-gate helpers). Do not hardcode caps in UI components.
- **Server-side validation:** Even if UI blocks an action, the backend must also reject it (e.g., creating a 3rd saved flow on Free).
- **Clear user messaging:** When blocking, show the reason + the next step (upgrade, delete, sign up).
- **Consistent copy tokens:** Use shared copy strings so the same rule is phrased identically across the app (owned by CH15).

5.2 Examples (minimum enforcement)

- If guest taps "Save Flow" -> show sign-up gate, do not create a local draft that survives app close.
- If free user tries to save flow #3 -> block and route to "Manage saved flows" with delete/upgrade options.
- If free user opens inbox item -> allow view; hide/disable Practice button with paywall messaging.
- If user rotates device -> UI remains portrait; no broken layout.
- If user creates branch #11 from a node -> block and show soft warning ladder message (CH30).

6. Placeholders Owned by CH02 (Must Be Decided Later)

These are unknowns that impact build decisions. Leave them as placeholders in implementation until resolved, but do not silently assume.

- **PLACEHOLDER: Minimum iOS version** • Owner: CH02 • Options: iOS 16 / iOS 17 / iOS 18 • Default: iOS 17 (temporary) • Decide-by: before first TestFlight.
- **PLACEHOLDER: iPad behavior** • Owner: CH02 • Options: allow in compatibility / explicitly block iPad / support iPad portrait only • Default: allow compatibility (temporary).
- **PLACEHOLDER: Localization** • Owner: CH02 • Options: English-only / English + Spanish • Default: English-only (temporary).
- **PLACEHOLDER: Accessibility baseline** • Owner: CH02 • Options: minimum WCAG-inspired checks vs full accessibility pass • Default: baseline (temporary) • Details owned by CH32.
- **PLACEHOLDER: Offline authoring guarantees** • Owner: CH02 • Options: no offline / drafts-only / full offline-first • Default: drafts-only (temporary; see CH28).

7. Acceptance Tests (Chapter-Specific)

Write tests as Given/When/Then. These are minimum tests required before V1 ship readiness. More exhaustive tests live in CH35.

- **Orientation lock**: Given the app is open, when the device rotates to landscape, then the UI remains portrait and no screen content becomes clipped or unusable.
- **Guest cannot save**: Given a guest user, when they attempt to save a flow, then the app blocks saving, shows the sign-up gate copy, and no saved flow is created locally or remotely.
- **Free saved flow cap**: Given a free user with 2 saved flows, when they attempt to save another flow, then the app blocks, explains the cap, and routes to manage/delete/upgrade.
- **Inbox view-only on Free**: Given a free user viewing an inbox item, when they attempt to start practice, then the Practice action is disabled and shows the upsell messaging.
- **Branch cap enforcement**: Given a node already has 10 outgoing branches, when the user attempts to add another, then the app blocks and shows the correct warning ladder message (per CH30).
- **Uploads are private-only**: Given a pro user uploads a video to a move, when they create an unlisted share link, then the share preview explicitly indicates uploads do not transfer and only links will be visible to recipients.
- **Trial behaves as Pro**: Given a user in trial state, when they attempt Pro-only actions (practice, uploads), then actions are allowed and no free caps are applied.

8. Replit Build Prompt (Implement CH02 as Enforced Constraints)

```
Goal: Implement the CH02 constraints as enforceable rules (not just documentation).

Inputs: Attach CH00 + CH02. Treat cross-references as dependencies; do not invent
missing rules.

Step 1 - Create a single source of truth config:
- Create /src/config/entitlements.ts (or .js) that defines plan states: GUEST, FREE,
PRO, TRIAL.
- Create /src/config/limits.ts defining caps: FREE_SAVED_FLOWS=2, FREE_INBOX_CAP=10,
MAX_BRANCHES_PER_NODE=10, PRO_UPLOAD_CAP_BYTES=2GB, PRACTICE_CREDITS_FREE_MONTHLY=3.

Step 2 - Create feature gate helpers:
- /src/logic/gates.ts with functions like canSaveFlow(user), canPracticeFlow(user,
flow), canUploadMedia(user), canAddBranch(node, user), canSaveInboxItemToLibrary(user).
- Ensure these helpers return both (allowed:boolean) and (reasonCode:string) so UI can
show consistent messages.

Step 3 - Enforce orientation:
- Lock the app to portrait. Ensure screens do not assume landscape widths.
(Implementation approach depends on framework.)

Step 4 - Enforce guest restrictions:
- Guest can browse demo content, but cannot save flows or create durable content.
- On entering builder as guest, show a pre-warning: 'Saving requires an account' with
CTA to Sign Up.
```

```
Step 5 - Enforce free caps server-side too:
- Backend must validate: cannot create saved flow #3 on FREE; cannot exceed inbox cap;
cannot exceed branch cap; cannot upload on FREE.

Step 6 - Wire consistent UX messaging:
- Use a shared copy map keyed by reasonCode to display identical text and CTAs across
app.

Deliverables:
- limits.ts + gates.ts implemented and used everywhere.
- Unit tests for gates (basic).
- Manual test checklist matches CH02 acceptance tests.

Stop & ask if:
- Any cap value is missing or conflicting; log it as PLACEHOLDER and halt that feature
until confirmed.
```

9. Troubleshooting Notes (Chapter-Specific)

- If the UI rotates: confirm orientation lock is applied at the app shell level AND screens don't reflow into landscape by accident.
- If guests can save drafts: check for local persistence (AsyncStorage/SQLite) being used without gating; ensure draft saving is gated or purged on exit.
- If a free user can bypass caps via imports: ensure gates apply on 'save to library' action, not only on creation.
- If share recipients see missing media: ensure UI clearly labels 'Uploads are private-only' and encourage link attachments for shareable media.
- If caps differ across screens: ensure all screens pull from limits.ts, not duplicated constants.