Handz V1 PRD Bundle

CH36 - Troubleshooting Playbook (R1)

Generated: 2026-01-02 • Bundle ID: HZ-V1

Doc ID: HZ-V1-CH36_Troubleshooting_Playbook_R1

Revision: R1 (2026-01-02)

Status: Draft

Depends on: CH00 Master Index (anchor), CH31 Error States (copy + severity), CH35 QA Acceptance Tests (test language + coverage)

Related: CH07 Auth, CH08 Entitlements, CH09-11 Moves, CH12-14 Flow Builder, CH15-16 Library, CH17-19 Sharing/Inbox/Import Conflicts, CH20-22 Practice, CH23-24 Mastery/Maintenance, CH27 Notifications, CH28 Offline & Sync, CH29 Storage & Limits, CH30 Safety/Abuse, CH33 Analytics, CH34 Export/Deletion

Supersedes: None

Owned Decisions: This chapter defines the debugging and troubleshooting workflow, required diagnostic signals, and the canonical triage ladder for V1.

Open Questions / Placeholders: PLACEHOLDER: exact analytics vendor (Owner: CH33). PLACEHOLDER: crash reporting vendor (Owner: CH33). PLACEHOLDER: in-app dev diagnostics entry gesture (Owner: CH04/CH31).

Purpose

This chapter is the operational playbook for diagnosing, reproducing, and fixing issues in Handz V1. It is written so a builder (including a vibe-coding agent) can identify the root cause of a bug without guessing, and can confirm the fix with repeatable regression checks. When a bug touches product rules, the owning chapter must be updated first (see CH00).

Non-negotiable rule: troubleshoot the product rule first, then the code.

If you find a contradiction between app behavior and the PRD bundle, fix the owning chapter (new revision) before patching implementation.

How to use this playbook

- **Step 0 - Capture context:** device, iOS version, app version/build, account state (Guest/Free/Trial/Pro), network state (online/offline/poor), and the exact screen route where the issue occurs (see CH04/CH05).
- **Step 1 - Reproduce:** attempt to reproduce in the smallest possible scenario. If it only happens after long sessions, create a minimal dataset that still triggers it (example: a flow with 1 branch vs a flow with 10 branches).

- **Step 2 - Classify:** is it a crash, a hard blocker, a soft blocker, a data integrity issue, a paywall/entitlement mismatch, a sync conflict, or a UX/copy mismatch (see CH31)?
- **Step 3 - Localize:** decide which subsystem owns it (Auth, Moves, Flow Builder, Sharing, Inbox, Practice, Entitlements, Offline/Sync, Storage/Uploads, Notifications, Analytics).
- **Step 4 - Diagnose with signals:** use the diagnostic checklist for that subsystem (logs, local database, remote database, entitlement state, network calls, and UI state).
- **Step 5 - Fix and prevent:** apply a fix that also prevents recurrence (validation, idempotency, debouncing, conflict resolution, or better error states).
- **Step 6 - Regression:** run the regression checklist for that subsystem and add/extend tests in CH35.

Bug report template (copy/paste)

Use this exact template when reporting issues to yourself, a collaborator, or a vibe-coding agent. The goal is to eliminate ambiguity.

```
Title: [short symptom] (e.g., "Practice timer ends early when phone locks")
Severity: Crash | Blocker | Major | Minor | Cosmetic (see CH31)
Account state: Guest | Free | Trial | Pro
Device/iOS: [model], iOS [version]
App build: [version], [build number], [TestFlight/App Store/dev]
Network: Offline | Poor | Normal
Route/screen: [tab/screen], starting from launch
Preconditions: [what must already exist? e.g., flow with 10 branches, 2 saved flows
used]
Steps to reproduce:
1) ...
2) ...
Expected: ...
Actual: ...
Frequency: Always | Often | Sometimes | Once
Logs/screenshots: [attach]
Data impact: None | Lost draft | Duplicated items | Wrong entitlement | Wrong practice
log
Workaround: [if any]
```

Triage ladder (what to check first)

- **Crashes and data loss first.** If users can lose flows, practice logs, or moves, treat as highest priority.
- **Entitlement mismatches second.** If Pro/Trial features behave like Free (or vice versa), fix gating before anything else (see CH08/CH25).
- **Sync integrity third.** If offline edits overwrite newer changes, fix conflict rules (see CH28).
- **UX and copy last.** Cosmetic issues can wait, but must still be fixed before App Store submission.

Required diagnostics in V1 builds

Handz V1 must ship with enough diagnostics to troubleshoot real user issues without needing private developer tools. This does NOT mean exposing sensitive data to users. It means providing safe, user-consented ways to capture context, and providing internal logging and error boundaries for developers.

Minimum diagnostics (must-have)

- **Global error boundary:** catches unhandled UI exceptions and shows a recovery UI with a 'Report issue' action (See: CH31).
- **Network request logging:** in dev builds, log request path, status, latency, and error type. In prod, log only aggregate/anonymous metrics unless user consents.
- **Local persistence audit:** ability to show whether a record is 'local-only', 'synced', or 'sync-failed' (See: CH28).
- **Entitlement snapshot:** a single struct showing current plan state (Guest/Free/Trial/Pro), trial expiry, and the feature gates currently active (See: CH08).
- **Storage snapshot:** current local media usage and server media usage vs the 2GB cap; link vs upload distinction (See: CH29).
- **Practice snapshot:** current practice credit balance for Free, and whether the current flow is 'saved' vs 'inbox' (See: CH20).
- **Share-link snapshot:** number of share links created today, and any throttle/abuse flags (See: CH17/CH30).
- **Bug report export:** one-tap 'Copy diagnostic bundle' that copies a sanitized text block into clipboard, for the user to paste into support (no secrets).

Diagnostics must NOT include

- Authentication tokens, raw payment receipts, or personally identifying info beyond user ID/handle (unless explicitly consented).
- Private move notes or private video contents in a bug report unless user explicitly includes them.

Golden rule for troubleshooting in production

If you cannot reproduce an issue locally, treat the user's report as a data point and improve observability: add a signal, log, or state snapshot so the next occurrence becomes diagnosable. Do not guess.

Common failure modes and quick routing

Use this as a 'where do I look' index. Each section below contains deep steps. If you are unsure which subsystem owns the bug, start here.

- **App won't launch / crashes on open:** see §1 (Crash and Launch).
- **Sign-in doesn't work, stuck on onboarding, can't save:** see §2 (Auth and Account) and CH07.

- **Free/Trial/Pro gates behave wrong, practice blocked unexpectedly:** see §3 (Entitlements and Paywalls) and CH08/CH25.
- **Moves missing, duplicates, aliases wrong (teep vs push kick):** see §4 (Moves System) and CH09-11.
- **Flow builder glitches, can't connect nodes, pan/zoom broken, performance:** see §5 (Flow Builder) and CH12-14.
- **Share link fails, receiver can't open, revoke doesn't work:** see §6 (Sharing) and CH17.
- **Inbox imports don't show, cap wrong, save-to-library conflicts:** see §7 (Inbox and Import Conflicts) and CH18-19.
- **Practice timer wrong, history wrong, streak wrong:** see §8 (Practice) and CH20-22.
- **Offline edits disappear or overwrite:** see §9 (Offline and Sync) and CH28.
- **Uploads exceed cap, links vs uploads confusion, can't share videos:** see §10 (Storage and Media) and CH29.
- **Notifications not firing / wrong time zone:** see §11 (Notifications) and CH27.
- **Warnings/anti-abuse ladder misfires:** see §12 (Safety/Abuse) and CH30.
- **Export/deletion broken:** see §13 (Export and Deletion) and CH34.

§1. Crash and Launch

Issues that prevent the app from opening, keep users on a blank screen, or crash during navigation. These are highest priority because they block onboarding and destroy trust.

Symptoms

- App crashes immediately after tapping the icon.
- App opens to a blank screen or infinite spinner.
- Crash occurs only after updating from an older build.
- Crash occurs only for accounts with many flows/moves.
- App becomes unresponsive during heavy canvas interactions (possible memory pressure).

Diagnosis checklist

- Confirm build type: dev vs TestFlight vs App Store. Confirm iOS version and device model.
- Check crash reporting console (if present) for stack trace, crash frequency, and affected OS versions.
- Attempt cold start, then warm start. Note whether crash happens before any UI appears (early init) or after rendering (screen-level).
- If crash after update: verify local migrations (schema, cached data) ran successfully. Look for corrupt local data that cannot be parsed.
- If crash only with large datasets: reproduce with a synthetic 'large' user (hundreds of moves, flows with many nodes). Watch memory usage and frame rate.
- Confirm network is not required for first paint. App must show a usable offline-safe shell even if network is down (see CH28).
- Check for infinite loops: repeated navigation redirects, repeated auth state toggles, or repeated re-renders from state changes.

Likely fixes

- Add a global error boundary around the root navigator. On crash-like UI exceptions, show a recovery screen and allow restarting the app shell.
- Guard JSON parsing and migrations with try/catch. If a single record is corrupt, quarantine it (mark as 'unreadable') instead of crashing the whole app.
- Make launch path idempotent: initialization should be safe to run multiple times without duplicating data or navigating repeatedly.
- Defer heavy work: do not hydrate the entire flow canvas or preload all videos at launch. Load lists lazily and paginate.
- Add timeouts for network-dependent calls. On timeout, fall back to cached data and show a non-blocking banner.

Prevention / hardening

- Add a lightweight 'safe mode' path: if the app crashes twice in a row, disable heavy features (canvas previews, video thumbnails) until user opts in.
- Add schema versioning for local caches. On mismatch, migrate or clear cache with user-safe messaging (do not delete cloud data).
- Add performance budgets: max nodes rendered in thumbnails; max simultaneous video thumbnail fetches.
- Add automated smoke test in CH35: launch app, reach dashboard, open a demo flow, return, repeat 3x.

Regression checklist (run after fix)

- Cold launch with network off: app still reaches dashboard shell and shows demo content or empty state.
- Cold launch after upgrade: existing flows and moves still appear; no duplicates; no crash.
- Navigate through all primary tabs and open at least one flow and one move detail screen.
- Stress test: open a heavy flow (many nodes/branches), pan/zoom for 30 seconds, exit. App should not crash.

§2. Auth and Account (Guest, Free, Pro)

Issues with sign-up, login, session persistence, guest restrictions, and account conversion prompts. In Handz V1, accounts are required to save flows and to unlock personalization.

Symptoms

- User cannot sign up with Google or email (button does nothing, spinner forever).
- User signs up but gets stuck on verification, or returns to welcome screen repeatedly.
- Guest user starts building a flow but later learns they cannot save (conversion prompt missing or confusing).
- After login, user data is missing (appears as a new account) or the wrong account is loaded.
- Log out/log in cycles cause duplicate demo flows or reset onboarding.

Diagnosis checklist

- Confirm the exact auth method used: Apple, Google, Email. Confirm whether email verification is required and whether it is completed.
- Confirm whether the user is in Guest mode or has an authenticated user ID. Check the entitlement snapshot and auth snapshot.
- Check for route loops: welcome -> setup -> dashboard -> welcome. Usually caused by auth state not persisting or a gating rule reading stale state.
- Verify account conversion flows: guest -> sign up should preserve temporary work only if explicitly designed (V1 rule: no local save for guests).
- Check uniqueness rules for username/handle: verify the check is debounced and idempotent (avoid race conditions).
- Test bad networks: sign-in should fail with an actionable message and allow retry without app restart.

Likely fixes

- Make auth state the single source of truth. All screens read from it, and navigation reacts only to stable transitions (signedOut -> signedIn).
- Add clear preflight messaging for Guest users: before entering Flow Builder, show a gate that explains 'You can explore, but saving requires an account.'
- Ensure Guest mode cannot create persistent local flow records. Treat any guest flow as an in-memory draft that is discarded when leaving the builder.
- Ensure demo content is not duplicated: demo flows must be tagged as demo and created at most once per install/account.
- When auth fails, show the CH31 error state with a retry action and a 'contact support' path.

Prevention / hardening

- Add auth state tests: token refresh, app restart, airplane mode, and 'login while offline' should produce predictable errors.
- Add idempotent onboarding: 'hasCompletedOnboarding' flag must be consistent and not reset accidentally.
- Add a dedicated 'Account required' gate component reused anywhere saving/exporting/sharing is attempted.

Regression checklist (run after fix)

- Sign up with Google and email, then force quit and reopen: user remains signed in.
- Enter Flow Builder as Guest: user sees account-required explanation before building; cannot save; leaving the screen discards draft.
- Log out then log back in: demo content does not duplicate; user data remains intact.
- Username availability check: typing quickly does not freeze UI; taken vs available states are accurate.

§3. Entitlements, Paywalls, Trials, and Feature Gating

Issues where Free vs Trial vs Pro rules behave incorrectly: features are blocked when they should be allowed, or allowed when they should be blocked. These issues directly impact trust and revenue.

Symptoms

- Pro user is treated as Free (practice locked, share link limit hit, exports blocked).
- Free user can practice without credits, or can practice inbox flows (should be blocked).
- Trial started but does not unlock Pro gates, or trial expiry does not re-lock features.
- Restore purchases does nothing, or shows wrong plan state.
- Paywall triggers too aggressively, interrupting core exploration.

Diagnosis checklist

- Capture entitlement snapshot: account type, trial active flag, trial expiry, plan identifier, last receipt refresh time.
- Verify feature gate evaluation order: Guest gate -> Free caps -> Trial/Pro overrides. A common bug is checking caps before checking Pro.
- Check device time vs server time. Trial expiry and monthly credit refresh must be based on a trusted time source to prevent abuse and time drift.
- Verify that practice credits apply only to saved flows and NOT to inbox items. Confirm the current flow's origin: saved library vs inbox.
- Simulate edge cases: user upgrades during an active practice session; user downgrades; user reinstalls and restores; user changes Apple ID.
- Confirm paywall copy matches the actual gate (CH31). If the message says 'Upgrade to practice' but the actual block is 'need to save flow first', it creates confusion.

Likely fixes

- Centralize gates in one module: a single function that decides allow/deny and returns the reason code used by UI messaging.
- Add a 'refresh entitlements' action: on app launch, on returning from paywall, and on manual pull-to-refresh in settings.
- Implement restore purchases correctly and show clear result states: success (plan restored), nothing to restore, error (retry).
- Fix gate messaging order: if a user hits 'practice' from an inbox item, show 'Save this flow to your library to practice it' before any subscription upsell.
- Enforce credit decrement atomically: a practice session either consumes a credit on start, or on completion, but not both. Use idempotent session IDs.

Prevention / hardening

- Add a 'Gate Reason Matrix' test suite (CH35): for each feature, test Guest, Free, Trial, Pro states and verify allow/deny and messaging.
- Log gating decisions in dev builds (reason codes). In prod, log aggregate counts to detect unexpected spikes.
- Never hard-code plan logic inside UI components; UI requests a gate decision from the gate module.

Regression checklist (run after fix)

- Free user with 0 credits: cannot start practice; sees paywall + explanation.
- Free user with credits: can practice only saved flows; inbox practice remains blocked.
- Trial user: all Pro gates open during trial; after expiry, Free gates apply again.
- Pro user: can practice without credits; gates remain open after app restart.
- Restore purchases: after reinstall, plan state matches account and device receipt.

§4. Moves System (Defaults, Aliases, Families, Variants, Customization)

Issues related to the move catalog: missing moves, duplicates, alias confusion (e.g., teep vs push kick), move family grouping, and user customization. V1 ships with a default move pack and allows user-created moves and tags, but ships with zero technique descriptions by default.

Symptoms

- Default moves missing after onboarding or after reinstall.
- User sees duplicate moves (same name appears twice) unexpectedly.
- Alias confusion: selecting 'Front Kick' shows as 'Teep', or vice versa.
- Edits to a move unexpectedly change flows that used that move (or do not change when expected).
- Shared flows bring in custom moves and cause conflicts with existing moves.

Diagnosis checklist

- Confirm whether the user is on the default move pack or a custom-selected pack (if onboarding offers style presets).
- Inspect move identifiers: each move must have a stable canonical ID. Display names can change, but IDs must not.
- Check whether duplicates share the same canonical ID (bug) or are intentionally separate moves (e.g., Push Kick vs Teep are separate).
- Verify family/variant metadata: 'base move' vs 'variant move' relationship must be stored explicitly (not inferred from name).
- Test edit scope: when editing a move, confirm whether the edit is library-wide or flow-local. Mis-scoped edits are a common source of user confusion.
- For shared imports: confirm the importer sees a conflict resolution UI when the incoming flow includes custom move data for moves they already have.

Likely fixes

- Enforce canonical ID uniqueness. Never generate IDs from display names.
- Treat aliases as separate 'display names' linked to a canonical move, OR as separate canonical moves when the product decides they should be distinct (Push Kick and Teep).
- Implement a clear scope choice for edits: 'Apply to my library move' vs 'Apply only inside this flow'. Default should be safe and reversible.
- Add a 'View move family' panel: show which moves are in the same family and which are variants of which base move. Let users pick multiple in a family.
- For imports: provide three safe options when conflicts occur: (1) keep my version, (2) use sender version only inside this flow, (3) add sender version as a new move in my library (renameable).

Prevention / hardening

- Onboarding must explain the difference between: name/alias, family, variant, and custom move.
- Add a 'Revert' system with granularity: revert a single move, revert a single flow, or revert the whole library to last sync snapshot.
- Add validation rules: disallow empty names, enforce max lengths, and warn on near-duplicate names to reduce accidental duplicates.
- Ship default move pack without descriptions to avoid disagreement. Encourage users to add their own notes/videos instead.

Regression checklist (run after fix)

- Fresh install: default moves appear; user can search and add moves to a flow.
- Rename a move alias: flows still reference the same canonical move and render correctly.
- Import a flow with custom move details: conflict UI appears; chosen option behaves correctly; no silent overwrite.
- Revert a single move: returns to previous state without affecting unrelated moves.

§5. Flow Builder (Canvas, Nodes, Branches, Merges, Reorder)

Issues in the core feature: creating and editing striking decision flows. V1 supports branches (up to 10 outgoing connections per move), merges (multiple incoming), optional sequence/explanation nodes, pan/zoom, and both top-to-bottom and left-to-right layouts. Dangling paths are allowed.

Symptoms

- Cannot connect a move to another move; connector handle unresponsive.
- Branches disappear, or connecting a new branch overwrites an existing one.
- Merge node drops some incoming paths or mis-renders ordering.
- Reordering fails: user drags a node but the order does not persist or links break.
- Pan/zoom jittery; canvas resets position; multi-touch conflicts with scroll.
- Large flows become slow, crash, or freeze; or nodes overlap making the flow unreadable.

Diagnosis checklist

- Confirm layout mode (top-to-bottom vs left-to-right) and zoom level when the bug occurs.
- Check node identity: each node needs a stable node ID separate from move canonical ID. Links must reference node IDs.
- Verify edge model: edges must allow 1-to-many outgoing and many-to-1 incoming. Bugs often come from assuming only 1 outgoing edge.
- Check 'max 10 branches' enforcement: ensure UI prevents adding the 11th, and returns a clear message. Ensure '10' is treated as a soft cap if the product later changes it.
- Test optional sequence nodes: ensure flows work with zero sequences between moves, and sequences can also branch and connect to moves.
- For reorder: confirm whether reorder affects only visual layout or actual execution order. V1 requires reorder to persist and reflect the user's chosen reading order.
- Check persistence: after edits, save, exit, and reopen. Ensure the graph renders identically and no links are lost.

Likely fixes

- Separate model from view: maintain a canonical graph model (nodes/edges) and derive layout from it. Do not store pixel coordinates as the only truth.
- Add validation before save: detect orphan nodes, broken edges, and missing references; repair or block save with an actionable message.
- Implement deterministic ordering for outgoing branches: store an explicit order array per node so rendering does not depend on insertion order.
- Debounce saves and make them idempotent: repeated saves should not duplicate nodes or edges.

- For performance: virtualize rendering, collapse subtrees, and avoid re-layout on every tiny interaction. Cache layout.
- Provide a 'zoom to fit' and 'center on selected' action to recover from lost nodes.

Prevention / hardening

- Add a 'graph invariants' unit test suite: no duplicate node IDs, edges reference valid nodes, order arrays match outgoing edges.
- Add interaction tests (CH35): create node, branch 3 ways, merge back, reorder branches, save, reopen, verify identical.
- Add soft warnings before limits: e.g., when a node reaches 8 branches, warn 'complexity rising' without blocking (see CH30/soft caps).
- Provide a 'summary node' option (if enabled) to collapse many branches into one labeled node for readability, while preserving underlying paths.

Regression checklist (run after fix)

- Create a flow with 1 root move, add 10 outgoing branches, then branch one of those branches into 5 more moves; save/reopen; no links lost.
- Create a merge node with 3 incoming paths; reorder one path; save/reopen; merge still has 3 incoming.
- Switch layout direction (TB vs LR) and verify the graph stays correct; only the presentation changes.
- Pan/zoom for 60 seconds; no accidental node drags; no canvas reset.

Addendum to §5 - Clarifying terms

Dangling path: a node that has an outgoing connector with no next node attached yet (an unfinished plan). V1 allows dangling paths so users can draft and return later.

Summary node: a special non-move node that collapses a set of outgoing branches into a single labeled hub (example: 'Opponent reactions'). Tapping it expands the hidden branches. Purpose: readability when a move has many branches.

Summary nodes are optional in V1. If implemented, they must be purely a view feature: they may not change the underlying graph semantics, and export/sharing must preserve the real branches.

§6. Sharing (Unlisted Links, Duplicate Flow, Revoke)

Issues with unlisted sharing links and duplication. V1 supports unlisted share links for flows and gameplans. Sharing should be safe, abuse-resistant, and predictable.

Symptoms

- Share link opens to an error or blank screen.
- Receiver opens link but sees missing moves or broken paths.
- Revoke link does not invalidate it.
- User hits daily share link limit unexpectedly.
- Shared flow includes uploaded videos that receiver cannot access (expected in V1: uploads do not transfer; links do).

Diagnosis checklist

- Confirm whether the sender shared a flow or a gameplan. Confirm share link type and version.
- Check whether the sender included custom moves or custom move notes. If yes, receiver should see an import conflict screen.
- Verify that the shared object is immutable: sharing should snapshot the sender's content at share time (unless the product defines live sharing).
- If receiver sees missing media: check whether the sender used uploads (gallery video) vs links (YouTube, etc.). V1 rule: only links are shareable across accounts; uploads are private to the uploader.
- Inspect rate limits: confirm today's share-link count and whether the account is Free/Pro. Ensure limits are 'soft caps' where possible and communicate clearly.
- If revoke fails: verify that the share link token is checked server-side at open time (not just cached locally).

Likely fixes

- Ensure share links resolve server-side to a share record with an active flag. Revocation flips active to false.
- Implement an import preflight on open: show what will be imported (flow, moves, notes, tags, links) and allow the user to choose scope.
- If the receiver lacks some moves, map by canonical ID or by alias mapping rules; if no match, import as new custom moves under a separate namespace.
- Add explicit share messaging: if a flow contains private uploads, show 'Uploads stay private; only links are shared' before generating the link.
- Make 'Duplicate into my library' idempotent: repeated taps should not create multiple duplicates.

Prevention / hardening

- Add share-link integrity tests: create link, open on another account, revoke, open again -> shows revoked state.
- Add abuse protections (CH30): per-day soft caps, IP-based throttles, and anomaly detection for link creation.
- Version share payloads so future schema changes do not break old links.

Regression checklist (run after fix)

- Share a simple flow (no custom moves) and import it on another account. Flow graph matches; moves map correctly.
- Share a flow with custom moves and custom notes. Receiver sees conflict UI and can choose 'flow-only' vs 'add to library'.
- Revoke link and confirm it no longer opens.
- Attempt to share a flow with uploaded videos: app warns that uploads will not transfer; receiver sees links only.

§7. Inbox and Import Conflicts (Free Inbox Cap, Save-to-Library)

Issues with the import inbox, inbox caps, and conflict handling when saving imported flows into the user's library. In V1, imports are allowed to keep the sharing funnel healthy, but Free users have tight save limits and cannot practice inbox items.

Symptoms

- Imported flow does not appear in inbox after opening a share link.
- Inbox shows more than the Free cap (should be 10) or blocks too early.
- Free user cannot save an imported flow even when they have capacity (2 saved flows rule).
- Saving an imported flow overwrites existing moves or notes silently.
- Inbox item becomes corrupted after sender updates their content.

Diagnosis checklist

- Confirm whether the user is Free/Pro and whether inbox cap rules are applied to their plan.
- Verify that inbox items are immutable snapshots at import time. They should not change if the sender edits their original flow later.
- Check the save-to-library flow: it must run the import conflict UI if incoming payload includes custom move data.
- Confirm that Free users can view inbox items but cannot practice them and cannot convert them to saved flows if already at 2 saved flows.
- Check for duplicate imports: opening the same share link multiple times should not create multiple inbox items.

Likely fixes

- Enforce inbox caps at insert time. If at cap, show a clear message and offer to delete older inbox items (or upgrade).
- Implement idempotent import by share-link token: same token -> same inbox record unless explicitly 'import again'.
- When Free user is at max saved flows, saving from inbox must show an upgrade prompt OR a manage-storage prompt (delete an existing saved flow).
- Implement safe conflict resolution: never silently overwrite a user's existing move notes. Require explicit choice.
- Store imported payloads with a schema version and validate on read; if invalid, quarantine and show a recoverable error.

Prevention / hardening

- Add import tests: open share link twice -> only one inbox item; delete -> open link -> new item appears.

- Add cap tests: Free inbox stops at 10; Pro cap is higher/soft; messaging is correct.
- Add conflict tests: sender customizes a common move (jab) -> receiver chooses to keep their jab vs flow-only vs new move.

Regression checklist (run after fix)

- Free user: can import and view up to 10 inbox items.
- Free user: cannot practice inbox items; practice button shows correct gate reason.
- Free user at 2 saved flows: attempt to save from inbox triggers correct flow (upgrade or delete existing).
- Import conflict choices behave correctly and are reversible (revert single flow/move).

§8. Practice Mode (Drills, Timer, Credits, Interruptions, History)

Issues with the practice/drill feature: starting sessions, selecting paths, timer behavior, recording outcomes, and updating progress metrics. Practice is a Pro feature with limited Free credits and optional demo practice content.

Symptoms

- Practice session cannot start even though user should be eligible (Pro or has credits).
- Timer ends early, pauses unexpectedly, or continues while app is backgrounded incorrectly.
- Practice history logs wrong duration, wrong path order, or wrong 'completed' vs 'interrupted' state.
- User selected a set of paths, but the drill uses different paths or repeats incorrectly.
- Credits decrement incorrectly (double-charge, no charge, or charge even when session fails).

Diagnosis checklist

- Confirm eligibility: account type, credit balance (Free), and whether the flow is saved vs inbox.
- Confirm the practice configuration: selected paths list, order, timer per path, and assumed reps per interval.
- Test background behavior: lock screen, switch apps, receive a call. Verify whether timer should pause or continue per product rules.
- Check session state machine: NotStarted -> Running -> Paused -> Completed/Interrupted. Bugs usually come from skipping transitions.
- Verify idempotent session IDs: each session should have one record; retries should not create duplicates.
- Confirm actual duration logging: record start timestamp, end timestamp, and compute duration. Do not rely only on timer target.

Likely fixes

- Follow fitness tracker convention: if user ends early, mark as 'Interrupted' and log actual duration. If timer finishes naturally or user taps 'Completed', mark as completed.
- If app goes to background, choose a consistent rule (pause vs continue) and implement it reliably. Most V1-friendly rule: pause on background and require user to resume (reduces time drift and cheating).
- Make credit decrement atomic and tied to session lifecycle: recommended approach is to reserve a credit at start and commit on completion; if interrupted before a minimum threshold, release the credit (product decision).
- Ensure path selection order is explicit and persisted with the session config so replays are deterministic.
- Add clear recovery UI if a session fails to start due to network/entitlement refresh: allow retry without losing selected paths.

Prevention / hardening

- Add timer tests: run 60s timer, background for 10s, resume -> expected behavior matches rule.
- Add practice history integrity tests: create session -> confirm record fields (paths, duration, status).
- Add 'dry run' mode: simulate session without decrementing credits (used for demo practice content).

Regression checklist (run after fix)

- Free user with credits: start practice on a saved flow; complete -> credit decrements once; history shows completed.
- End early -> session marked interrupted; duration equals actual elapsed time; credit behavior matches spec.
- Attempt practice on inbox flow -> blocked with correct message.
- Pro user: unlimited practice; no credits visible; history still records sessions.

§9. Offline and Sync (Drafts, Conflicts, Retries)

Issues when editing flows/moves offline and syncing later. V1 must be resilient: users may train in gyms with poor service. The app must avoid silent data loss and must surface sync state clearly.

Symptoms

- Edits made offline disappear after reconnecting.
- Older data overwrites newer data from another device.
- Flow shows 'syncing' forever.
- User cannot open a flow while offline even though it was previously viewed.
- Duplicate flows appear after reconnecting.

Diagnosis checklist

- Confirm the user's network state at edit time and at sync time.
- Check record sync metadata: lastModifiedAt, lastSyncedAt, localDirty flag, and conflict flag.
- Confirm conflict policy: last-write-wins is dangerous for graphs; prefer merge or explicit conflict UI for flow graphs.
- Check whether media uploads were queued offline and then failed later; failures must not block non-media sync.
- Confirm retries: exponential backoff, and a manual 'retry sync' action in settings or on the record.

Likely fixes

- Implement per-record sync states with clear UI: Synced, Pending, Failed, Conflict. Provide 'Retry' and 'Resolve' actions.
- For flow graphs, avoid silent merges that can drop edges. Use a conflict UI that lets user choose: keep local, keep remote, or duplicate (safe fork).
- Separate critical and non-critical sync: allow text/graph updates to sync even if media upload fails.
- Make create operations idempotent: if a retry occurs, it should not create duplicates.

Prevention / hardening

- Add offline test suite: edit flow offline, create new flow, reconnect, verify server has exactly one copy.
- Add conflict simulations: edit same flow on two devices then reconnect; ensure conflict UI appears and no silent overwrite.
- Add sync observability: log retry counts and failure codes to detect systemic issues.

Regression checklist (run after fix)

- Offline edit persists locally and syncs later.
- Conflict scenario yields explicit resolution UI and results in expected final state.

- Sync failures show a banner and do not block app navigation.
- No duplicate flows after reconnecting.

§10. Storage and Media (Links vs Uploads, 2GB Cap, Sharing Rules)

Issues with attaching media to moves/flows, respecting storage limits, and communicating shareability. V1 supports both links and uploads, but only links are shareable across users.

Symptoms

- User cannot upload video even though they are under cap.
- Upload succeeds but later disappears or fails to play.
- Storage meter incorrect (shows 0GB used when uploads exist, or shows over cap incorrectly).
- User shares a flow and expects uploaded videos to transfer; receiver cannot see them.
- Video thumbnails cause slowdowns or memory spikes.

Diagnosis checklist

- Confirm the user's storage usage: local cache, remote usage, and the 2GB cap.
- Check whether the upload failed due to file type, size, network timeout, or permission denial.
- Confirm whether the media is attached as a link or as an upload. Only links should appear in shared payloads.
- Verify media access rules: uploads should require auth and should not be guessable by URL.
- Inspect caching: repeated playback should not re-download; but cache should not exceed safe device storage limits.

Likely fixes

- Implement clear upload error messages: file too large, unsupported format, network failure, permission denied.
- Make storage meter authoritative: refresh from server and reconcile with local cache; show last updated time.
- On share: show a warning if the flow contains uploads. Offer to convert upload to link (user paste link) as a shareable alternative.
- Lazy-load thumbnails and paginate media lists; avoid loading multiple video players at once.

Prevention / hardening

- Add upload tests near cap: upload until 1.9GB, then attempt another upload, confirm correct gating.
- Add share tests: uploaded media remains private; shared flow contains links only; messaging is consistent.
- Add security tests: uploads require authenticated access; revoked account cannot access old uploads.

Regression checklist (run after fix)

- Upload small video and play it; restart app; video still available.
- Attempt to upload when at cap -> blocked with correct message and guidance.
- Share a flow with uploads -> receiver sees no uploads and sees correct explanation.
- Scrolling a list with many videos does not crash or freeze.

§11. Notifications and Scheduling (Local Time, Reminders)

Issues with reminders for maintenance and practice, including time zone behavior and permission handling.

Symptoms

- Notifications never arrive even after enabling.
- Notifications arrive at the wrong time (time zone drift).
- User changes the reminder time but old reminders still fire.
- Permission prompt never shows or shows repeatedly.
- Streak/consistency metrics do not match reminders.

Diagnosis checklist

- Confirm notification permission status and whether the user allowed alerts, sounds, and badges.
- Confirm the device time zone and whether the app uses local time for scheduling (V1 rule: user local time).
- Check whether reminders are scheduled as repeating events and whether rescheduling cancels old ones.
- If reminders depend on server state (maintenance plan), confirm plan generation and local scheduling are in sync.
- Test with device time changes: manual time change, DST boundary, travel to another time zone.

Likely fixes

- Always provide a 'Test notification' action in settings so users can verify permissions.
- When user edits a reminder, cancel previous scheduled notifications before scheduling new ones.
- Use local time scheduling with clear UI: show next scheduled time in the user's local zone.
- If permissions denied, show an in-app card explaining how to enable them in Settings.

Prevention / hardening

- Add scheduling tests including DST and time zone change scenarios.
- Log reminder scheduling events (anonymized) to detect if scheduling fails on specific OS versions.
- Keep reminders optional and non-punitive: missing reminders should not lock features.

Regression checklist (run after fix)

- Enable reminders and confirm at least one fires at the right time.
- Change reminder time -> old reminder stops, new one fires.
- Deny permissions -> app shows guidance and does not loop prompts.

- Travel/time zone change -> reminders adapt to new local time.

§12. Safety, Abuse, and Warning Ladder (Soft Caps)

Issues where anti-abuse limits trigger incorrectly or fail to trigger. V1 prefers soft caps: warn before blocking, and keep restrictions non-intrusive for legitimate users (fighters/coaches spending long sessions).

Symptoms

- User is blocked from creating branches or share links even though behavior is normal.
- Warning ladder messages show too early, too often, or with confusing copy.
- Abusive behavior (link spam, automated imports) is not detected and harms the system.
- Legitimate users doing a 2-hour planning session are throttled incorrectly.

Diagnosis checklist

- Confirm which rule triggered: branch soft cap, share link cap, import inbox cap, upload cap, or suspicious behavior flag.
- Inspect counters: per-day counts, per-hour bursts, and rolling windows. Confirm time zone used for resets.
- Check whether the user is Pro/Trial vs Free. Pro should have higher soft caps, but still protected against extreme abuse.
- Review warning ladder state: how many warnings have been shown, when, and whether the next step is a block or a cooldown.
- Check copy source: messages should match CH31 severity and be consistent across the app.

Likely fixes

- Implement a warning ladder with memory: once a warning is shown, do not repeat it every time in the same session.
- Prefer frictionless soft caps: show a banner and allow continuation when safe; only hard block on clear abuse or extreme limits.
- Provide an escape hatch: if a legitimate user hits a block, allow appeal or 'contact support' with diagnostic bundle.
- Tune thresholds: if real users hit caps during normal usage, raise limits or switch to a rolling window instead of strict daily.

Prevention / hardening

- A/B test warning copy and frequency (later). For V1, keep copy short, calm, and non-accusatory.
- Add anomaly detection for link creation bursts and repeated failed imports.
- Keep all limits as configuration (not hard-coded) so they can be adjusted without a full rebuild.

Regression checklist (run after fix)

- Normal heavy use (2-hour session, many edits) does not trigger hard blocks.

- Creating many share links quickly triggers only a soft warning first, then a cooldown, not an immediate permanent block.
- Free inbox cap of 10 enforced consistently with clear messaging.
- After cooldown ends, user can resume normal activity.

§13. Export, Data Portability, and Account Deletion

Issues with exporting flows/gameplans/moves, and with deleting an account. These are trust and compliance features and must behave predictably.

Symptoms

- Export produces an empty file or missing branches.
- Export includes private uploads unexpectedly (privacy bug).
- Account deletion leaves data behind or breaks re-registration.
- User requests a copy of data but export is incomplete.

Diagnosis checklist

- Confirm export type: flow-only, library export, or gameplan export. Confirm whether export includes move notes, tags, and media links.
- Verify that exports map canonical IDs correctly and include schema version.
- Check privacy rules: uploads must not be exported unless explicitly selected; links can be included.
- For deletion: confirm soft-delete vs hard-delete policy, and confirm how long backups retain data (policy placeholder).

Likely fixes

- Implement export serialization from the canonical graph model, not from UI layout.
- Add export validation: after writing export, re-import it in a sandbox and compare graphs (structural equality).
- For deletion: revoke share links, delete uploads, and remove user data references. Then confirm login is blocked.
- Provide clear user messaging for deletion: what is removed, what is retained for legal reasons (if any).

Prevention / hardening

- Add export/import round-trip tests for simple and complex flows (branches, merges, sequences).
- Add privacy tests: exports do not leak private uploads.
- Add deletion tests: after deletion, user cannot access content; re-signup starts clean.

Regression checklist (run after fix)

- Export a complex flow and import it back: graph identical, no missing nodes/edges.
- Export does not include private uploads by default.
- Delete account -> reinstall -> cannot access old data; new account works.

§14. App Store Readiness (Paywall Copy, Scientific Claims, Compliance)

Issues that cause App Store rejection or user mistrust: unclear subscription terms, misleading performance claims, missing privacy disclosures, and inconsistent paywall flows. This section supports your goal of shipping V1 to Apple first.

Symptoms

- App Store reviewer flags misleading claims (e.g., '2x faster') without substantiation or disclaimers.
- Subscription terms unclear (trial length, price, renewal) or missing restore purchases.
- Paywall shown before user can understand value; reviewer sees it as gating core functionality too early.
- Privacy labels mismatch: app collects data not disclosed, or shares data without consent.

Diagnosis checklist

- Audit all in-app marketing copy, especially any 'scientific leverage' claims. Ensure disclaimers are present and not hidden.
- Confirm paywall contains: price, billing period, trial length, auto-renew statement, and restore purchases link.
- Verify the upgrade path is consistent: same paywall entry points, same plan options, and same outcomes.
- Confirm privacy settings: analytics opt-in/out, data export/deletion, and consent for uploading media.

Likely fixes

- Use cautious language: 'uses learning principles such as spaced repetition' rather than absolute outcomes, unless you have strong citations. Always include 'results may vary'.
- Add an in-app 'Methodology' page: explain what the app tracks (practice sessions, paths), how plans are generated, and that averages are not guarantees.
- Ensure restore purchases works and is discoverable from paywall and settings.
- Delay hard upsell until after a user experiences the core concept (demo flow + demo practice).

Prevention / hardening

- Maintain a single source for copy strings and version them. When updated, re-run a 'copy audit' checklist.
- Keep a compliance checklist (CH35 addendum): subscription terms, privacy disclosures, and claim disclaimers.
- Add screenshot-based QA: ensure no paywall overlaps or truncated text on common devices.

Regression checklist (run after fix)

- All paywall screens show correct price, trial, renewal language, and restore purchases.
- Scientific-claims screens include disclaimers and do not promise guaranteed outcomes.
- Privacy settings and labels are consistent with app behavior.

Vibe-coding troubleshooting workflow (Replit agent prompts)

These prompts are designed to be pasted into a vibe-coding agent when something breaks. They force the agent to diagnose before changing code, and to keep changes aligned with the PRD bundle.

Prompt T1 - Diagnose a bug before coding

```
You are implementing Handz V1. Before changing code, follow this checklist:
1) Restate the product rule from the PRD that governs this behavior (cite the
chapter/section).
2) Restate the reported bug using the Bug Report Template (CH36).
3) Produce a minimal reproduction plan (smallest dataset and steps).
4) List the top 3 likely root causes and what evidence would confirm each.
5) Add or identify the diagnostic signals needed (logs/state snapshots).
Only after steps 1-5, propose the smallest safe code change.
Then list the regression checklist you will run (CH36 + CH35).
```

Prompt T2 - Fix with guardrails

```
Implement the fix with these constraints:
- Do NOT change product behavior unless the PRD is updated first.
- Centralize the rule in the correct module (gating, graph model, import resolver,
etc.).
- Add a unit/integration test for the bug (CH35 style).
- Add/extend an error state if the failure can happen again (CH31).
- Add a sanitized diagnostic snapshot for this subsystem (CH36 'Minimum diagnostics').
Output: (a) what changed, (b) why it fixes root cause, (c) what tests you ran, (d) any
new edge cases created.
```

Prompt T3 - Create a diagnostic bundle copy

```
Add a 'Copy Diagnostic Bundle' action in Settings that copies a sanitized text block:
- App version/build
- Device/iOS
- Account state (Guest/Free/Trial/Pro)
- Entitlement snapshot
- Storage snapshot (local/remote usage vs 2GB cap)
- Practice snapshot (credits for Free, flow origin saved vs inbox)
- Last 20 app logs (sanitized, no tokens)
- Current route/screen
Ensure privacy: no personal text notes, no upload URLs, no auth tokens.
```

Acceptance criteria for CH36

This chapter is complete when the app and the build process can use it to diagnose issues without guessing.

- **AC36-1:** Bug report template exists and is used consistently by the team/agent.
- **AC36-2:** App implements minimum diagnostics: error boundary, entitlement snapshot, storage snapshot, practice snapshot, share-link snapshot, and copy diagnostic bundle.

- **AC36-3:** For each critical subsystem (Auth, Entitlements, Flow Builder, Sharing, Inbox, Practice, Offline/Sync), the playbook includes symptoms, diagnosis, fixes, prevention, and regression checks (this document).
- **AC36-4:** Regression checklists are mirrored into CH35 test cases before App Store submission.
- **AC36-5:** No diagnostic bundle includes secrets or private user content without consent.

Revision log

R1 (2026-01-02): Initial chapter created from CH00 structure. Includes subsystem-by-subsystem playbook, Replit prompts, and acceptance criteria.