# PromptVault: Core Experience Stabilization

**Status:** Proposed **Initiative:** Core Experience Stabilization **Release:** V1 Reliability Baseline **Owner:** Product **Collaborators:** Engineering, Design, QA

## TL;DR

PromptVault currently contains several high-visibility and workflow-breaking defects across its core experience, including unexpected deletion behavior, obstructed controls, unreliable keyboard interactions, and contradictory command-palette states.

Individually, some of these issues are small. Collectively, they make the product feel unpredictable and unfinished, which can reduce user trust.

Before investing engineering effort in new capabilities, PromptVault should establish a **stable and trustworthy product baseline**.

This release will focus on repairing the existing product experience rather than expanding functionality.

The release will prioritize:

1. Protecting user-created content from unintended deletion.
2. Ensuring core workflows can always be completed.
3. Making supported keyboard interactions predictable.
4. Eliminating contradictory or visibly broken UI states.
5. Establishing regression coverage for critical workflows.
6. Creating a reliable baseline for future usability testing and feature development.

## 1. Problem / Why Now

PromptVault's value depends on users trusting it to reliably store, retrieve, organize, and insert saved content.

Users save prompts and snippets because they expect that content to remain available when they need it later.

The current product contains several defects that violate that expectation.

Observed issues include:

- Removing a snippet from a group may remove the underlying snippet from the broader library.
- The floating + New button can obstruct controls during selection workflows.
- Multiple command-palette items can appear highlighted simultaneously.
- Keyboard shortcuts may fail or invoke unrelated browser actions.
- Temporary PromptVault interfaces do not consistently support predictable keyboard dismissal.

Together, these defects create a larger user problem:

> ***Users cannot consistently predict how PromptVault will behave when completing core actions.***

That uncertainty can undermine trust.

A user does not necessarily think about these as separate bugs.

They experience:

**Open command palette** → two items appear selected.

**Try keyboard command** → browser Print dialog opens.

**Reorganize content** → saved item unexpectedly disappears.

**Try another action** → a floating button blocks the required control.

The user's takeaway may simply be:

> ***"This product isn't reliable yet."***

This matters particularly because PromptVault asks users to store content they expect to retrieve later.

There is also a product-learning reason to fix these issues now.

Known defects can distort future usability testing. A user struggling with a new feature may actually be reacting to broken selection behavior, unreliable keyboard interactions, or confusing existing states surrounding that feature.

Creating a stable baseline therefore improves both the **product experience** and the **quality of subsequent product learning**.

---

# 2. Goals

## User Goals

Users should be able to:

- trust that saved content will not disappear because of unrelated organizational actions;
- complete core workflows without inaccessible or obstructed controls;
- understand which element or navigation state is currently active;
- use supported keyboard interactions without unexpected browser behavior;
- dismiss temporary PromptVault interfaces predictably;
- confidently rely on PromptVault's existing functionality.

## Business Goals

### Establish a stable product baseline before investing engineering effort in new capabilities

PromptVault should resolve material reliability problems in its existing experience before expanding the product.

A stable baseline reduces the risk of building new functionality on top of unresolved interaction, state-management, or data-integrity problems.

It also creates a more reliable foundation for evaluating future engineering investments.

### Improve the quality of subsequent usability testing by removing known defects that could distort results

Known defects can introduce noise into usability studies.

Removing those defects first allows future research to more accurately evaluate:

- comprehension;
- usability;

- discoverability;
- workflow efficiency;
- feature value.

This increases confidence that future product decisions are based on the feature being tested rather than unrelated defects in the existing product.

## Non-Goals

This release will not meaningfully expand PromptVault's feature set.

Out of scope:

- multi-group snippet membership;
- nested folders;
- redesigned group architecture;
- AI-powered categorization;
- major Starred/Favorites redesign;
- persistent variable-input history;
- new collaboration or sharing capabilities;
- major command-palette information architecture changes;
- major visual redesigns;
- advanced search functionality.

These ideas may be evaluated after PromptVault meets the reliability exit criteria defined in this PRD.

---

# 3. Scope & Priorities

| Issue | Priority | Scope | Rationale |
|---|---|---|---|
| Removing from group unexpectedly deletes underlying snippet | P0 | In | Data integrity and user trust |
| + New obstructs active selection controls | P0/P1 | In | Blocks workflow completion |
| Keyboard commands fail or trigger browser actions | P1 | In | Core interaction unpredictability |
| Escape dismissal | P1 | In | Predictable interaction behavior |
| Double command-palette highlighting | P1 | In | Highly visible contradictory state |
| Persistent active-tab state | P2 | In if low-cost | Improves system-state clarity |
| Severe title truncation/display issues | P2 | Conditional | Only when materially confusing |
| Rename snippets | - | Out | New capability |
| Multi-group membership | - | Out | New capability/data-model expansion |
| Nested folders | - | Out | New information architecture |
| Starred redesign | - | Out | Product enhancement |
| Variable auto-save | - | Out | New capability and additional privacy considerations |

## Prioritization principle

The release should address defects in this order:

**1. Data integrity** Anything that can unexpectedly destroy or modify user content.

**2. Workflow blockers** Anything preventing users from completing an existing core action.

**3. Interaction reliability** Anything causing an existing interaction to behave unpredictably.

**4. High-visibility consistency** Obvious contradictory states that materially make the product appear broken.

---

# 4. User Experience

The stabilization release should make PromptVault's core experience feel predictable.

A reliable interaction should satisfy three conditions:

1. **The user understands the current product state.**
2. **The user understands what their next action will do.**
3. **The product behaves as expected after that action.**

## Flow A - Reorganizing saved content

### 1. User opens a group

The group displays the snippets currently associated with it.

### 2. User removes a snippet from the group

PromptVault removes the snippet from that group.

The original snippet remains available in the main snippet library.

The action must not behave like permanent deletion.

### 3. User chooses to permanently delete a snippet

Permanent deletion is represented as a separate destructive action.

The interface clearly communicates that the underlying item will be removed.

The user can cancel before deletion occurs.

---

## Flow B - Using the Command Palette

### 1. User opens the Command Palette

The palette opens in a stable state.

Only one result may represent the active keyboard selection.

### 2. User navigates

Supported interactions should include:

- ↑ previous result;
- ↓ next result;
- Enter activate selected result;
- Esc dismiss the current temporary surface.

### 3. User transitions between mouse and keyboard

Hover and keyboard selection should not create two conflicting active states.

The interface should make clear which item will be activated by Enter.

### 4. User dismisses the palette

Pressing Esc closes the palette.

Where technically practical, focus returns to the previous user context.

---

## Flow C - Selection Mode

### 1. User enters selection mode

Contextual actions become available.

### 2. Floating creation controls adapt

The + New CTA must not cover or interfere with contextual controls.

It may:

- hide;
- reposition;
- or otherwise yield space to the current workflow.

### 3. User completes or exits the workflow

All required actions remain accessible throughout the interaction.

---

# 5. Requirements + Acceptance Criteria

## REL-01 - Group removal must not delete underlying content

**Priority: P0**

### Requirement

Removing a snippet from a group must affect only the relationship between that snippet and the selected group.

### Acceptance Criteria

When a user removes a snippet from a group:

- the snippet disappears from that group;
- the underlying snippet remains in the main library;
- snippet content remains unchanged;
- unrelated items remain unchanged;
- the action is not represented as permanent deletion.

---

## REL-02 - Permanent deletion must be explicitly destructive

**Priority: P0**

### Requirement

Permanent deletion and removal from a group must be represented as separate actions.

### Acceptance Criteria

When deleting the underlying snippet:

- the interface identifies the item being deleted;
- the action uses explicit destructive language;
- the user has an opportunity to cancel;
- the underlying snippet is deleted only after the destructive action is confirmed;
- associated references are cleaned up appropriately.

Example:

> ***Delete "Email Rewrite"?*** *This will permanently remove this snippet from PromptVault.*
>
> Cancel | Delete

---

## REL-03 - Floating controls must not obstruct active workflows

**Priority: P0/P1**

### Requirement

Persistent or floating creation controls must never obstruct contextual actions required to complete an active workflow.

### Acceptance Criteria

Verify that contextual actions remain accessible with:

- one item selected;
- multiple items selected;
- the user at the bottom of a long list;
- short panel heights;
- long content lists;
- all officially supported display/window configurations.

---

## REL-04 - Command Palette must expose one unambiguous active selection

**Priority: P1**

### Requirement

Only one item may visually represent the current active command-palette selection unless PromptVault explicitly enters a multi-select mode.

### Acceptance Criteria

No contradictory selected states appear during:

- mouse hover;
- keyboard navigation;
- mouse → keyboard transitions;

- keyboard → mouse transitions;
- scrolling;
- filtering;
- reopening the palette;
- rapid movement between results.

Engineering should determine whether the current defect originates from overlapping hover, focus, selected, or keyboard-navigation states.

---

## REL-05 - Supported keyboard interactions must behave predictably

**Priority: P1**

### Requirement

PromptVault should advertise only keyboard interactions that operate consistently across officially supported environments.

### Minimum Supported Set

- ↑ - previous result;
- ↓ - next result;
- Enter - activate selected result;
- Esc - dismiss active temporary surface.

### Acceptance Criteria

- all advertised interactions perform their documented action;
- unsupported shortcuts are not advertised;
- PromptVault shortcuts do not unexpectedly invoke unrelated browser actions where avoidable;
- behavior is verified across supported environments.

---

## REL-06 - Escape provides predictable dismissal

**Priority: P1**

### Requirement

Esc dismisses the highest currently active dismissible PromptVault surface.

### Acceptance Criteria

- the correct surface closes;
- no unrelated action is triggered;
- dismissal does not destroy user content;
- repeated Escape behavior is deterministic.

Example:

PromptVault → Command Palette → Confirmation Dialog

First Esc:

PromptVault → Command Palette

Second Esc:

PromptVault

## REL-07 - Focus should return appropriately after dismissal

**Priority: P2**

### Requirement

Where technically practical, closing a temporary PromptVault surface should return focus to the user's previous interaction context.

### Acceptance Criteria

Keyboard-oriented users should not be forced to re-establish their context with the mouse after dismissing a temporary interface.

---

## REL-08 - Primary navigation must expose persistent current state

**Priority: P2**

### Requirement

The active top-level section must remain visually distinguishable from inactive sections.

Applies to:

- Prompts;
- Snippets;
- Saves.

The implementation may use color, typography, background, border, or another accessible treatment.

The requirement is **state clarity**, not specifically the color orange.

---

# 6. Success Metrics

Because this is a stabilization initiative, success should primarily be measured through reliability and product quality rather than forced engagement or revenue metrics.

## Release Quality Bar

### Content integrity

**Target:** Zero known P0 defects where normal organizational actions unexpectedly destroy user-created content.

### Critical workflow reliability

The following workflows must pass defined release-blocking regression scenarios:

- save;
- retrieve;
- organize;
- select;
- remove;
- delete;
- insert.

**Target:** 100% pass rate across release-blocking scenarios.

### Keyboard reliability

Every officially supported keyboard interaction must pass the supported environment test matrix.

**Target:** 100% pass rate before release.

### Visual-state reliability

There should be no known high-severity instances of:

- contradictory selected states;
- inaccessible primary controls;
- overlapping critical controls;
- visibly impossible or conflicting UI states

within core workflows.

## Product / Research Outcome

After stabilization:

- users should be able to complete core workflows without encountering known trust-breaking defects;
- subsequent usability testing should be able to evaluate future features without major known defects acting as confounding variables.

## Secondary Signals

Where instrumentation exists, monitor:

- command palette → successful insertion rate;
- command-palette abandonment;
- abnormal deletion events;
- keyboard interaction failures;
- repeated interaction attempts;
- defect reports per active user;
- tester feedback mentioning reliability or confusion.

---

# 7. Technical Considerations

## State Management

The duplicate-highlight defect may indicate multiple UI states being represented simultaneously.

Engineering should explicitly define precedence between:

- hover;
- focus;
- keyboard selection;
- persistent selection.

The solution should address the underlying state logic rather than merely changing highlight colors.

## Deletion Semantics

Removing a group relationship and deleting the underlying content should be represented as separate operations.

Conceptually:

```
RemoveFromGroup(itemId, groupId)
```

should be distinct from:

```
DeleteItem(itemId)
```

This reduces the risk of organizational actions accidentally triggering destructive behavior.

## Keyboard Event Handling

Keyboard interactions must be evaluated against browser and operating-system behavior.

At minimum, test officially supported environments such as:

- Chrome on macOS;
- Chrome on Windows;

plus any additional environments PromptVault officially supports.

Shortcuts should not be introduced simply because they are intuitive if they conflict with browser or operating-system behavior.

## Focus Management

Opening and closing temporary surfaces should deliberately manage focus.

Particular attention should be given to:

- command-palette opening;
- initial search focus;
- keyboard navigation;
- modal dismissal;
- focus restoration.

## Regression Protection

Where practical, regression coverage should be added for behavior involving:

- deletion;
- group removal;
- selection;
- keyboard navigation;
- command-palette state;
- overlay positioning;
- dismissal.

The objective is not merely to fix current defects but to reduce the likelihood that future development reintroduces them.

# 8. Risks / Open Questions

## Risks

### Stabilization scope expands indefinitely

There will always be additional defects.

**Mitigation:** Use the defined scope, prioritization criteria, and exit conditions rather than attempting to reach "zero bugs."

### Cosmetic issues consume disproportionate engineering effort

Not every visual imperfection deserves stabilization priority.

**Mitigation:** Include visual defects only when they occur in a core workflow and materially affect comprehension, task completion, or perceived reliability.

### Reliability fixes create regressions elsewhere

State management, deletion behavior, keyboard interactions, and layout controls may affect shared product logic.

**Mitigation:** Add targeted regression coverage and run critical workflows end-to-end before release.

### Stabilization delays future product learning

Reliability work should not become a permanent reason to avoid product exploration.

**Mitigation:** Exit stabilization once the agreed quality bar is met.

## Open Questions

- What browsers and operating systems are officially supported?
- Is the group-removal behavior confirmed to delete the underlying object, or is this another state/presentation defect?
- Should permanent deletion always require confirmation?
- Should Undo be provided after non-destructive group removal?
- What state should receive priority when mouse hover and keyboard selection occur simultaneously?
- What should receive focus after the Command Palette closes?
- Which P2 visual issues materially affect user trust enough to justify inclusion?
- Which regression scenarios should block release versus generate follow-up defects?

# 9. Milestones + Exit Criteria

## Milestone 1 - Define Critical Workflows - XX weeks

- reproduce known defects;
- document save, retrieve, organize, insert, and delete flows;
- define supported environments;
- establish severity definitions;
- create release-blocking regression scenarios.

## Milestone 2 - Content Integrity & Workflow Blockers - XX weeks

Address:

- group-removal/deletion semantics;
- permanent deletion behavior;
- obstructed contextual controls;
- related regression coverage.

These issues are addressed first because they most directly threaten user trust and task completion.

## Milestone 3 - Interaction Reliability - XX weeks

Address:

- supported keyboard behavior;
- browser shortcut conflicts;
- Escape dismissal;
- focus management;
- command-palette selection state.

## Milestone 4 - High-Visibility Consistency - XX weeks

Address qualifying:

- contradictory highlights;
- primary navigation-state problems;
- severe layout/truncation defects;
- other visible issues meeting the stabilization criteria.

This milestone should **not** expand into a broader visual redesign.

## Milestone 5 - Regression & Dogfood - XX weeks

- run critical workflows end-to-end;
- test supported environments;
- internally dogfood the stabilized experience;
- fix release-blocking regressions.

## Milestone 6 - Baseline Usability Validation - XX weeks

Run focused usability testing against the stabilized product.

Evaluate whether users can:

- understand current system state;
- predict what actions will do;
- complete critical workflows;
- recover from mistakes;
- distinguish organizational actions from destructive actions;
- interact with the product without known reliability defects interfering with the study.

The results become the baseline for subsequent feature development and usability testing.

## Exit Criteria

PromptVault may transition from **Core Experience Stabilization** to **Feature Expansion** when:

- no known P0 reliability defects remain;
- all P1 defects affecting critical workflows are resolved or explicitly accepted;
- save → retrieve → organize → insert → delete workflows pass release-blocking regression scenarios;
- all documented keyboard interactions behave consistently across supported environments;
- no high-severity contradictory interface states remain in primary workflows;
- no critical controls are obstructed;
- organizational actions do not unexpectedly destroy underlying content;
- baseline usability testing identifies no new trust-breaking defect;
- remaining known defects are documented and do not compromise the agreed reliability baseline.