> **Affiliate Disclosure:**
> This document contains affiliate links. If you click an affiliate link, I may earn a commission. There is no additional cost to you.

The SaaS Product Adoption & Education Framework

Operational Templates for Reducing Friction and Scaling User Enablement

ASSET LIBRARY v1.0
OPERATIONAL GUIDE

A. Feature-to-Walkthrough Prompt

> **Type:** System Prompt / LLM Instruction Template
> **Usage:** Copy and paste this prompt when you need to generate a script for a specific feature. Replace the bracketed input data.

OPERATIONAL CONTEXT AND RATIONALE

The gap between a deployed feature and user adoption is often defined by the latency of instructional content. In resource-constrained environments, the traditional cycle—waiting for a stable UI, drafting a storyboard, seeking marketing approval, and producing high-fidelity assets—causes documentation drift. By the time the asset is released, the feature has often evolved, or the user has already encountered friction.

This asset, the **Feature-to-Walkthrough Prompt**, is designed to collapse the creation cycle by enforcing a rigid "Problem -> Action -> Outcome" logic. This structure is not merely a stylistic choice; it is a mechanism to reduce cognitive load. Users engaging with new features are typically in a state of high intrinsic load; they are trying to solve a problem while simultaneously learning a new interface. Extraneous load—generated by marketing fluff, "exciting" adjectives, or non-linear storytelling—competitively inhibits the processing of necessary technical information.

The prompt enforces a "builder-to-builder" tone. This ensures that the output is strictly functional. It treats the user not as a consumer to be persuaded, but as an operator to be instructed. This aligns with the "Keep a Changelog" philosophy, which emphasizes that logs and updates are for humans, not machines, but must remain distinct from marketing copy. By standardizing the input (feature context) and the output (linear script), this tool allows product operators to generate

accurate, validated scripts in seconds, ensuring that help centers and release videos are synchronous with code deployment.

> **Implementation reference (optional):** For teams standardizing video-based walkthrough production, a commonly used platform is https://www.synthesia.io/?via=Explore-Now.

Input Data / Input Variables (Provide All)

```
Feature Name: [Insert Name]
Technical Function: [Describe inputs, processing, and outputs]
User Permission Level: [Admin/Editor/Viewer]
Pre-conditions: [State required settings or prior data]
Raw Context:
User Role: [Admin / Editor / Viewer]
Interface Context:
Change Log Item: [Paste technical ticket or developer note]
```

Prompt Instructions (System Role + Objective)

```
System Role: You are a Technical Product Operator specializing in converting raw feature specifications into linear, operational walkthrough scripts. You do not write marketing copy. You do not tell stories. You map user inputs to system outputs.
Act as a Technical Writer for internal product documentation.
Objective: Transform a raw feature update (ticket, changelog, or brief) into a "Problem -> Action -> Outcome" script for video recording or documentation.
Convert the Input Data above into a step-by-step walkthrough script.
```

STRICT CONSTRAINTS (MUST BE ENFORCED)

Structure Enforcement (Problem -> Action -> Outcome)

- **Problem:** State the specific friction point or missing capability that triggered this feature. Use the format: "Previously, users [encountered specific limitation]." This grounds the user in the "Why" without selling.
- **Action:** List the exact literal clicks, keystrokes, or commands required. Do not summarize. Use imperative verbs (e.g., "Click," "Select," "Type").
- **Outcome:** Describe the immediate system verification of the action. What changes on screen? What data is returned? This provides the "feedback loop" necessary for confidence.

Structure: Use a linear "Action -> Result" format.

Vocabulary: Use only literal navigational verbs (Click, Select, Type, Toggle, Drag).

> **Prohibited:**
> • Do not use adjectives describing difficulty or quality (e.g., "easy," "intuitive," "seamless," "powerful").
> • NO adjectives of quality (e.g., "seamless," "easy," "powerful," "exciting").
> • NO future tense promises (e.g., "this will help you...").
> • NO rhetorical questions.
> • No Intros/Outros: Start immediately with the first action. End when the task is complete.
> • NO "In this video..." or "Today we are going to show you..."

Visual Cues: Indicate UI elements in brackets, e.g., **[Save Button]**.

- Bold interface elements (buttons, fields, menu items).
- Use code blocks for strings users must type.

OUTPUT FORMAT (REQUIRED)

- **Prerequisites:** Bulleted list of states required before starting.
- **Step-by-Step Script:** Numbered list.
- **Format:** "Navigate to [Page]. Click [Button]."
- **Validation:** One sentence describing the visual confirmation of success.

FORMATTING (ADDITIONAL REQUIRED FORMAT FROM RESEARCH 2)

- Use Markdown tables for step-by-step actions.
- Use code blocks for strings users must type.

Contextual Logic Library (Scenario Dictionary)

Use the following translation matrix to interpret raw developer inputs into operational script language.

Feature Type	Developer Input Example	Operational Script Translation Pattern
UI State Change	"Moved toggle to modal"	Action: Navigate to [Page]. Click. The modal appears. Outcome: The is visible in the top-right corner.
Data Validation	"Added regex check for email"	Action: Enter an invalid email format (e.g., "name@domain"). Outcome: The system displays the error message: "."
Performance	"Reduced query latency by 50ms"	Action: Load the. Outcome: The table data populates immediately. (Note: Do not quantify speed unless visible).
Permissions	"RBAC update for Read-Only"	Action: Log in as. Navigate to. Outcome: The button is greyed out/disabled.
Deprecation	"Removed legacy export"	Problem: The CSV export button in the sidebar is no longer supported. Action: Navigate to the.
Integration	"Webhook payload expansion"	Action: Trigger the [Event]. Check the payload log. Outcome: The JSON object now includes the [new_field] key.

Feature Type	Developer Input Example	Operational Script Translation Pattern
Bulk Action	"Batch delete added"	Action: Select multiple rows using the checkboxes. Click. Outcome: A confirmation dialog appears showing "[Number] items selected."

Output Generator (Copy and Fill)

```
Walkthrough Script: [Feature Name]

Context (Problem State)
Trigger:
Scope: Affects in [Interface Context].

Execution Flow (Action -> Outcome)

Step | User Action (Literal) | System Outcome (Visual Verification)
---- | --------------------- | ------------------------------------
01 | Navigate to [Menu Item] > ****. | The **** view loads.
02 | Locate the [Component Name] panel. | is visible.
03 | [Action from Logic Library]. | [Outcome from Logic Library].
04 | Click ****. | System displays toast notification: "[Message]".
05 | Refresh the page. | The status column updates to ****.
```

EDGE CASE NOTES (IF APPLICABLE)

- If [Condition X] exists, ensure is skipped.
- Note: This feature is hidden for.

Component Analysis and Logic

THE "PROBLEM -> ACTION -> OUTCOME" FRAMEWORK

This framework is derived from task-based documentation standards and cognitive load theory.

- **Problem:** This aligns with the "Pre-training Principle" in instructional design. By defining the limitation first, we prime the user's mental model for the solution. It answers "Why am I here?" without using marketing language.
- **Action:** This utilizes "imperative mood" instruction. Instructions must be binary: did the user perform the action or not? There is no room for interpretation.
- **Outcome:** This provides immediate "System Feedback." In software interaction, users require confirmation that their input was registered. By explicitly scripting the outcome (e.g., "The button turns green"), the operator ensures the video verifies the success state.

NEGATIVE CONSTRAINTS (THE "NO FLUFF" RULE)

Excluding adjectives and sales framing is critical for "builder-to-builder" communication.

- **Superlatives (e.g., "Seamless", "Easy"):** These are subjective. If a user struggles, describing the process as "easy" induces frustration and dissonance.
- **Future Tense (e.g., "This will allow..."):** Users read documentation to act now. Present tense ("This allows...") is actionable; future tense is speculative.
- **Rhetorical Questions:** These add conversational overhead (extraneous cognitive load) without adding information.

THE CONTEXTUAL LOGIC LIBRARY

This matrix serves as a translation layer between "Dev-Speak" and "User-Speak." Developers communicate in terms of implementation (e.g., "Added regex"), while operators must communicate in terms of verification (e.g., "Enter invalid email").

- **Data Validation:** This is often invisible. The script must force an error state to make the feature visible in a walkthrough.
- **Performance:** Speed is subjective. The script focuses on the absence of loading spinners rather than claiming "lightning fast" speeds.
- **Permissions:** Security features are defined by what cannot be done. The script instructs the operator to verify the disabled state.

IMPLEMENTATION GUIDE: HANDLING COMMON SCENARIOS

Scenario A: The Backend Update (Invisible Feature)
Challenge: A feature like "Improved API rate limiting" has no UI. *Operational Solution:* The prompt directs the user to the Outcome in the logs or error messages.

```
Input: "Rate limit increased to 100 req/min."
Script Output:
Action: Send 101 requests via Postman.
Outcome: Request 101 returns 429 Too Many Requests.
Video Strategy: Record the terminal/API client, not the main UI.
```

Scenario B: The UI Overhaul (Visual Volatility)
Challenge: A feature moves a button from the left sidebar to the top nav. *Operational Solution:* The prompt focuses on the new path.

```
Input: "Nav refactor."
Script Output:
Problem: The sidebar navigation has been deprecated to increase canvas space.
Action: Click the [Hamburger Menu] in the top-left.
Outcome: The navigation drawer slides out.
```

Scenario C: Complex Multi-User Workflows

Challenge: A feature requires Action A by an Admin and Action B by a User. *Operational Solution:* The prompt generates two distinct sequences.

```
Script Output:
Phase 1 (Admin): Enable setting. Outcome: Toggle is green.
Phase 2 (User): Refresh page. Outcome: New feature is visible.
Video Strategy: Use a "Split Screen" or "Persona Swap" title card between phases.
```

TROUBLESHOOTING THE PROMPT

- **Issue:** The output is too technical.
  Cause: The "Raw Context" input was likely a direct paste of a complex Git commit message.
  Fix: Pre-process the input. Instead of pasting the full code diff, paste the User Story from the ticket.
- **Issue:** The output lacks flow.
  Cause: The steps are too granular (e.g., "Move mouse to X", "Click X").
  Fix: Group actions. Modify the prompt instructions to "Group related clicks into a single step if they occur in the same modal."
- **Issue:** The output contains marketing language.
  Cause: The "Negative Constraints" were ignored or the LLM reverted to default "helpful assistant" mode.
  Fix: Re-run with the instruction: "Rewrite with extreme dryness. Remove all adjectives."

B. Modular Onboarding Sequence Template

> **Type:** Content Architecture Framework
> **Usage:** Use this structure to organize video/doc files. Isolate volatile UI components from stable conceptual components to minimize update debt.

OPERATIONAL CONTEXT AND RATIONALE

Monolithic onboarding videos (e.g., "The 20-minute Full Platform Tour") are operational liabilities. In a fast-shipping SaaS environment, a single UI change—such as moving a "Settings" button—renders the entire 20-minute asset obsolete. This leads to a binary choice: leave inaccurate content live (eroding trust) or re-record the entire video (burning resources).

This **Modular Onboarding Sequence Template** solves this by applying "Object-Oriented Content Design" principles. By treating content as discrete, interchangeable blocks , operators can swap out a 30-second "Action Block" without touching the "Intro" or "Concept" blocks. This dramatically reduces the "Blast Radius" of a product update.

The architecture mimics software engineering:

- **Abstraction:** Separating the concept (stable) from the interface (volatile).
- **Encapsulation:** Containing UI-specific instructions in isolated blocks.
- **Dependency Management:** Explicitly listing prerequisites to allow for dynamic playlist generation.

This approach prioritizes Clarity and Maintainability over cinematic flow. Users in B2B SaaS prefer accurate, searchable, and granular answers over high-production narratives.

ARCHITECTURE RULES

- **Atomicity:** Each module must be under 90 seconds.
- **Dependency:** Modules must be referenced by ID, not title, to allow title updates without link breaking.

> **Stability Tagging:**
> **[STABLE]** = Concepts, strategic "why," API logic. (Review every 6 months).
> **[VOLATILE]** = UI navigation, specific button placements, dashboard layouts. (Review every release).

SEQUENCE STRUCTURE:

- **Module 01: The Logic Model [STABLE]**
  Content: Explains the data object hierarchy or calculation logic. No UI shown.
  Visuals: Diagrams, Schematics, Flowcharts.
  Trigger for Update: Fundamental change in data model only.
- **Module 02: Configuration & Inputs [VOLATILE]**
  Content: Where to click to enable the feature. Input field definitions.
  Visuals: Screen capture of Settings/Input forms.
  Trigger for Update: UI redesign, new field added, menu move.
- **Module 03: Execution Workflow [VOLATILE]**
  Content: The active task. (e.g., "Running the report," "Sending the campaign").
  Visuals: Screen capture of the main workspace.
  Trigger for Update: Workflow steps change.
- **Module 04: Output Interpretation [STABLE]**
  Content: How to read the results, logs, or analytics.
  Visuals: Static screenshots of example outputs (annotated).
  Trigger for Update: New metrics added or removal of data points.

The Asset: Modular Onboarding Sequence Template

Purpose: Allow onboarding content to change in parts without rebuilding everything.

I. STRUCTURAL HIERARCHY

Level 1: The Container (The "Course")
Definition: The linear sequence assigned to a user role.
Maintenance Frequency: Low (Quarterly).
Composition: An ordered list of Level 2 Modules.
Example: "Admin Onboarding v3" (Contains Modules A, B, C).

Level 2: The Module (The "Topic")
Definition: A discrete functional outcome (e.g., "Inviting a User," "Configuring API Keys").
Maintenance Frequency: Medium (Monthly/Per Release).
Composition: 1 Intro Block + 1-3 Action Blocks + 1 Verification Block.
Constraint: Max length 4 minutes / 600 words.

Level 3: The Block (The "Atom")
Definition: The smallest unit of instruction (e.g., "The Login Screen," "The Dashboard Widget").
Maintenance Frequency: High (Weekly/Daily).
Versioning: Individually tagged.
Nature: Highly Volatile (contains UI specifics).

II. NAMING & VERSIONING CONVENTION

Use the following schema to tag assets in your CMS/LMS to ensure rapid retrieval and replacement.

Asset Level	Naming Pattern	Example	Version Rule
Container	ROLE_FLOW_v{Major}	ADMIN_ONBOARD_v2	Increment on full restructure.
Module	MOD_{Func}{ID}v{Major}.{Minor}	MOD_USER_INVITE_v1.4	Increment Minor on text tweaks; Major on UI overhaul.
Block	BLK{UI_Element}{ID}_{State}	BLK_SIDEBAR_NAV_EXPANDED	New ID for every UI change.

III. THE MODULAR TEMPLATE

Copy the markdown below to structure every onboarding module.

```
Module ID:
Status:
Last Verified: by [Operator Name]

Module Boundary (Meta-Data)
Prerequisites:
User Role:
Software Version: [Min. App Version]
Volatility Score: [High/Low] (High = UI changes frequently; keep text generic)

Intro Block (Stable)
Purpose: State the Job-to-be-Done (JTBD).
Script Anchor: "In this module, you will configure [Feature] to achieve [Outcome]."
Visual Asset: OR
Restriction: Do NOT show specific UI screens here. This ensures the intro remains valid even if the UI changes.
```

```
Action Block A (Volatile - The "Happy Path")
Input Trigger:
Step Sequence:
Action: [Literal Instruction] -> Visual:
Action: [Literal Instruction] -> Visual:
Action: [Literal Instruction] -> Visual:
Audio/Text Rule: Refer to elements by function ("The Save Button"), not just position
("Bottom right"), to extend asset life if buttons move slightly.

Action Block B (Volatile - The "Configuration")
Optional: Only include if configuration is distinct from the main action.
Step Sequence:
Action: [Literal Instruction] -> Visual:
Action: [Literal Instruction] -> Visual:

Verification Block (Stable)
Purpose: Confirm success state.
Script Anchor: "When complete, will display [Indicator]."
Visual Asset:
```

IV. COMPONENT LIBRARY (STANDARDIZED BLOCKS)

Use these pre-defined block structures for common SaaS patterns to avoid rewriting standard procedures.

Block Type: Authentication (SSO/Login)
Volatility: Low
Standard Script: "Enter your credentials provided by your administrator. If Single Sign-On is enabled, select your identity provider."
Visual Requirement: Blur the email/password fields in recording.

Block Type: Data Table Management
Volatility: Medium (Column headers change)
Standard Script: "Use the filter bar to narrow results. Columns can be sorted by clicking the header. Actions for individual rows are located in the ellipsis menu."
Maintenance Note: If a new column is added, update only the visual asset; the script often remains valid.

Block Type: API Key Generation
Volatility: High (Security UI changes often)
Standard Script: "Navigate to Settings, then API Access. Generate a new token. Copy this token immediately; it will not be shown again."
Visual Requirement: NEVER record a real production key. Use a test environment key or obscure the string in post-production.

V. UPDATE LOGIC (THE "REWORK KILLER")

When a product update releases, execute this logic to determine minimum viable update work:

Analyze the Changelog: Does the change affect the Concept or the Click-path?

- **Concept Change:**
  Action: Rewrite "Intro Block" and "Verification Block."
  Effort: High (Requires new conceptual framing).
- **Click-path Change (Button moved, Color changed):**
  Action: Replace "Visual Asset" in Action Block A/B. Keep "Intro Block" and audio

track if possible.
Effort: Low (Reshoot screen only).

- **Field Change (New input required):**
  Action: Record discrete insert clip for the new field. Splice into Action Block.
  Effort: Medium.

Component Analysis and Logic

THE "BLOCK" ARCHITECTURE

The "Block" is the atomic unit. By enforcing a distinction between Intro Blocks (Stable) and Action Blocks (Volatile), we optimize for reuse.

- **Stable Content:** Concepts, "Why" explanations, and JTBD statements rarely change. A video intro stating "API keys allow secure communication between servers" remains true regardless of where the "Generate Key" button is located.
- **Volatile Content:** UI specifics. By isolating these into "Action Blocks," we can surgically replace 15 seconds of video without re-recording the voiceover for the concept.

NAMING & VERSIONING

Adopting a "SemVer" (Semantic Versioning) approach for content prevents version conflict.

- **Container v2:** Represents a major curriculum overhaul.
- **Module v1.4:** Represents a minor tweak (e.g., text clarification) that doesn't break the container.
- **Block IDs:** Using unique IDs allows for programmatic retrieval. If an LMS supports dynamic playlists, you can reference BLK_LOGIN_LATEST to always pull the newest login screen recording.

VOLATILITY SCORING

This is a risk management tool.

- **High Volatility:** Areas of the product actively under development (e.g., "Beta Features"). For these, the template advises keeping voiceovers extremely generic (e.g., "Configure the settings as shown on screen") so the audio doesn't need re-recording when the visual changes.
- **Low Volatility:** Mature features (e.g., "Logout"). These can have higher production value and specific voiceovers.

IMPLEMENTATION GUIDE: BUILDING FOR SCALE

Strategy for "Greenfield" Implementation
If starting from scratch:

1. Audit: List all 20 core user tasks.
2. Factor: Identify shared blocks (e.g., 15 tasks require "Login").
3. Record: Produce the "Shared Blocks" first (Login, Navigation).
4. Assemble: Build unique "Action Blocks" for specific tasks.
5. Compile: Stitch blocks into Modules.

Strategy for "Brownfield" Implementation (Refactoring)
If dealing with a library of monolithic videos:

1. Slice: Take existing long videos and cut them at the conceptual boundaries (between "Why" and "How").
2. Tag: Apply the naming convention to these slices.
3. Replace: When a feature updates, only re-record the specific slice (Block) that broke.

ADVANCED SCENARIOS

Scenario: The "White-Label" Product
Challenge: You ship the same software to 5 different enterprise clients with different branding colors. *Solution:*

- Keep Intro Blocks generic (no UI shown).
- Record Action Blocks on a "Neutral" theme (grayscale).
- Use the Clarity-over-Charisma Guide to enforce tight cropping (Zoom: 200%). By cropping tight on the functionality, you often exclude the branded headers/sidebars, making the video reusable across clients.

Scenario: The "Rapid Iteration" Beta
Challenge: A feature changes UI every week. *Solution:*

- Use Text-to-Speech (TTS) for the audio track in the Action Block.
- Use the template's Volatility Score: High.
- This allows you to regenerate the audio instantly by typing new text, rather than booking a voice actor or setting up a mic.

C. Release Update Script Prompt

> **Type:** Change Communication Template
> **Usage:** Use to generate patch notes or update video scripts.

OPERATIONAL CONTEXT AND RATIONALE

Release notes often suffer from a low signal-to-noise ratio. Marketing teams want to "spin" every bug fix as a "platform enhancement," while developers want to dump raw Git commit logs. Neither serves the user. The user's primary question is: "Does this break my workflow, or does it make it better?"

This **Release Update Script Prompt** is designed to act as a filter. It ingests raw technical data and outputs a prioritized, utility-focused script. It draws on "Keep a Changelog" principles —sorting by importance, clearly labeling breaking changes, and adhering to strict date/versioning standards.

The prompt enforces a "No Fluff" policy. Emotional framing ("We are thrilled to announce...") is explicitly banned. In builder-to-builder communication, "thrilled" is noise; "latency reduced by 20%" is signal. The goal is to respect the user's limited time and allow them to parse the update for relevance within seconds.

Input Data (Provide All)

```
Version Number:
Release Date:
Raw Changelog Items: [List of items]
Change Log Item: [Paste technical ticket or developer note]
Impacted User Segment: [Who sees this?]
Deprecations: [What was removed?]
```

Prompt Instructions (System Role + Objective)

```
System Role: You are a Release Communications Officer. Your goal is to inform, not sell. You strip away marketing fluff to deliver raw utility to existing users.
Prompt Instructions: Draft a Release Update notification based on the input data.
Objective: Convert a list of technical changes into a categorized, user-centric release script.
```

TONE RULES (STRICT)

- **Neutrality:** No celebratory language (avoid "excited," "thrilled," "finally").
- **No fluff:** Do not apologize. Do not "excite." Just inform.
- **Brevity:** Maximum 3 sentences per change.
- **Justification:** Do not explain why we built it, only what it does.
- No emotional framing.
- No justification or celebration.
- This is informational, not promotional.

FILTERING & CATEGORIZATION LOGIC

Apply the following taxonomy to the raw input. If an item is "Internal/Chore," discard it unless it creates a visible performance boost.

Tag	Definition	Script Tone
****	New capability added.	Direct: "You can now..."
[FIX]	Bug resolved.	Neutral: "Resolved an issue where..."
****	Existing behavior changed.	Advisory: "Note that X is now Y..."
****	Feature removed/deprecated.	Critical: "Action required: X is gone."
****	Security update.	Reassuring: "Security patches applied."

REQUIRED OUTPUT SECTIONS (MUST BE PRESENT)

The Change (Statement of Fact):
Constraint: Use the format: "The [Feature Name] now supports [Function]." or " The [Element] has moved to [Location]."

Operational Impact (The 'So What'):
Constraint: specific operational changes only. Example: "Reports will now take 30 seconds longer to generate due to increased data granularity."

Required Action (Immediate Task):
Constraint: Binary instruction. Example: "No action required" or "Update API key by [Date]."

Script Generation Template

```
 Headline: Release [Version] –

 Section 1: The "Must Know" (Breaking Changes & Deprecations)
 Trigger: Any **** or **** tags.
 Script Structure: "Attention: The [Old Feature] has been replaced by [New Feature]. You
must [Action] by to avoid disruption."

 Section 2: New Capabilities (The "What")
 Trigger: **** tags.
 Script Structure: "We added [Feature Name].
 Where: Find it in [Menu Location].
 Why: Use this to.
 How: [1-sentence instruction]."

 Section 3: Fixes & Polish (The "details")
 Trigger: [FIX] and **** tags.
 Script Structure: "This release also resolves [Issue A] and. Full patch notes are available
in the text below."

 Section 4: Call to Verification
 Script Structure: "This update is live. Please verify your settings."
```

Translation Matrix (Tech-to-User)

Use this table to rewrite developer jargon into operator instructions.

Developer Jargon	Release Script Language
"Refactored backend for latency reduction"	"Reports now load faster."
"Implemented OAuth2.0 endpoints"	"You can now log in with Google/Microsoft."
"Fixed race condition in cart"	"Resolved an error when adding multiple items quickly."
"Deprecating v1 API"	"The v1 API will stop working on. Switch to v2."
"Nuked the legacy sidebar"	"The sidebar has been updated to the new layout."
"Bumped dependency versions"	(Do not include in script unless security critical)
"Sanitized input fields"	"Improved security on form submissions."

EXECUTION INSTRUCTIONS

- Paste the Raw Changelog.
- Apply Filtering Logic (Discard chores).
- Map remaining items to Script Generation Template.
- Apply Translation Matrix to specific terms.
- Output the final bulleted script.

Component Analysis and Logic

THE FILTERING LOGIC (TRIAGE)

The prompt forces a triage of information based on user impact, not developer effort.

- ** **(Deprecation):**** This is prioritized above all else (Section 1). Losing functionality is a "critical incident" for a user's workflow. The script must warn before it informs.
- ** **(New Features):**** Placed second. While "exciting," they are optional. The user can ignore them and still do their job.
- **[FIX] (Bug Fixes):** Placed third. Unless the bug was blocking a critical workflow, users generally expect software to work. Celebrating bug fixes too loudly can imply the software was previously broken.

THE TONE MATRIX

This enforces "Neutrality."

- Developer: "Nuked the legacy sidebar." (Aggressive, internal slang).
- Marketing: "Experience our stunning new navigation!" (Hype, vague).

- Operator (Target): "The sidebar has been updated to the new layout." (Factual, locates the change).

BREAKING CHANGES VS. DEPRECATIONS

The prompt distinguishes between immediate breaks and future breaks.

- Breaking Change: "The API response format has changed." -> Script: "Update your integrations immediately."
- Deprecation: "The old API will work for 3 months." -> Script: "Plan your migration by."

IMPLEMENTATION GUIDE: DISTRIBUTION CHANNELS

The "Video" Script
When using this prompt for a Loom/Video update:
Visuals: Section 1 (Must Know) requires a static slide with large text (e.g., "DEADLINE: JUNE 30"). Section 2 (New Capabilities) requires screen recording.
Pacing: Read the bullet points with 0.5s pauses between items.

The "In-App" Modal
When using this prompt for an in-app changelog (e.g., Beamer, Headway):
Links: The "How" section of the script should link directly to the Modular Onboarding asset (Asset B) for that feature.
Formatting: Use the Markdown headers provided in the template.

The "Email" Blast
Subject Line: Use the "Headline" generated by the prompt.
Body: Paste Section 1 and Section 2. Omit Section 3 (Fixes) and link to the full changelog instead to keep the email concise.

TROUBLESHOOTING THE PROMPT

- **Issue:** The script is too long.
  Cause: Too many [FIX] items included.
  Fix: Modify the input instructions to "Group all minor UI fixes into a single bullet point: 'Various UI polish and minor bug fixes'."
- **Issue:** The "Why" is missing.
  Cause: Developer notes often say "Added X" without explaining the use case.
  Fix: The operator must inject the "Why" based on the ticket. The prompt asks for "Solve Specific Problem" – if this is missing from the input, the prompt should flag it as a missing variable.

D. Clarity-over-Charisma Guide

> **Type:** Production Standard
> **Usage:** Reference sheet for video editors and creators.
> **Purpose:** Prevent uncanny valley and overproduction in product videos. Must keep videos usable, not impressive.

OPERATIONAL CONTEXT AND RATIONALE

There is a dangerous trend in SaaS product education: the "Uncanny Valley" of over-production. High-fidelity, marketing-style videos with upbeat music, fast cuts, and enthusiastic voiceovers often alienate technical users. They increase "Extraneous Cognitive Load" —the user has to expend mental energy filtering out the "vibe" to find the "instruction."

Furthermore, highly polished videos are brittle. If a video relies on complex after-effects, color grading, and a specific voice actor, updating it requires a full production crew. This leads to outdated content.

This **Clarity-over-Charisma Guide** establishes a "Minimum Viable Quality" standard. It prioritizes Utility, Editability, and Cognitive Efficiency. It sets strict limits on visual density and pacing to ensure that the content is accessible to all users (including non-native speakers and those with accessibility needs) and easy for the product team to update.

PACING AND AUDIO

- **Speech Rate:** Maintain 130–150 words per minute. Faster increases cognitive load; slower indicates marketing fluff.
- **Pauses:** Insert a 0.5-second silence between distinct actions. Insert a 1.0-second silence between screen transitions.
- **Silence:** Do not fill dead air with background music. If music is mandatory, use a repetitive, mid-tempo loop with no melody spikes, mixed at -25db relative to voice.

I. Cognitive Load & Pacing Standards

THE "140-160 WPM" RULE

- **Standard:** Voiceover pacing must fall between 140 and 160 Words Per Minute.
- **Why:** Slower than 140 feels condescending; faster than 160 degrades retention.
- **Calculation:** A 100-word script should take ~40 seconds.
- **Silence is Syntax:**
    - Insert a 1.5-second pause after completing a complex action (e.g., submitting a form) before speaking the next instruction.
    - Insert a 0.5-second pause between sentences.

VISUAL DENSITY CAP

- **Rule:** Max 3 visual focal points per scene.
- **Bad:** Showing the entire dashboard while talking about one button.
- **Good:** Zooming/Cropping to show the button, the label, and the cursor.
- **Signaling:** Use explicit visual cues (bounding boxes, highlights) to guide the eye. Do not rely on mouse movement alone.

THE "6-MINUTE" HARD LIMIT

- **Constraint:** No single video asset shall exceed 6 minutes.
- **Data:** Engagement drops 50% after minute 6.

- **Action:** If a script exceeds 800 words, split it into two modular assets (e.g., "Setup" and "Advanced Configuration").

VISUAL DENSITY & SCREEN RECORDING

- **Resolution:** Record at 1080p minimum. Crop to the specific active window or div, not the full desktop.

II. Technical Configuration Reference (The "Look")

CANVAS & RESOLUTION

- **Recording Resolution:** 1920x1080 (1080p).
- **Do NOT record in 4K:** 4K downscaled to a laptop screen makes UI text unreadable.
- **Do NOT record ultrawide:** Aspect ratio must be 16:9 standard.
- **UI Scaling:** Set OS Display Scaling to 125% or 150% before recording. This ensures button labels are legible on mobile devices/small embed windows.

MOUSE MOVEMENT:

- Disable mouse acceleration.
- Edit out "mouse wandering" (erratic movements while thinking).
- Cursor path must be linear: A to B.

CURSOR SPECIFICATIONS

- **Size:** Scale cursor to 200-250% of default size.
- **Highlight:** Apply a semi-transparent yellow/green halo (opacity 40%) around the cursor.
- **Behavior:**
    - Idle: Hide cursor when not performing an action.
    - Motion: Use "smoothed" mouse pathing in post-production (e.g., Camtasia "Cursor Smoothing" or ScreenFlow "Smooth"). Jittery mouse movement indicates nervousness/amateur production.

Zooming: Do not pan/zoom continuously. Cut to the zoomed view instead of animating the zoom.

AUDIO ENGINEERING (THE "NO-HISS" STANDARD)

- **Sample Rate:** 44.1kHz or 48kHz (16-bit or 24-bit).
- **Noise Floor:** Must be below -60dB.
- **Processing Chain:**
    - Noise Gate: Threshold -45dB (Cuts silence).

    - Compressor: Ratio 3:1, Threshold -18dB (Evens out volume spikes).
    - EQ: High-pass filter at 80Hz (Removes rumble).
- **Tone:** Monotone is acceptable; "Radio DJ" voice is forbidden. Prioritize enunciation over energy.

SCRIPT CONSTRAINTS

- **Sentence Length:** Maximum 20 words per sentence.
- **Active Voice:** Always use [Subject] [Verb] [Object].
  Bad: "The settings can be changed by clicking here."
  Good: "Click Settings to change defaults."
- **No Analogies:** Do not use metaphors (e.g., "Think of this like a filing cabinet").
  Describe the digital object literally (e.g., "This is the database directory").

III. Screen Action Protocol

THE "PARK AND READ" METHOD

Never speak while moving the mouse.
Sequence:

1. Speak: "Click the Settings icon."
2. Move Mouse: Move cursor to icon.
3. Action: Click.
4. Pause: Wait for UI update.
5. Speak: "Select the User tab."

ZOOM/PAN LOGIC

- **Jump Cuts vs. Pans:** Prefer Jump Cuts (instant zoom) over smooth pans. Smooth pans induce motion sickness and waste seconds.
- **Zoom Depth:**
    - Full Screen (100%): Establishing context (Start of video).
    - Action Area (200%): Filling forms, reading lists.
    - Detail View (400%): Checking checkboxes, reading error messages.

TEXT READABILITY (FLESCH-KINCAID)

- **On-Screen Text (Callouts):** Must score Grade 8 or lower.
- **Sentence Length:** Max 20 words per bullet point.
- **Duration:** Text must remain on screen long enough to be read twice (approx. 0.3 seconds per word).

IV. QA Checklist for Assets

Before shipping any asset, verify against this binary checklist:

Check	Requirement	Pass/Fail
Pacing	Is WPM between 140-160?	[ ]
Legibility	Is UI text readable on a mobile screen (simulated)?	[ ]
Cursor	Is cursor smoothed and scaled?	[ ]
Logic	Does the visual strictly match the audio (Park and Read)?	[ ]
Metadata	Is the file named correctly (e.g., MOD_LOGIN_v2.mp4)?	[ ]
Cleanliness	Are test accounts/sensitive data obscured?	[ ]
Silence	Is there 0.5s silence at the absolute start and end?	[ ]

Component Analysis and Logic

THE PHYSICS OF ATTENTION (VISUAL DENSITY)

The "3 Focal Points" rule is derived from the "Spatial Contiguity Principle." When eyes have to scan a 4K screen to find a 16px button, learning stops.

Implementation: By scaling the OS UI to 150%, we artificially lower the resolution, forcing elements to be larger relative to the frame. This makes the video legible even when compressed into a small <iframe> on a Help Center article.

THE "PARK AND READ" PROTOCOL

This separates the visual channel from the auditory channel.

- **Synchronous Presentation:** If you move the mouse while talking, the user splits attention between tracking the motion and decoding the speech.
- **Asynchronous Presentation (Park and Read):**
  Audio: "Click Settings." (Auditory Channel Active).
  Visual: Mouse moves. (Visual Channel Active).
  This toggling ensures 100% of cognitive resources are available for each distinct input.

AUDIO ENGINEERING AS UTILITY

The "Radio DJ" voice is discouraged because it adds personality, which is a "brand" element.

- **Builder Tone:** The voice should sound like a competent colleague sitting next to you. It should be flat, clear, and paced.

- **Technical Spec:** The noise gate and compression settings ensure that even if recorded in a less-than-perfect room, the audio sounds "tight" and professional without requiring a studio.

IMPLEMENTATION GUIDE: THE "GOOD ENOUGH" STUDIO

Hardware:

- Any USB Dynamic Microphone (e.g., ATR2100x, Samson Q2U). Avoid condenser mics in untreated rooms.
- Standard Laptop Webcam (for intros only).

Software:

- Screen Recording: OBS (Open Broadcaster Software) for raw capture.
- Editing: Descript (for text-based audio editing) or Camtasia (for cursor effects).

> **AI Alternative:** If you prefer to skip the recording studio entirely, you can use AI video generation platforms. Review Synthesia here: https://www.synthesia.io/?via=Explore-Now

Workflow:

1. Scripting: Use Asset A.
2. Audio: Record the voiceover first. This sets the timing.
3. Video: Record the screen actions while listening to the voiceover (or match them in post).
4. Assembly: Import to editor. Apply "Cursor Smoothing." Add "Zoom/Pan" cuts. Export.

ACCESSIBILITY AND INCLUSIVITY

Color Blindness:
Do not rely on color alone (e.g., "Click the red button"). Say "Click the Delete button."
Ensure high contrast for text overlays (White text on Black background).

Captions:
All videos must have "burned-in" captions or a synchronized .srt file.
Captions must match the script exactly (verbatim), as technical instructions cannot be paraphrased without risk of error.

Localization Readiness:
By keeping the voiceover separate from the screen recording (Park and Read), you can easily dub the audio into other languages without re-recording the screen, provided the UI is recognizable.