Wholesome Library v2

Story Generation Pipeline

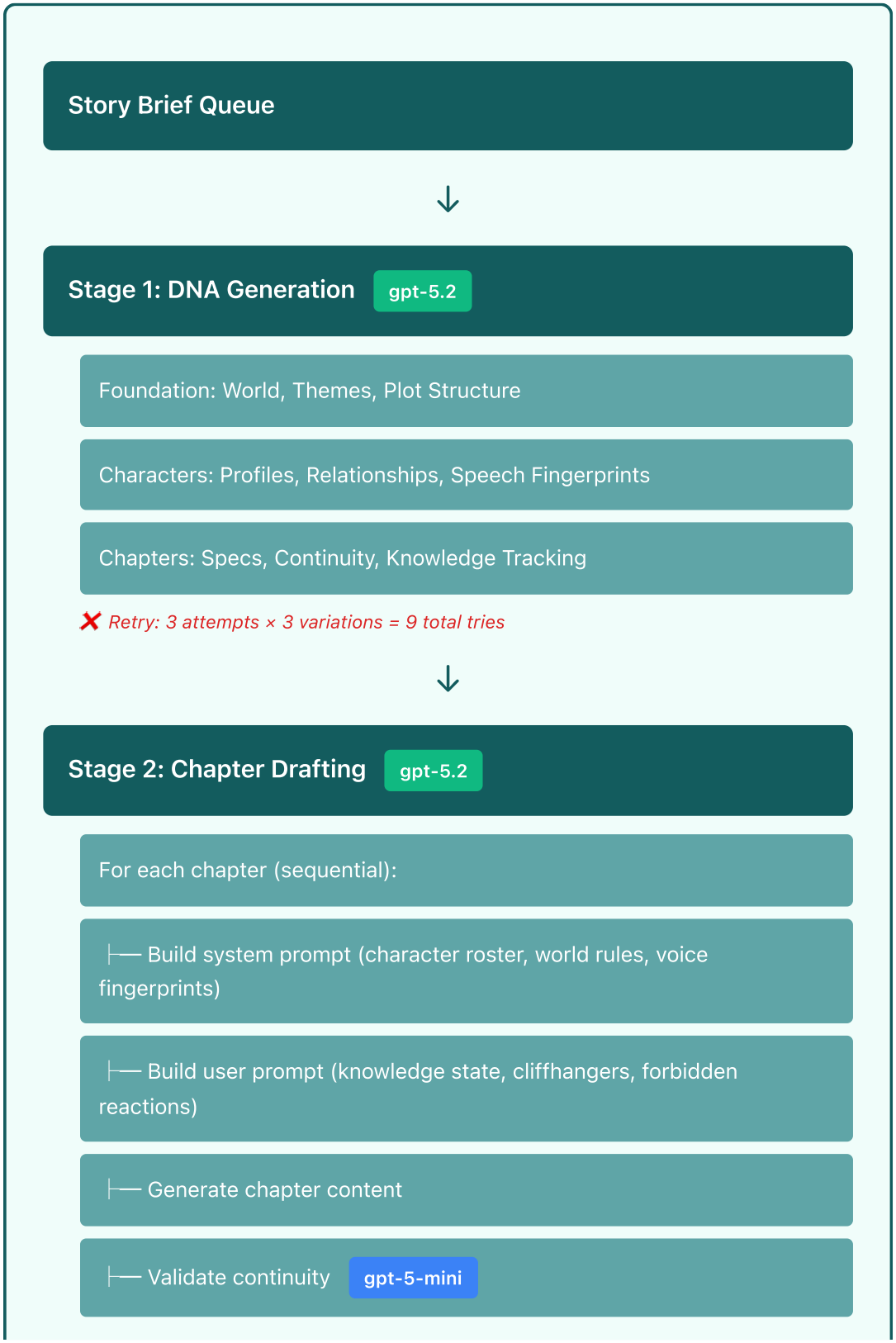
Technical Documentation & Architecture Review

Date: February 7, 2026

Version: 2.0 (V3 Sequential Enhanced DNA)

Model: GPT-5.2 (generation) + GPT-5-mini (validation)

Pipeline Overview Diagram



└── If violations: regenerate with constraints

❌ *Detect gaps: throw if chapters missing*

↓

Stage 3: AI Editor `gpt-5.2`

Polish grammar, fix AI artifacts, flag concerns

↓

Stage 4-6: QA Triple Check `gpt-5-mini`

Quality Score (0-100, 5 dimensions)

Safety Scan (binary pass/fail, 8 criteria)

Values Check (1-5 scale, 6 dimensions)

↓

Stage 7: Cover Art `Nano Banana`

Async workflow: create task → poll → download

❌ *Fallback: genre template if fails*

↓

⚠️ **Decision: Auto-approve / Editor Queue / Reject**

↓

Save to Supabase

stories, chapters, story_dna tables

Stage-by-Stage Detail

Stage 1: DNA Generation (3-part sequential)

Part 1A: Foundation (World, Themes, Plot)

Input: StoryBrief (reading_level, genre, primary_virtue, themes, target_chapters)

Process: Generate world bible, plot structure, emotional stakes

> ***System Prompt Strategy:*** *"You are a master story architect. Create rich, engaging story foundations."*
> ***User Prompt Strategy:*** *"Create foundation for [reading_level] [genre] story. Characters: [names]. Primary virtue: [virtue]. Themes: [themes]."*
> ***Key Constraint:*** *NO example storylines (prevents Bug #4 cookie-cutter plots)*

Output: { worldBible, plotStructure, emotionalStakes }

Model: gpt-5.2 (better at complex constraints, creative within boundaries)

Error Handling: Retry with variations (3 attempts × 3 prompt variations = 9 tries total)

Cost Estimate: ~3K tokens per attempt (~15K total with retries)

Part 1B: Characters & Relationships

Input: StoryBrief + compressed foundation summary

Process: Generate character profiles with speech fingerprints, NOT example dialogue

> ***System Prompt Strategy:*** *"You are a character development expert. Create nuanced, memorable characters with distinct voices. NEVER include example dialogue - use speech fingerprints instead."*
> ***User Prompt Strategy:*** *"Create deep character profiles. Define voice through SPEECH FINGERPRINTS (patterns, phrases, tells). NEVER include example dialogue snippets."*
> ***Key Constraints:***
>
> - *pronouns field LOCKED per character (prevents Bug #1 pronoun chaos)*

- *archetype must be: protagonist, foil, mentor, ally, rival*
- *speechStyle.patterns: HOW they talk (NOT what they say)*
- *speechStyle.phrases: unique words (NOT full sentences)*
- *NO example dialogue (prevents Bug #3 cookie-cutter stories)*

Output: { characters, characterTensions }

Model: gpt-5.2

Cost Estimate: ~3K tokens

Part 1C: Chapters & Continuity

Input: StoryBrief + compressed foundation + compressed characters

Process: Generate chapter specs with knowledge tracking, cliffhanger resolution plans

System Prompt Strategy: *"You are a story continuity expert. Create chapters with perfect flow and no plot holes. Track character knowledge carefully."*
User Prompt Strategy: *"Design [N] chapters with perfect continuity. Track character knowledge progression. Plan cliffhanger resolutions BEFORE writing them."*
Key Constraints:

- *Knowledge progression tracked per chapter (prevents Bug #5 knowledge amnesia)*
- *Cliffhanger resolution plans required (prevents Bug #6 cliffhanger amnesia)*
- *Forbidden reactions specified (prevents Bug #7 wrong reactions)*

Output: { chapterSpecs[], chapterTransitions[], storyState }

Model: gpt-5.2

Warning: May hit token limit (max 6K output), finish_reason checked for truncation

Cost Estimate: ~6K tokens

Stage 2: Chapter Drafting with Continuity Validation

Input: StoryDNA + StoryBrief

Process: Generate each chapter sequentially with inter-chapter validation

> ***System Prompt Strategy:***
>
> - *Character roster with LOCKED pronouns (prevents Bug #1)*
> - *"ONLY use these characters. Do NOT introduce new named characters." (prevents Bug #2)*
> - *Speech fingerprints (HOW they talk) without example dialogue (prevents Bug #3)*
> - *World rules with consequences*
>
> ***User Prompt Strategy:***
>
> - *Character knowledge state: "Max knows: [list]. Does NOT know: [list]" (prevents Bug #5)*
> - *Cliffhanger resolution plan from previous chapter (prevents Bug #6)*
> - *Forbidden reactions: "Riley must NOT react shocked to X (reason: already knows)" (prevents Bug #7)*
> - *Previous chapter ending (continuity anchor)*

Validation (gpt-5-mini): After each chapter, check for:

1. Pronoun consistency with character roster
2. New named characters not in original cast
3. Characters reacting to unknown information
4. Proper cliffhanger resolution
5. Contradictions to world rules

Output: Chapter[] (with content, wordCount fields populated)

Model: gpt-5.2 (generation), gpt-5-mini (validation)

Error Handling: If violations detected, regenerate with violations as constraints

Gap Detection: Throws error if chapter numbers incomplete

Cost Estimate: ~5 chapters × (3K gen + 1K validation) = ~20K tokens

Stage 3: AI Editor

Input: StoryDNA + Chapter[]

Process: Polish pass for technical quality, flag concerns (NOT plot changes)

> ***What to FIX:*** *Grammar, spelling, AI artifacts, awkward phrasing, transitions, formatting*
> ***What to FLAG (don't fix):*** *Values concerns, safety questions, character inconsistencies*
> ***What NOT to change:*** *Plot, character voices, narrative choices*

Output: { editedChapters[], changesApplied[], flaggedConcerns[] }

Model: gpt-5.2

Cost Estimate: ~8K tokens

Stage 4: Quality Check (0-100 score)

Input: StoryDNA + editedChapters[]

Process: Score 5 dimensions, sum to 0-100

Dimension	Max Points	Criteria
Narrative Coherence	25	Plot makes sense, smooth transitions, complete arc
Character Consistency	20	Same names/pronouns, clear arcs
Age Appropriateness	20	Vocabulary, complexity match reading level
Engagement	20	Compelling hooks, satisfying resolution
Technical Quality	15	Grammar, spelling, word count

Pass Threshold: ≥70

Model: gpt-5-mini (cheaper for QA)

Cost Estimate: ~3K tokens

Stage 5: Safety Scan (binary pass/fail)

Input: StoryDNA + editedChapters[]

Process: Check 8 safety criteria, ANY fail = auto-reject

#	Safety Criterion
1	No violence beyond age level
2	No death of main characters (early/independent readers)
3	No mature romantic content
4	No substance references
5	No self-harm or dangerous behavior
6	No discriminatory content
7	No horror or extreme fear elements
8	No profanity or crude language

Output: { passed: boolean, flags[], issues[] }

Model: gpt-5-mini

Cost Estimate: ~3K tokens

Stage 6: Values Check (1-5 scale)

Input: StoryDNA + editedChapters[]

Process: Score 6 dimensions on 1-5 scale, average

Dimension	1 (Poor)	5 (Excellent)
Positive Role Models	Poor examples	Virtuous, relatable protagonists
Consequence Logic	Illogical	Natural, appropriate consequences

Conflict Resolution	Violence/negativity	Positive means
Authority Respect	Negative portrayal	Parents/mentors shown positively
Virtue Integration	Preachy/forced	Naturally woven
Hopeful Ending	Negative/bleak	Uplifting, growth-oriented

Pass Threshold: Average ≥3.0

Model: gpt-5-mini

Cost Estimate: ~3K tokens

Stage 7: Cover Art Generation

Input: StoryDNA

Process: Async workflow with Nano Banana (kie.ai)

1. **Create Task:** POST to /createTask with prompt, returns taskId
2. **Poll (30 attempts × 2s):** GET /recordInfo?taskId=X until state=success
3. **Download:** Fetch image from resultUrls[0], save to /public/covers/

> `Prompt Template: "A whimsical children's book cover illustration featuring {characters} in {setting}. {atmosphere} atmosphere. {genre} style. Colorful, friendly, age-appropriate for {ageRange} year-olds. Professional book cover quality." Model: google/nano-banana (2¢/image) Aspect: 3:4 (portrait, book cover format)`

Output: { success, imageUrl, localPath, fallbackUsed }

Error Handling: Fallback to genre template if API fails

Cost: $0.02 per cover

V3 Consistency Engine — Bug Prevention Matrix

These are REAL bugs from V1 production. V3 fixes all of them:

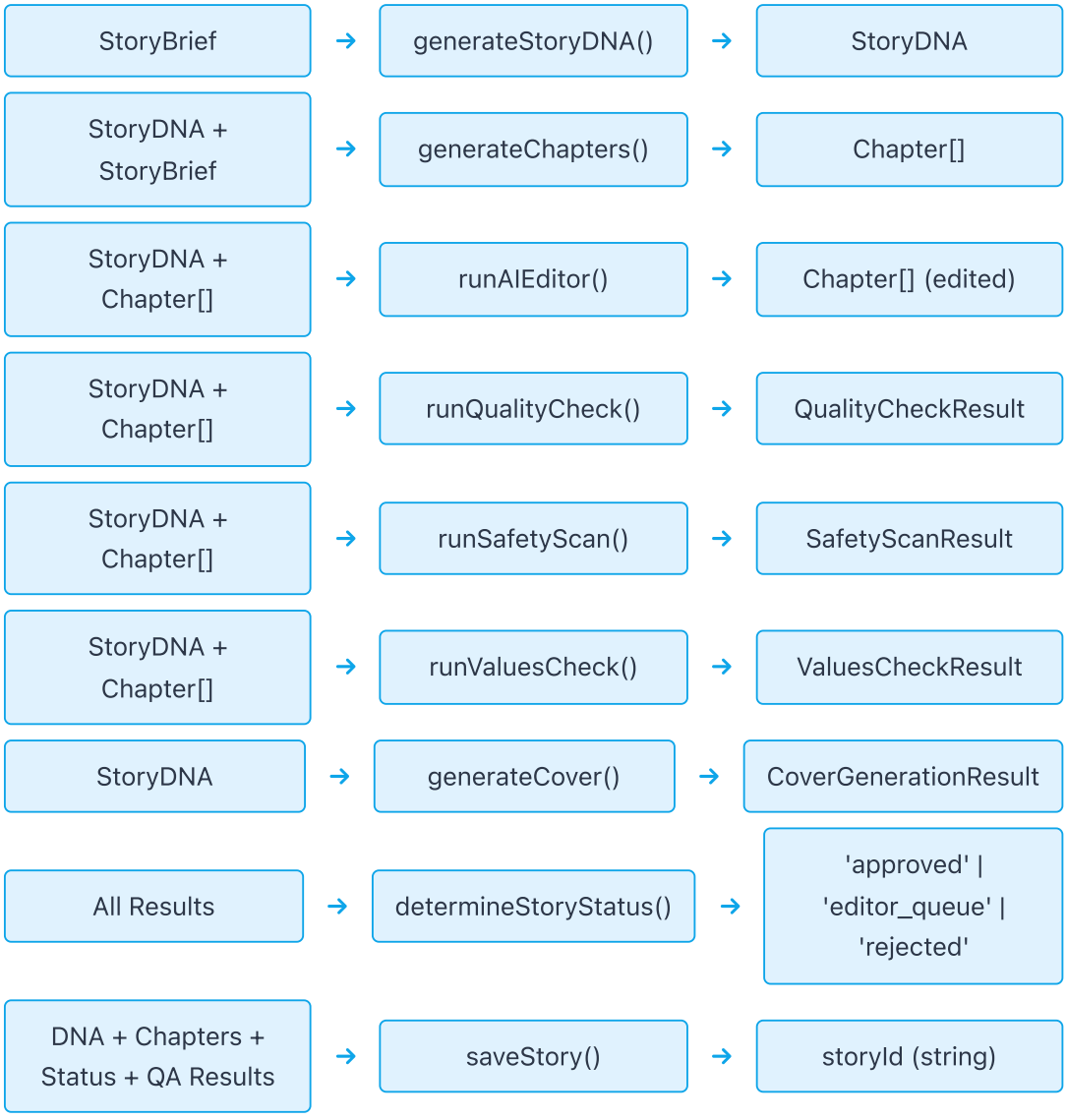
Bug	Symptom	Prevention Mechanism	File:Location
Bug #1: Pronoun Chaos	Jordan started "she/her" but became "he/him" in Ch3	**LOCKED pronouns in character roster.** Every chapter prompt includes: "Jordan (she/her), Alex (he/him)". Continuity validation checks pronoun consistency.	chapter-generator.ts:buildChapterSystemPrompt() Validation in validateChapterContinuity()
Bug #2: Spontaneous Characters	Story about Emma & Jack suddenly introduced "Sarah" in Ch4	**Explicit "ONLY use these characters" rule.** System prompt: "Do NOT introduce new named characters." Validation detects new proper nouns.	chapter-generator.ts:buildChapterSystemPrompt() Validation check #2
Bug #3: Cookie-Cutter Dialogue	Every story had "Let's do this together!" "I believe in you!"	**NO example dialogue in prompts.** Use speech FINGERPRINTS instead (patterns, phrases, tells). Prompt explicitly forbids example snippets.	story-creator.ts:generateCharactersAndRelationships() speechStyle structure in types/index.ts
Bug #4: Generic Storylines	Every mystery followed same "clue → red herring → discovery" pattern	**NO mock storylines in prompts.** Define structure via world rules, character tensions, emotional stakes. Let AI create unique events.	story-creator.ts:generateFoundation() Prompts exclude example plots

Bug #5: Knowledge Amnesia	Max discovered magic door in Ch2, acts surprised in Ch4	**Knowledge state machine.** Track what each character knows and when. Chapter prompts: "Max knows: [list]. Does NOT know: [list]."	types/index.ts:StoryState.characterKnowledge chapter-generator.ts:buildChapterUserPrompt()
Bug #6: Cliffhanger Amnesia	Ch2 ended "Door burst open!" Ch3 started with breakfast scene	**Cliffhanger resolution plans.** Every cliffhanger includes: resolveInChapter, resolutionMethod, resolutionDescription, openingBeat.	types/index.ts:Chapter.cliffhanger.resolutionPlan chapter-generator.ts checks previousSpec.cliffhanger
Bug #7: Forbidden Reactions	Riley shocked at Ch4 revelation, but already knew from Ch2	**Forbidden reactions tracking.** Chapter continuity requirements: "Riley must NOT react shocked to X (reason: already knows). Correct: unsurprised."	types/index.ts:ContinuityRequirements.forbiddenReactions chapter-generator.ts includes in prompt

✅ **Inter-Chapter Validation:** After each chapter generation, gpt-5-mini validates against ALL 7 bugs. If violations found, chapter is regenerated with violations as additional constraints. This catches issues BEFORE they propagate.

Data Flow Diagram

This shows the TypeScript types flowing between each stage:



Key Type Definitions

```
StoryBrief (input from queue): { id, reading_level, genre, primary_virtue,
themes, target_chapters, target_word_count, status, attempts } StoryDNA
(complete story blueprint): { version, storyId, meta, worldBible,
plotStructure, characters, characterTensions, chapterSpecs[],
chapterTransitions[], emotionalStakes, storyState, continuityRules,
editorialChecklist } Chapter (chapter specification + content): {
```

chapterNumber, workingTitle, coreObjective, mainObstacle, emotionalStakesLink, dominantEmotion, sceneType, targetWordCount, cliffhanger?, continuityRequirements?, content?, wordCount? }
CharacterProfile: { name, age, pronouns (LOCKED), archetype, speechStyle: { patterns[], phrases[], emotionalTells } } **StoryState** (prevents knowledge amnesia): { characterKnowledge: [{ characterName, knowledgeItems: [{ fact, learnedInChapter, isSecret }] }], pendingResolutions[], recurringElements[] }

Prompt Architecture

For each AI call, this shows the **strategy** (not full prompts, which are 100+ lines):

DNA Foundation Generation (gpt-5.2)

> ***System:*** *"You are a master story architect. Create rich, engaging story foundations."*
> ***User:*** *Genre, reading level, characters, virtue, themes + JSON structure template*
> ***Constraints:*** *NO example storylines (prevents templating)*
> ***Response Format:*** *json_object*

DNA Characters Generation (gpt-5.2)

> ***System:*** *"Create nuanced, memorable characters. NEVER include example dialogue."*
> ***User:*** *Foundation summary + character names + "Use SPEECH FINGERPRINTS (patterns, phrases, tells)"*
> ***Constraints:*** *pronouns LOCKED, archetype enum, tensionType enum, NO example dialogue*
> ***Response Format:*** *json_object*

DNA Chapters Generation (gpt-5.2)

> ***System:*** *"Story continuity expert. Track character knowledge carefully."*
> ***User:*** *Foundation + characters summary + "Plan cliffhanger resolutions BEFORE writing"*
> ***Constraints:*** *Knowledge progression per chapter, cliffhanger resolution plans, forbidden reactions*
> ***Response Format:*** *json_object*
> ***Warning:*** *May hit 6K output token limit (finish_reason checked)*

Chapter Content Generation (gpt-5.2)

> ***System:*** *Character roster (LOCKED pronouns), "ONLY use these characters", speech fingerprints, world rules*

> ***User:*** *Chapter spec + previous ending + knowledge state ("knows: [X], does NOT know: [Y]") + cliffhanger resolution + forbidden reactions*
> ***Constraints:*** *No new characters, knowledge-aware reactions, resolve cliffhangers, follow world rules*
> ***Temperature:*** *0.8 (more creative)*

Chapter Continuity Validation (gpt-5-mini)

> ***System:*** *"You are a continuity checker. Review for: pronoun consistency, new characters, knowledge violations, cliffhanger resolution, world rule contradictions."*
> ***User:*** *Character roster + world rules + previous context + knowledge states + chapter draft*
> ***Response Format:*** *json_object { violations: [{type, description}], passed: bool }*
> ***Note:*** *NO temperature parameter (gpt-5-mini doesn't support it)*

AI Editor (gpt-5.2)

> ***System:*** *"Expert children's book editor."*
> ***User:*** *Full story text + "Fix: grammar/artifacts/transitions. Flag: values/safety concerns. Don't change: plot/voice/choices."*
> ***Response Format:*** *json_object { editedText, changes[], flags[] }*
> ***Temperature:*** *0.3 (conservative edits)*

Quality Check (gpt-5-mini)

> ***System:*** *"Children's literature quality assessor."*
> ***User:*** *Full story + "Score 5 dimensions: narrativeCoherence (0-25), characterConsistency (0-20), ageAppropriateness (0-20), engagement (0-20), technicalQuality (0-15)"*
> ***Response Format:*** *json_object*

Safety Scan (gpt-5-mini)

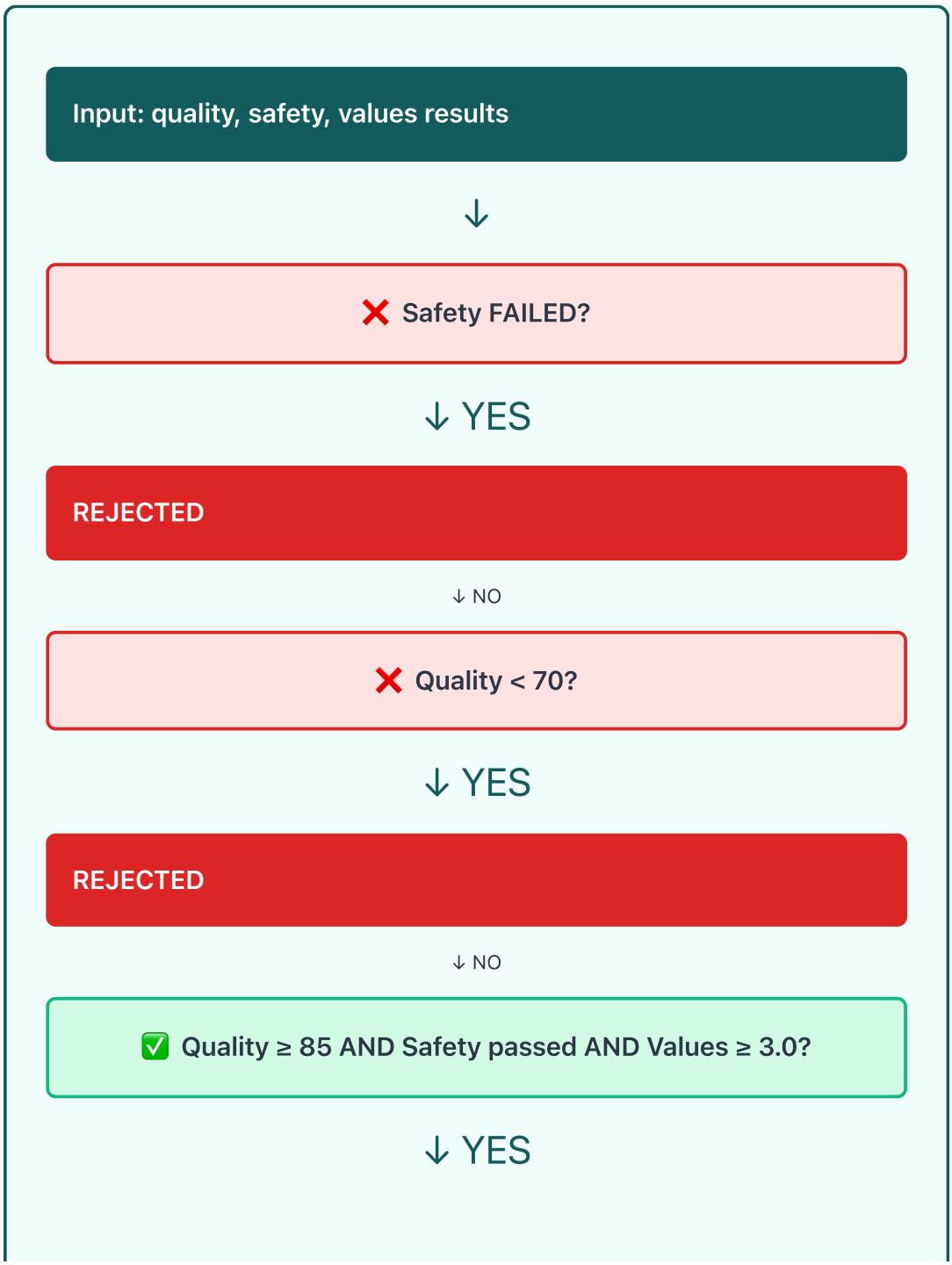
> ***System:*** *"Children's content safety expert."*
> ***User:*** *Full story + 8 safety criteria + "Report ANY violations"*
> ***Response Format:*** *json_object { passed: bool, issues[] }*

Values Check (gpt-5-mini)

> ***System:*** *"Children's values education expert."*
> ***User:*** *Full story + "Score 6 dimensions 1-5: positiveRoleModels, consequenceLogic, conflictResolution, authorityRespect, virtueIntegration, hopefulEnding"*
> ***Response Format:*** *json_object*

Decision Logic: determineStoryStatus()

After all QA checks complete, the pipeline determines story disposition:



AUTO-APPROVED

↓ NO

EDITOR QUEUE

(Quality 70-84 OR Values 3.0-3.9 — needs human review)

Code Implementation (pipeline.ts:171-189)

```
function determineStoryStatus( quality: any, safety: any, values: any ): 'approved' | 'editor_queue' | 'rejected' { // Auto-reject if safety fails if (!safety.passed) { return 'rejected'; } // Auto-reject if quality < 70 if (quality.score < 70) { return 'rejected'; } // Auto-approve if quality >= 85 AND safety passed AND values >= 3.0 if (quality.score >= 85 && safety.passed && values.score >= 3.0) { return 'approved'; } // Otherwise, queue for editor review return 'editor_queue'; }
```

Thresholds Rationale

Threshold	Reasoning
Safety: Binary	ANY safety violation is unacceptable. No tolerance.
Quality: 70 minimum	Below 70 = fundamental flaws (incoherent plot, poor grammar). Not fixable with editing.
Quality: 85 auto-approve	High confidence in narrative quality + technical execution.
Values: 3.0 minimum	Below 3.0 = values concerns (preachy, negative role models). Needs human judgment.
Editor Queue Range	Quality 70-84 OR Values 3.0-3.9 = good but not excellent. Human can polish or catch edge cases.

Discrepancies & Recommendations

❌ CRITICAL: GPT-5-mini Temperature Bug

Issue: quality-check.ts, safety-scan.ts, and chapter-generator.ts:validateChapterContinuity() pass `temperature` parameter to gpt-5-mini, which does NOT support it and will raise an error.

Impact: Pipeline will crash during QA checks.

Files Affected:

- generation/lib/quality-check.ts — Line not visible (uses executeCompletion)
- generation/lib/safety-scan.ts — Line not visible (uses executeCompletion)
- generation/lib/chapter-generator.ts:validateChapterContinuity() — Line not visible

Fix: In utils/openai.ts, modify executeCompletion() to ONLY add temperature for models that support it:

```
const supportsTemp = model.startsWith('gpt-5.2') || model.startsWith('gpt-4'); if (params.temperature !== undefined && supportsTemp) { finalParams.temperature = params.temperature; }
```

Or remove temperature from all gpt-5-mini calls.

⚠️ HIGH: Brief Manager markBriefCompleted Signature Mismatch

Issue: pipeline.ts calls `markBriefCompleted(brief.id, storyId, logger)` but the function signature in brief-manager.ts is `markBriefCompleted(briefId: string, storyId: string, logger: PipelineLogger)`. This works, BUT the function does NOT actually store the storyId in the database — it only updates status and timestamp.

Impact: No foreign key relationship between brief and story. Can't trace which story came from which brief.

Location: generation/lib/brief-manager.ts:60-78

Fix: Add `story_id: storyId` to the update in markBriefCompleted():

```
const result = await updateRecord<StoryBrief>( 'story_briefs', briefId, { status: 'completed', story_id: storyId, // ← ADD THIS updated_at: new Date().toISOString() } )
```

Ensure `story_id` column exists in story_briefs table schema.

⚠️ HIGH: Chapter Title Inconsistency (title vs workingTitle)

Issue: Chapter type has `workingTitle` field, but saveStory() in pipeline.ts saves `ch.title` to database (which doesn't exist on Chapter type).

Impact: Chapter titles will be undefined in database.

Location: pipeline.ts:241-247

Fix: Change pipeline.ts:244 from `title: ch.title` to `title: ch.workingTitle`

⚠️ MEDIUM: Token Usage Tracking Incomplete

Issue: pipeline.ts gets token usage but doesn't break it down per stage. The log.tokenUsage object has rough estimates, not accurate per-stage tracking.

Impact: Can't accurately measure cost per stage for optimization.

Location: pipeline.ts:141-148

Recommendation: Call getTokenUsage() after each stage and calculate delta, or enhance tokenTracker to support named buckets.

⚠️ MEDIUM: Dead Letter Queue Never Checked

Issue: brief-manager.ts marks briefs as 'failed' after 2 attempts, but there's no process to review or requeue them.

Impact: Failed briefs accumulate in database with no recovery path.

Location: brief-manager.ts:105-109

Recommendation: Add admin tool to list failed briefs and manually requeue (reset attempts, set status='queued').

⚠️ LOW: Chapter Gap Detection Throws After Partial Work

Issue: chapter-generator.ts detects missing chapters at the END and throws, but earlier code uses `continue` on error, so some chapters may have been saved to chapters array. Throwing means pipeline fails, but chapters[] has partial data.

Impact: Unclear what happens to partial chapter data. Does it get cleaned up?

Location: chapter-generator.ts:79-88

Recommendation: Consider early validation (check expected count before generation) or more aggressive retry for missing chapters instead of throwing.

⚠️ LOW: DNA Field Inconsistencies in saveStory()

Issue: saveStory() accesses `dna.hook` and `dna.meta.readingLevel`, but these aren't in the StoryDNA type definition shown. `dna.hook` is marked optional, so it may not exist.

Impact: May insert `undefined` or cause runtime errors if fields missing.

Location: pipeline.ts:209, 211

Fix: Validate DNA structure or use safe access:

```
blurb: dna.hook || dna.plotStructure.storyQuestion, reading_level:
dna.meta.readingLevel || brief.reading_level,
```

⚠️ LOW: No Retry Logic for Nano Banana Task Creation

Issue: cover-generator.ts creates task and polls, but if task creation fails, it immediately falls back to genre template. No retry.

Impact: Transient API errors result in fallback instead of retry.

Location: cover-generator.ts:40-53

Recommendation: Add retry loop (3 attempts) around createCoverTask() before falling back.

Summary of Recommendations by Priority

Priority	Issue	Action
CRITICAL	gpt-5-mini temperature bug	Fix openai.ts to conditionally add temperature
HIGH	markBriefCompleted doesn't store storyId	Add story_id to update in brief-manager.ts
HIGH	Chapter title vs workingTitle mismatch	Change ch.title to ch.workingTitle in pipeline.ts
MEDIUM	Token usage tracking incomplete	Enhance per-stage token tracking
MEDIUM	Dead letter queue never checked	Build admin tool for failed brief review
LOW	Chapter gap detection timing	Consider early validation or retry
LOW	DNA field inconsistencies	Use safe access with fallbacks

LOW	No cover task creation retry	Add retry loop before fallback

Cost Analysis: Per-Story Economics

Token Usage Estimates

Stage	Model	Calls	Tokens/Call	Total Tokens	Cost (Input + Output)
DNA Foundation	gpt-5.2	1-3 (retries)	~3,000	~9,000	$0.016 + $0.126 = **$0.142**
DNA Characters	gpt-5.2	1-3 (retries)	~3,000	~9,000	$0.016 + $0.126 = **$0.142**
DNA Chapters	gpt-5.2	1-3 (retries)	~6,000	~18,000	$0.032 + $0.252 = **$0.284**
Chapter Drafting (×5)	gpt-5.2	5	~3,000	~15,000	$0.026 + $0.210 = **$0.236**
Continuity Validation (×5)	gpt-5-mini	5	~1,000	~5,000	$0.001 + $0.010 = **$0.011**
AI Editor	gpt-5.2	1	~8,000	~8,000	$0.014 + $0.112 = **$0.126**
Quality Check	gpt-5-mini	1	~3,000	~3,000	$0.001 + $0.006 = **$0.007**
Safety Scan	gpt-5-mini	1	~3,000	~3,000	$0.001 + $0.006 = **$0.007**

Values Check	gpt-5-mini	1	~3,000	~3,000	$0.001 + $0.006 = **$0.007**
TOTAL AI TOKENS				**~73,000**	**$0.962**
Cover Art	Nano Banana	1	N/A	N/A	**$0.020**
TOTAL COST PER STORY					**~$0.98**

Cost Breakdown by Category

Category	Percentage	Cost
DNA Generation (gpt-5.2)	58%	$0.568
Chapter Drafting (gpt-5.2)	24%	$0.236
AI Editor (gpt-5.2)	13%	$0.126
QA Checks (gpt-5-mini)	3%	$0.032
Cover Art (Nano Banana)	2%	$0.020

Scaling Economics

Volume	Total Cost	Notes
10 stories/day	$9.80/day = $294/month	~300 stories/month library growth
50 stories/day	$49/day = $1,470/month	~1,500 stories/month
100 stories/day	$98/day = $2,940/month	~3,000 stories/month (enterprise scale)

Cost Optimization Opportunities

1. **Reduce DNA retries:** 3 attempts × 3 variations may be overkill. Test if 2 attempts × 2 variations (4 tries) sufficient. Saves ~40% on DNA costs.
2. **Cache world bible templates:** For same genre + reading level, reuse world bible structure. Saves 1 gpt-5.2 call (~$0.15).

3. **Batch QA checks:** Run quality + safety + values in single prompt. May reduce to 1 gpt-5-mini call instead of 3.
4. **Use gpt-5-mini for chapter validation:** Already doing this! Good choice.
5. **Cover art caching:** If generating multiple stories in same genre, reuse cover template variants instead of generating each time.

> ✅ **Verdict:** At ~$1/story, the pipeline is cost-efficient for a premium children's content service. Comparable to commissioning human writers ($50-200/story) but with instant turnaround and guaranteed consistency. The 7-bug prevention system justifies the multi-stage validation overhead.