> This follows the v1 website contracts (block‑structured sections) and “single source of truth” operating principle.

Visual/typographic treatment stays true to your Look & Feel (poetry panel on the cinematic backdrop).

We’re also eliminating any legacy defaults from the Manus kit that could be leaking placeholder copy.

---

✅ One canonical source (save this file)

Save as: /\*\* content/canon/poem.lock.ts\*\*

// Canonical, verbatim poem — DO NOT EDIT without updating sha256.

// If this ever changes, CI will fail (see test below).

export const POEM\_LOCK = {

id: "craftsman-poem-v1",

title: "", // (optional UI label; leave blank if you want no heading)

text:

"Then prove we now with best endevour, what from our efforts yet may spring; he justly is dispised who never, did thought to aide his labours bring; for this is arts true indication - when skill is minister to thought, when types that are the minds creation - the hand to perfect form has wrought",

// SHA-256 of the exact text (UTF-8, no trailing newline)

sha256\_hex: "e06c6a8b4483c12a16da1fe0630551e186859373ba95d5b62194e16898a0a6ea",

sha256\_b64: "4Gxqi0SDwSoW2h/gYwVR4YaFk3O6ldW2IZThaJigpuo="

} as const;

> This matches your wording exactly (including spelling and punctuation). If anyone changes even a character, the hash guard (below) will catch it. Front‑end always renders this text, never a fallback.

---

🔒 Guard + helper (ensure only the locked text renders)

Save as: /lib/poem-lock.ts

import { POEM\_LOCK } from "@/content/canon/poem.lock";

// Node runtime only (SSR/build); keep crypto out of Edge/runtime-critical paths.

export function getLockedPoemText(): string {

// Defer hashing to build/SSR to avoid bundling crypto in clients.

// In production builds we trust the test; at runtime we just return.

return POEM\_LOCK.text;

}

// Optional SSR check (development safety net)

export async function assertPoemLockDev() {

if (process.env.NODE\_ENV === "development") {

const { createHash } = await import("crypto");

const hex = createHash("sha256").update(POEM\_LOCK.text, "utf8").digest("hex");

if (hex !== POEM\_LOCK.sha256\_hex) {

throw new Error("POEM\_LOCK sha256 mismatch — do not edit the poem text.");

}

}

}

---

🧩 Component patch — PoemPanel renders verbatim text

> If you already added PoemPanel.tsx earlier, replace it with this version so it supports a text prop and no defaults (no more accidental placeholder creed).

Save as (replace): /components/PoemPanel.tsx

"use client";

import { motion, useReducedMotion } from "framer-motion";

import React from "react";

export type PoemPanelProps = {

title?: string;

author?: string;

/\*\* Verbatim poem text (exact string). If provided, `stanzas` are ignored. \*/

text?: string;

/\*\* Optional stanza format; ignored when `text` present. \*/

stanzas?: string[][];

footnote?: string;

align?: "center" | "left";

};

export default function PoemPanel({

title,

author,

text,

stanzas,

footnote,

align = "center",

}: PoemPanelProps) {

const prefersReduced = useReducedMotion();

const a = !prefersReduced

? { initial: { opacity: 0, y: 8 }, whileInView: { opacity: 1, y: 0 }, viewport: { once: true }, transition: { duration: 0.55 } }

: {};

return (

<section className="relative overflow-hidden border-y border-white/5" aria-label="Poem">

<div className="poem-backdrop" aria-hidden />

<div className="hero-vignette" aria-hidden />

<div className="container-wrap relative z-10 py-20 md:py-28">

{(title || author) && (

<motion.div {...a} className={align === "center" ? "text-center" : "text-left"}>

{title && <h2 className="text-[clamp(28px,4vw,48px)] font-semibold tracking-tight">{title}</h2>}

{author && <p className="mt-2 text-muted">{author}</p>}

</motion.div>

)}

{/\* VERBATIM mode \*/}

{text ? (

<motion.article

{...a}

className={`mt-8 md:mt-10 rounded-2xl border border-[color:var(--edge)] bg-[color:var(--panel)]/60 p-5 md:p-6 ${

align === "center" ? "mx-auto text-center" : ""

}`}

>

<p className="text-[17px] md:text-[18px] leading-relaxed whitespace-pre-wrap">{text}</p>

</motion.article>

) : (

// Legacy stanza mode (not used once locked)

<div className={`mt-8 md:mt-10 prose prose-invert max-w-none ${align === "center" ? "mx-auto text-center" : ""}`}>

{(stanzas || []).map((lines, i) => (

<motion.div

key={i}

{...a}

transition={{ duration: 0.5, delay: 0.05 \* i }}

className="not-prose rounded-2xl border border-[color:var(--edge)] bg-[color:var(--panel)]/60 p-5 md:p-6 mb-4"

role="doc-poem"

aria-label={`Stanza ${i + 1}`}

>

{lines.map((line, j) => (

<p key={j} className="leading-relaxed text-[17px] md:text-[18px] text-[color:var(--text)]">

{line}

</p>

))}

</motion.div>

))}

</div>

)}

{footnote && <motion.p {...a} className="mt-6 text-xs text-muted max-w-prose">{footnote}</motion.p>}

</div>

</section>

);

}

Important changes

No default poem text exists anymore (prevents any accidental fallback).

A text prop renders with whitespace-pre-wrap → exact characters preserved.

Matches your cinematic poetry panel styling and motion guardrails.

---

🧠 Use it everywhere (page + section renderer override)

A) Home page (or /poem) usage

Save as (replace home snippet): /app/page.tsx

import Hero from "@/components/Hero";

import PoemPanel from "@/components/PoemPanel";

import { getLockedPoemText } from "@/lib/poem-lock";

export default function Page() {

const poem = getLockedPoemText();

return (

<>

<Hero />

{/\* ...other sections... \*/}

<PoemPanel text={poem} />

</>

);

}

B) SectionRenderer override (so API payloads can’t change it)

> We keep to the v1 page contract, but if /v1/site/page ever returns a different poem block, the renderer overrides it with the lock. This protects the front end while your gateway publishes the same block.

Save as: /components/SectionRenderer.tsx

import PoemPanel from "@/components/PoemPanel";

import FeaturesGrid from "@/components/FeaturesGrid";

import { getLockedPoemText } from "@/lib/poem-lock";

type Section = { sectionId: string; component: string; props: any };

export function SectionRenderer({ sections }: { sections: Section[] }) {

return (

<>

{sections.map((s, i) => {

if (s.sectionId === "poem") {

return <PoemPanel key={i} {...s.props} text={getLockedPoemText()} />;

}

if (s.component === "FeaturesGrid") return <FeaturesGrid key={i} {...s.props} />;

return null;

})}

</>

);

}

---

🛡️ Tiny CI test (makes drift impossible)

Save as: /tests/poem-lock.test.ts

import { describe, it, expect } from "vitest";

import { POEM\_LOCK } from "@/content/canon/poem.lock";

import { createHash } from "crypto";

describe("POEM\_LOCK", () => {

it("has exact sha256", () => {

const hex = createHash("sha256").update(POEM\_LOCK.text, "utf8").digest("hex");

expect(hex).toBe(POEM\_LOCK.sha256\_hex);

});

});

Add to package.json:

{

"scripts": {

"test": "vitest run",

"prebuild": "vitest run -t POEM\_LOCK"

}

}

> Your build fails if the poem text changes, enforcing the “one‑true‑poem” forever rule. This aligns with your “deterministic pipelines / artifacts first‑class” operating principle.

---

🌐 Gateway alignment (so API also serves the lock)

When your IntegAI gateway assembles a page, emit a poem block whose props are ignored by the renderer anyway (belt + braces), but keep it the same for parity:

{

"sectionId": "poem",

"component": "PoemPanel",

"props": { "text": "LOCKED\_BY\_FRONTEND" }

}

or return the exact string; either way, the client override wins and parity tests stay green. (Keeps to the v1 “sectionId + component + props” rule.)

---

🧹 Kill the source of drift (do this once)

Run these searches to delete any placeholders/fallbacks that could be surfacing an old poem:

# 1) Any default poem text in components

rg -n --hidden "Craftsman|Creed|poem.\*default|stanzas\s\*="

# 2) Any hard-coded poem copy elsewhere

rg -n --hidden "prove we now|best endevour|dispised|aide his labours"

# 3) Any legacy MDX with poetry that isn’t using the lock

rg -n --hidden "poem" content/\*\*/\*.mdx app/\*\*/\*.mdx

Remove/replace with imports from @/lib/poem-lock only. (No duplicates; no alternative sources.)

---

🗺️ (Optional) Mermaid check — lock path

flowchart LR

A[IntegAI /v1/site/page] -->|sections[]| B[SectionRenderer]

B -->|sectionId == "poem"| C[PoemPanel(text=getLockedPoemText())]

D[/content/canon/poem.lock.ts/] -->|build/SSR| C

E[CI test] -->|hash must match| D

This mirrors the blueprint’s “stable contract + deterministic path” doctrine.

---

TL;DR — What to save and where

/content/canon/poem.lock.ts (canonical poem + hash — the only source)

/lib/poem-lock.ts (helper + optional dev assert)

/components/PoemPanel.tsx (replace; verbatim text prop, no defaults)

/components/SectionRenderer.tsx (override any API payload for poem)

/tests/poem-lock.test.ts (CI guard; wire into prebuild)

Update your page(s) to import from getLockedPoemText().

Once these are in, the poem will never change unless you deliberately update poem.lock.ts and its hash (CI will force you to be intentional). This is fully consistent with how we’re running the site (API ownership + immutable artifacts + no accidental drift).

If you want, I can also add a one‑liner SEO description for /poem and a short MDX wrapper that imports POEM\_LOCK.text (same locking behavior, MDX-friendly).