# BYOP Architecture: Copyright-Safe Handling of Licensed Content in EdTech

**Prepared by:** SHEP Engineering **Date:** April 28, 2026 **Classification:** Technical Reference — may be shared with technical and legal advisors

---

## Context and Problem

Educational technology products frequently need students to practice with content they do not own the copyright to — bar exam questions from NCBE, MBE item sets from commercial courses, professor-authored hypotheticals, or past state bar essays. The platform's value is in grading and feedback, not in hosting the content. But if the platform receives, processes, transmits, or stores that content on its servers, it risks creating unauthorized copies or derivative works under 17 U.S.C. § 106, regardless of whether the student had authorization to use the material personally.

The question is: **how do you build a system that grades a student's response to a copyrighted question without your servers ever possessing the question?**

SHEP's answer is the **Bring Your Own Prompt (BYOP)** architecture — a three-layer privacy stack that keeps licensed content confined to the student's browser while still delivering AI-powered grading at the same quality as platform-authored content.

---

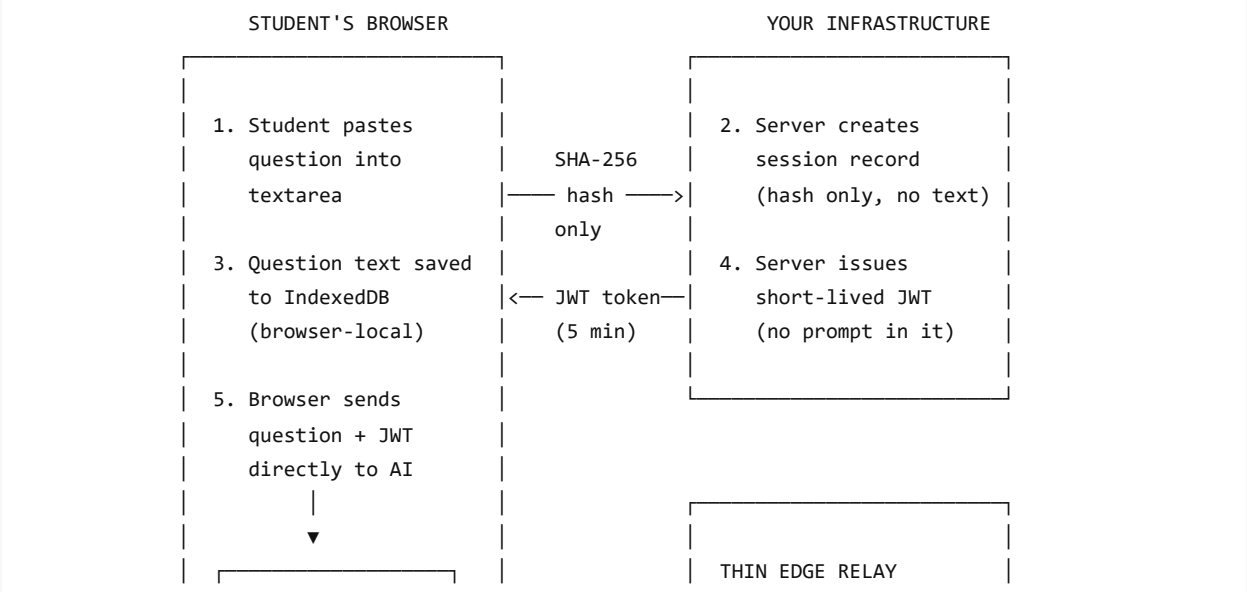
## The Core Claim the Architecture Enables

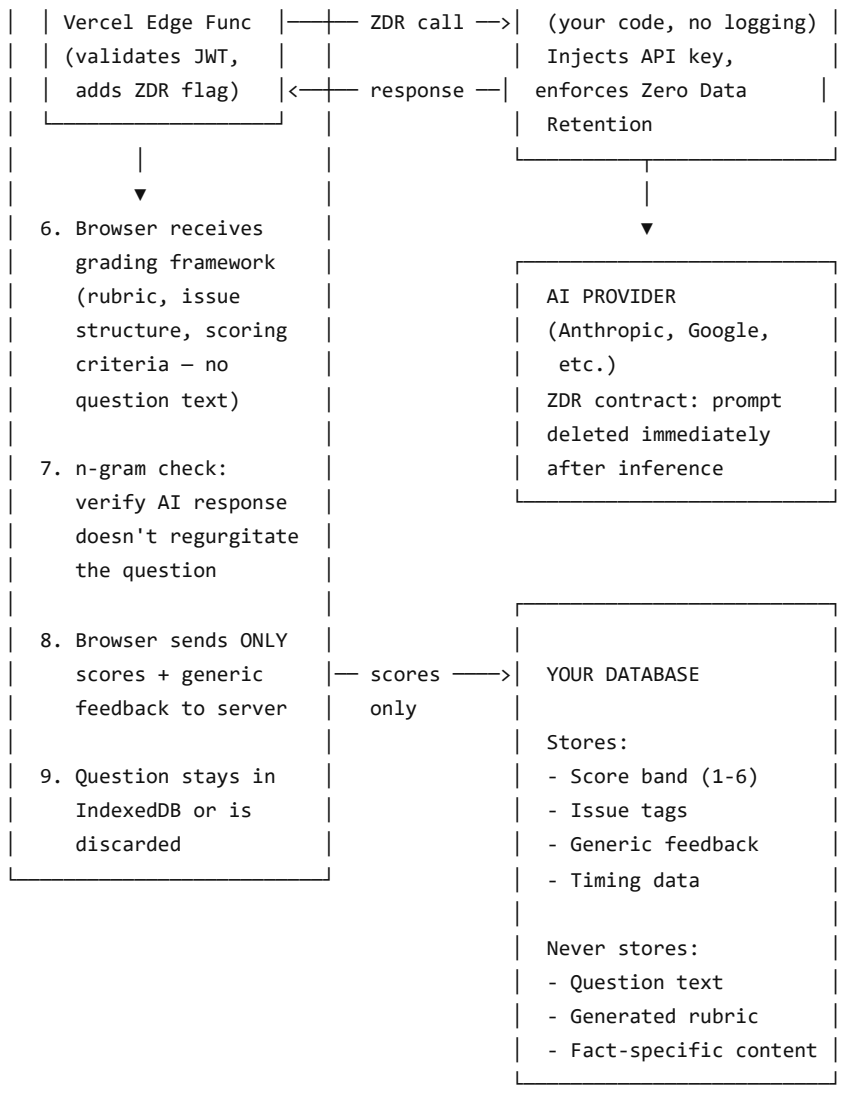
> "*Our servers never receive the question text. The student's browser communicates directly with the AI provider. We receive only generic scores and feedback. Neither we nor the AI provider retain the question.*"

This is a factual claim about system behavior, not a legal conclusion. Every design decision below exists to make this claim verifiable.

---

## Architecture Overview






---

## Technologies End-to-End

| Layer | Technology | Role |
|---|---|---|
| **Client application** | SvelteKit (SSR + SPA), TypeScript | The student-facing application. All prompt handling happens client-side. |
| **Browser-local storage** | IndexedDB (via `idb` library) | Stores the question text on the student's device only. Per-origin, per-browser. No server sync. Cleared if the student clears site data. |
| **Prompt integrity** | SHA-256 hash | Server stores a hash of the prompt for tamper detection and audit trail. The hash is not reversible to the original text. |
| **Authentication relay** | Short-lived JWT (5-minute TTL, single-use, session-bound) | The server issues a signed token so the browser can authenticate to the AI provider without the server ever seeing the prompt payload. The JWT contains only session metadata — never the prompt. |

| | | |
|---|---|---|
| **Edge relay** | Vercel Edge Function | A thin relay (~50 lines of code) that validates the JWT, injects the real AI provider API key, sets the Zero Data Retention flag, and forwards the request. Runs on Vercel's edge network. Has no request-body logging. The prompt passes through this relay in transit but is never written to disk or logged. |
| **AI inference gateway** | Vercel AI Gateway with ZDR | Routes inference requests to downstream providers. When ZDR is enabled, Vercel permanently deletes prompts and responses immediately after the request completes. |
| **AI providers** | Anthropic Claude, Google Gemini | The LLMs that generate the grading framework from the uploaded question. Under ZDR contracts negotiated by Vercel, these providers do not retain prompts after the response is returned, except as required for abuse monitoring under narrow legal exceptions. |
| **Anti-regurgitation** | Client-side n-gram Jaccard similarity | Before submitting any results to the server, the browser checks whether the AI response contains verbatim segments of the question (5-gram overlap > 15%). This prevents the system from accidentally laundering copyrighted text into stored feedback. |
| **Bundle integrity** | HMAC signing (`bundle-proof.ts`) | The generated grading framework is HMAC-signed in the browser to detect tampering between generation and grading. A student cannot modify the rubric to inflate their score. |
| **Grading engine** | Custom V3.2 engine (TypeScript) | The same grading engine used for platform-authored content. It operates on a structured rubric (rule atoms, issue archetypes, scoring criteria) — never on the raw question text. The engine is agnostic to content source. |
| **Result storage** | Neon PostgreSQL | Stores only the performance result: score band, issue tags, IRAC dimension scores, generic feedback phrased without reference to the question's fact pattern. No `prompt_text` column exists in the schema. |
| **Session expiration** | TTL-based cleanup | Upload sessions expire after 90 minutes. Any transient server-side cache of the grading framework is hard-capped at 24 hours. Performance results expire 30 days after the relevant bar exam date. |

---

## The Three-Phase Privacy Stack

### Phase 1: Zero Data Retention (ZDR) at the Provider Level

Every AI inference call that touches user-uploaded content includes `providerOptions: { gateway: { zeroDataRetention: true } }`. Under Vercel's negotiated agreements, the AI provider deletes the prompt and response from all systems immediately after the HTTP response completes. This is a contractual guarantee, not a technical one — but it is the same mechanism enterprises use for HIPAA and SOC 2 compliance with AI vendors.

**What this achieves:** Even if the prompt does transit through a server, the provider does not retain it. This eliminates the "your AI vendor has a copy" attack surface.

**Cost:** $0.10 per 1,000 requests for team-wide ZDR via Vercel AI Gateway.

### Phase 2: Output Minimization

The grading framework generated from the question (issue structures, fact hooks, expected reasoning paths, acceptable conclusions) is a derivative work of the copyrighted content. Phase 2 ensures this derivative work is never durably stored as a platform asset.

- The generated rubric packet lives only in the browser (IndexedDB) or in transient server memory during grading.
- After grading completes, only generic scores and feedback are persisted. The rubric is discarded.
- The database column for the rubric stores only metadata: `{ classifierResult, issueLabels, bundleDigest }` — not the rubric content itself.
- Feedback is phrased in general skill terms ("Your rule statement was incomplete") rather than fact-specific terms ("You failed to discuss that the assigned claim was disputed"). The former is safe to store; the latter starts to look like an answer key.

**What this achieves:** Even if someone subpoenas your database, it contains no copyrighted content and no content from which the original question could be reconstructed.

### Phase 3: Browser-Direct Inference

The prompt never reaches your main servers at all. The browser communicates directly with the AI provider through a thin edge relay. Your backend participates only in two ways: (1) issuing the short-lived JWT that authorizes the browser to use the relay, and (2) receiving the final scores and generic feedback after grading completes.

**What this achieves:** You can make the strongest possible factual claim — "our servers never receive the question" — rather than the weaker claim of "our servers receive it briefly but don't store it." The first is unambiguous in a courtroom. The second invites questions about memory dumps, crash logs, and accidental retention.

**The tradeoff:** Phase 3 is a significant engineering investment. It requires a new authentication flow (JWT issuance), an edge relay, client-side orchestration of the AI pipeline, and handling of failure modes that are normally server-side concerns (network errors, timeouts, partial responses). It also means the upload flow requires JavaScript — no progressive enhancement fallback. For many products, Phase 1 + Phase 2 already provide a strong posture, and Phase 3 is an incremental tightening.

---

## Adapting This Architecture for MBE Questions

The BYOP architecture was designed for essay-style questions (MEE), but the same principles apply to multiple-choice content like MBE items — with one important difference.

**MBE items are more sensitive than essays.** An MEE question is a fact pattern with open-ended calls. An MBE item is a self-contained unit: stem, four answer choices, and a correct answer. MBE items are the core of NCBE's licensing revenue. Storing or transmitting MBE items through your servers creates a higher-risk posture than MEE questions.

The architecture adapts directly:

1. **The student pastes or photographs the MBE question in the browser.** If using OCR (e.g., for photographed items from a physical book), run the OCR model client-side using a WASM-based engine (Tesseract.js) or a local vision model — never send the image to your server for processing.
2. **The browser sends the item to the AI provider via the edge relay (ZDR).** The AI classifies the subject, identifies the tested concept, and returns an explanation of each answer choice — all without your server seeing the question.
3. **The browser stores only the learning record:** subject, tested concept, whether the student got it right, and a generic explanation. The item text, answer choices, and correct answer stay in IndexedDB or are discarded.

4. **The n-gram check is especially important for MBE.** MBE explanations are more likely to quote the stem or answer choices than MEE feedback is to quote a fact pattern. Tune the similarity threshold more aggressively (e.g., 10% rather than 15%) and consider a character-level check in addition to word-level n-grams.

5. **Never store the correct answer on your server.** The student's browser knows they selected "B" and the correct answer was "C." Your server knows only that the student got the question wrong and that it tested "consideration under contracts." The mapping between answer letter and correctness stays client-side.

---

## What Your Legal Team Should Evaluate

This architecture creates a factual posture. Whether that posture is legally sufficient depends on your jurisdiction, your terms of service, your relationship with content licensors, and the specific copyright claims at issue. The architecture does not provide legal immunity — it provides a defensible set of facts.

Key claims to validate with counsel:

1. **"We don't receive the content"** — verifiable through code audit and network traffic analysis. No prompt text in server logs, database, or error reporting.

2. **"We don't retain the content"** — verifiable through ZDR contracts, database schema inspection, and retention policy enforcement.

3. **"We don't create durable derivative works"** — the generated rubric is transient. Scores and generic feedback arguably fall below the threshold of "derivative work" since they cannot be used to reconstruct the original.

4. **"User-directed processing"** — the student initiates the upload, attests to authorized use, and controls local storage. Section 512 of the DMCA provides a safe harbor framework for material stored "at the direction of a user," contingent on notice-and-takedown and repeat-infringer policies.

5. **Provider retention exceptions** — ZDR contracts typically have narrow exceptions for abuse monitoring and legal compliance. These exceptions are time-limited (usually hours, not days) and are industry-standard for enterprise AI use.

---

## Summary

The BYOP architecture separates the platform's grading intelligence (which it owns) from the copyrighted content (which the student supplies and which the platform never possesses). The key insight is that AI inference can happen at the edge of the network — in the student's browser and through a stateless relay — rather than on the platform's servers. This makes "we never had the content" a verifiable engineering fact rather than a policy promise.

The three phases can be adopted incrementally. Phase 1 (ZDR) is a configuration change. Phase 2 (output minimization) is a storage refactor. Phase 3 (browser-direct inference) is the most engineering-intensive but produces the strongest legal posture. For most products, Phase 1 + 2 is sufficient to launch; Phase 3 is the long-term target.