Multi-Outcome Assessment at Scale

Architecture for Competency-Based Education

Erik Treviño

April 2026

Contents

Abstract

Competency-based education (CBE) requires students to be assessed against multiple independent learning outcomes on a single submission. A student's project receives separate scores for Knowledge and Thinking, Written Communication, Collaboration, and other dimensions — each with its own rubric, its own proficiency scale, and its own longitudinal trend. This assessment model is pedagogically established [8][9][15] and practiced by thousands of schools worldwide, yet no commercial Learning Management System supports it natively. Canvas, Schoology, and Google Classroom all store grades as a single score per assignment — an architectural constraint, not a feature gap, that cannot be patched without rebuilding the gradebook schema, every report, every export, and every API endpoint [10]. This paper presents the production architecture of Echo Reborn, a K-12 LMS serving 63 schools and 22,000+ students that implements multi-outcome assessment as a first-class data model. We describe the four-dimensional gradebook schema (`students x activities x constructs x rubric dimensions`), the Row Level Security architecture that enforces FERPA [1] compliance at the PostgreSQL level with 155+ policies across 57 tables, the immutable SHA-256 hash-chained audit log, and the Common Cartridge 1.3 [5] export that preserves pedagogical metadata. The platform was built in approximately 100 days by a single developer, scored 9.85/10 on independent CTO assessment, and operates at $0.003/student/month — 300x cheaper than the incumbent vendor. All SQL and code examples are drawn from the production codebase.

Introduction

The competency-based education movement is accelerating across K-12 and higher education. States including New Hampshire, Vermont, Maine, Colorado, and Oregon require or encourage proficiency-based diplomas [17]. The Aurora Institute (formerly iNACOL) reports that over 600 member organizations are implementing competency-based models [17]. International Baccalaureate schools — over 5,700 globally — use criterion-based assessment that is inherently multi-outcome [5]. Project-based learning networks assess students against multiple dimensions simultaneously using rubrics developed with academic partners including Stanford's Center for Assessment, Learning and Equity (SCALE) [15].

Despite this pedagogical momentum, the technology infrastructure has not kept pace. The K-12 LMS market is dominated by three platforms — Canvas (Instructure/KKR), PowerSchool/Schoology (Bain Capital), and Google Classroom — none of which natively support scoring a single submission against multiple independent learning outcomes. Their data models store grades as `student x assignment -> single_score` [10]. This is not a missing feature. It is a schema-level constraint that would require rebuilding the gradebook, every report, every export, and every integration to change.

This paper presents the architecture of a production system that solves this problem. Echo Reborn is a K-12 LMS built from the schema up for multi-outcome assessment. It serves a national education network of 63 schools and 22,000+ students whose previous platform vendor (Agilix) cancelled their contract, leaving them with no commercial alternative that supported their pedagogical model.

The contributions of this paper are:

1. A four-dimensional gradebook data model that natively supports `students x activities x constructs x rubric_dimensions`
2. A Row Level Security architecture with 155+ PostgreSQL policies enforcing FERPA [1] compliance at the database layer
3. A SHA-256 hash-chained audit log with API-accessible integrity verification
4. A Common Cartridge 1.3 export that preserves multi-outcome rubric mappings
5. Production metrics demonstrating viability: 136K LOC, 1,289 tests, 9.85/10 CTO assessment, built in ~100 days

Problem Statement

The Single-Score Limitation

Every major commercial LMS stores grades using a fundamentally identical data model: one student submits one assignment, one teacher gives one score. The Canvas REST API [10] exposes this directly — the `Submission` object contains a single `score` field and a single `grade` field per student per assignment. While Canvas supports rubrics, the API stores rubric assessments as supplementary data that rolls up into a single score. The gradebook itself is a two-dimensional grid: students (rows) by assignments (columns), with one number per cell.

Schoology and Google Classroom follow the same pattern. The IMS Global LTI 1.3 specification [6] — the standard for tool-to-LMS grade passback — supports only a single `scoreGiven`/`scoreMaximum` pair per line item. Even the interoperability standards assume single-score grading.

This creates an architectural impossibility for multi-outcome assessment. When a teacher needs to score a single student project against Knowledge and Thinking, Written Communication, and Collaboration independently — each with its own rubric dimensions and proficiency levels — the single-score model cannot represent the data. Schools are forced into workarounds: creating phantom assignments for each outcome, maintaining parallel spreadsheets, or abandoning their assessment model to fit the tool.

Why Patching Fails

Adding multi-outcome scoring to an existing LMS is not a feature addition. It requires:

- **Schema change:** Every gradebook table must support multiple scores per submission, with foreign keys to constructs, rubrics, and dimensions
- **Query rewrite:** Every gradebook query, report, and aggregation assumes one score per cell
- **Export rewrite:** CSV exports, Common Cartridge exports, and SIS integrations all assume single-score
- **API rewrite:** Every grade-related API endpoint returns one score; clients consume one score
- **UI rewrite:** The gradebook component renders a 2D grid; multi-outcome requires nested columns

For a platform like Canvas with 30M+ users, this is a multi-year, multi-million-dollar engineering effort that serves less than 5% of the market. No private-equity-owned company will undertake it.

The Compliance Dimension

The problem is compounded by compliance requirements. FERPA (20 U.S.C. §1232g) [1] mandates that student education records are protected and access is restricted to authorized personnel. COPPA [2] requires verifiable parental consent for children under 13. The PowerSchool data breach of December 2024 [14] — exposing 62 million student records — demonstrated the consequences of inadequate security architecture in educational technology.

A multi-outcome assessment platform handles more granular student data than a traditional LMS: not just "student got an 85 on the essay" but "student scored Proficient on Evidence, Developing on Reasoning, and Advanced on Written Expression." This finer-grained data requires correspondingly rigorous access controls.

Related Work

Assessment Frameworks

Robert Marzano's taxonomy [8] defines a proficiency scale (typically 0.0-4.0) across specific learning targets, directly supporting multi-outcome grading where each target is assessed independently. His framework in *Classroom Assessment and Grading That Work* addresses the mechanics of standards-based and topic-based grading — the theoretical foundation for the schema described in this paper.

Norman Webb's Depth of Knowledge framework [9] categorizes cognitive demand into four levels (Recall, Skills/Concepts, Strategic Thinking, Extended Thinking), providing a complementary dimension

of cognitive complexity orthogonal to content knowledge. Multi-outcome rubrics in practice often incorporate both content mastery (Marzano) and cognitive depth (Webb) as independent assessment dimensions.

The CompetencyWorks initiative and Aurora Institute [17] have documented implementation patterns for K-12 competency-based education, including the assessment infrastructure challenges that this paper addresses.

Content and Interoperability Standards

IMS Global Common Cartridge 1.3 [5] specifies content packaging for LMS interoperability — bundling HTML content, QTI assessments, web links, and LTI tool links into a ZIP archive with an XML manifest. However, the specification has no native support for multi-outcome assessment metadata. Assessment items are tied to single-score QTI, and content metadata is limited to basic Dublin Core descriptors. Exporting multi-outcome rubric mappings through Common Cartridge requires extending the specification with custom metadata, as we describe in Section 8.

LTI 1.3 [6] enables tool integration with grade passback via Assignment and Grade Services, but supports only a single score per line item — directly inheriting the single-score limitation of the host LMS.

Database Security in Education

PostgreSQL Row Level Security [4] provides the mechanism for enforcing access controls at the database layer rather than the application layer. Supabase [3] builds on this by integrating RLS with JWT-based authentication, enabling policies that reference the authenticated user's claims directly in SQL. NIST SP 800-171 [12] specifies security requirements for protecting sensitive information in nonfederal systems, with Access Control (3.1.x) and Audit and Accountability (3.3.x) families that map directly to FERPA's requirements.

The Multi-Outcome Data Model

The Four-Dimensional Gradebook

Traditional LMS gradebooks are two-dimensional: students (rows) by assignments (columns), one score per cell. Multi-outcome assessment requires four dimensions:

- **Dimension 1:** Students (rows)
- **Dimension 2:** Activities (column groups)

- **Dimension 3:** Learning constructs per activity (sub-columns)
- **Dimension 4:** Rubric dimensions per construct (drill-down detail)

This data model is implemented through six core tables in the production schema. The following SQL is taken directly from the Echo Reborn migration `00001_initial_schema.sql`:

Core Schema Tables

```sql
-- Unified table for outcomes, standards, and competencies
-- Supports hierarchy via parent_id (standard strands, sub-competencies)
CREATE TABLE learning_construct (
  id              uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  organization_id uuid NOT NULL REFERENCES organization(id),
  type            construct_type NOT NULL,  -- outcome | standard | competency
  name            text NOT NULL,
  slug            text NOT NULL,
  description     text,
  parent_id       uuid REFERENCES learning_construct(id),
  display_order   integer NOT NULL DEFAULT 0,
  is_custom       boolean NOT NULL DEFAULT false,
  metadata        jsonb NOT NULL DEFAULT '{}',
  created_at      timestamptz NOT NULL DEFAULT now(),
  updated_at      timestamptz NOT NULL DEFAULT now(),
  deleted_at      timestamptz,
  UNIQUE (organization_id, slug)
);

ALTER TABLE learning_construct ENABLE ROW LEVEL SECURITY;
```

The `construct_type` enum (`outcome`, `standard`, `competency`) is a deliberate design decision. Rather than creating separate tables for NTN learning outcomes, state standards, and competency-based constructs, a single unified table handles all three. This means the same gradebook, the same rubric infrastructure, and the same reporting pipeline work regardless of whether a school uses NTN outcomes, Common Core standards, or a custom competency framework.

```sql
-- Maps activities to the constructs they assess (THE JOIN TABLE)
CREATE TABLE activity_construct (
  id                    uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  activity_id           uuid NOT NULL REFERENCES activity(id) ON DELETE CASCADE,
  learning_construct_id uuid NOT NULL REFERENCES learning_construct(id),
  rubric_id             uuid REFERENCES rubric(id),
  points_possible       numeric NOT NULL DEFAULT 0,
  weight                numeric NOT NULL DEFAULT 1.0,
  created_at            timestamptz NOT NULL DEFAULT now(),
  updated_at            timestamptz NOT NULL DEFAULT now(),
  UNIQUE (activity_id, learning_construct_id)
);
```

This table is the mechanism that enables multi-outcome assessment. A teacher creates an activity and tags it with relevant constructs: Knowledge and Thinking, Written Communication, Collaboration.

Each `activity_construct` row specifies which rubric applies and what weight this construct carries in the grade calculation.

```
-- THE KEY TABLE: one score per construct per submission
-- This is what makes multi-outcome scoring work
CREATE TABLE construct_score (
  id                    uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  submission_id         uuid NOT NULL REFERENCES submission(id),
  activity_construct_id uuid NOT NULL REFERENCES activity_construct(id),
  score                 numeric,
  scored_by             uuid NOT NULL REFERENCES app_user(id),
  scored_at             timestamptz NOT NULL DEFAULT now(),
  feedback              text,
  is_self_assessment    boolean NOT NULL DEFAULT false,
  is_peer_assessment    boolean NOT NULL DEFAULT false,
  created_at            timestamptz NOT NULL DEFAULT now(),
  updated_at            timestamptz NOT NULL DEFAULT now()
);

-- Per-dimension breakdown within a construct score
CREATE TABLE dimension_score (
  id                   uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  construct_score_id   uuid NOT NULL REFERENCES construct_score(id) ON DELETE CASCADE,
  rubric_dimension_id uuid NOT NULL REFERENCES rubric_dimension(id),
  level                integer NOT NULL CHECK (level BETWEEN 1 AND 10),
  comment              text,
  created_at           timestamptz NOT NULL DEFAULT now(),
  updated_at           timestamptz NOT NULL DEFAULT now(),
  UNIQUE (construct_score_id, rubric_dimension_id)
);
```

The `construct_score` table is the heart of the system. A single submission generates multiple rows here — one per tagged construct. When a teacher grades a project tagged with three learning outcomes, three `construct_score` rows are created, each with independent scores, feedback, and rubric dimension breakdowns via `dimension_score`.

The `is_self_assessment` flag supports a core pedagogical practice: students self-assess against the same rubric criteria, and teachers can view the student's self-assessment alongside their own scoring. This side-by-side comparison is a fundamental element of the partner network's pedagogical model [15].

Gradebook Materialized View

The four-dimensional join across `submission -> construct_score -> activity_construct -> learning_construct` is expensive for gradebook rendering. We solve this with a materialized view that pre-aggregates per-student, per-construct scores:

```
-- From migration 00016_gradebook_materialized_view.sql
CREATE MATERIALIZED VIEW IF NOT EXISTS mv_student_construct_scores AS
```

```sql
SELECT
  s.student_id,
  ac.learning_construct_id,
  a.course_id,
  a.id AS activity_id,
  cs.score,
  cs.scored_at,
  ROW_NUMBER() OVER (
    PARTITION BY s.student_id, s.activity_id
    ORDER BY s.attempt_number DESC, s.submitted_at DESC
  ) AS attempt_rank
FROM submission s
JOIN construct_score cs ON cs.submission_id = s.id
JOIN activity_construct ac ON ac.id = cs.activity_construct_id
JOIN activity a ON a.id = s.activity_id
WHERE
  s.status IN ('submitted', 'returned', 'resubmitted')
  AND cs.is_self_assessment = false
  AND a.deleted_at IS NULL;
```

The `attempt_rank` window function handles resubmissions: `rank = 1` is always the latest attempt. The materialized view is refreshed concurrently via `pg_cron`, ensuring gradebook reads never block on aggregation queries. RLS on the underlying tables still applies at query time.

Why This Schema Is Fundamentally Different

In Canvas, the grade for a submission is a scalar: `score = 85`. In Echo, the grade for a submission is a matrix:

Construct	Rubric Dimension	Level	Label
Knowledge and Thinking	Claim	3	Proficient
Knowledge and Thinking	Evidence	2	Developing
Knowledge and Thinking	Reasoning	3	Proficient
Written Communication	Development	4	Advanced
Written Communication	Organization	3	Proficient
Collaboration	Contribution	3	Proficient
Collaboration	Teamwork	2	Developing

Seven independent data points from one submission. The gradebook aggregates these per-construct across all activities to show longitudinal trends: “This student’s Evidence scores have improved from Developing to Proficient over the semester.” No single-score gradebook can represent this.

Row Level Security Architecture

FERPA at the Database Layer

FERPA [1] requires that education records are accessible only to authorized personnel with legitimate educational interest. Most LMS platforms enforce this at the application layer — middleware checks permissions before returning data. This creates a single point of failure: any bug, misconfiguration, or API bypass that skips the permission check exposes student data.

Echo enforces FERPA compliance at the PostgreSQL level using Row Level Security [3][4]. Every table has RLS enabled. Every query is filtered by the authenticated user's JWT claims. No application code can bypass this — it is a database constraint.

Production RLS Policy: Construct Scores

The following policy is from the production codebase. It governs who can read construct scores — the most sensitive gradebook data:

```
CREATE POLICY "construct_score_select_policy"
  ON construct_score FOR SELECT
  USING (
    EXISTS (
      SELECT 1 FROM submission s
      JOIN activity a ON a.id = s.activity_id
      JOIN course c ON c.id = a.course_id
      JOIN enrollment e ON e.course_id = c.id
      WHERE s.id = construct_score.submission_id
        AND c.organization_id = auth.jwt() ->>> 'organization_id'
        AND (
          e.user_id = auth.uid()
          OR EXISTS (
            SELECT 1 FROM app_user u
            WHERE u.id = auth.uid()
              AND u.organization_id = c.organization_id
              AND u.role IN ('school_admin', 'network_admin')
          )
        )
    )
    AND construct_score.deleted_at IS NULL
  );
```

This policy enforces three constraints simultaneously: (1) the requesting user's organization must match the course's organization (tenant isolation), (2) the user must be enrolled in the course OR be an admin for that organization (role-based access), and (3) soft-deleted records are excluded from results (FERPA data retention with query filtering).

Defense in Depth

RLS is the foundation, but not the only layer. Echo implements four independent security layers:

Layer 1: Database RLS. 155+ policies across 57 tables filter every query by `organization_id`, user role, and enrollment status. The policies are defined in PostgreSQL and execute on every query through the Supabase API, regardless of application code.

Layer 2: Application Scoping. Server Components and Server Actions fetch data through the authenticated Supabase client. All admin queries explicitly include `.eq('organization_id', orgId)` — not relying on RLS alone, but providing defense in depth. A P0 security remediation specifically addressed cases where application queries were overly broad, even though RLS prevented data leakage.

Layer 3: E2E Verification. Dedicated Playwright test suites verify that Teacher A cannot see Teacher B's students and School A cannot see School B's data. These tests run in CI on every pull request, catching RLS regressions before they reach production.

Layer 4: Immutable Audit Log. Every access to PII is logged with user ID, action, target, timestamp, and IP address.

COPPA Compliance

For students under 13, COPPA [2] requires verifiable parental consent before collecting personal information. Echo implements this at the database level:

```sql
-- From migration 00020_coppa_consent.sql
CREATE TABLE coppa_consent (
  id              uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id         uuid NOT NULL REFERENCES app_user(id) ON DELETE CASCADE,
  consented_by    uuid NOT NULL REFERENCES app_user(id),
  consent_type    coppa_consent_type NOT NULL,
  ip_address      text,
  consented_at    timestamptz NOT NULL DEFAULT now(),
  revoked_at      timestamptz,
  organization_id uuid NOT NULL REFERENCES organization(id),
  created_at      timestamptz NOT NULL DEFAULT now(),
  updated_at      timestamptz NOT NULL DEFAULT now()
);

-- Parents can only consent for their linked children
CREATE POLICY coppa_consent_insert_parent
  ON coppa_consent FOR INSERT
  WITH CHECK (
    consented_by = auth.uid()
    AND EXISTS (
      SELECT 1 FROM parent_student_link psl
      WHERE psl.parent_id = auth.uid()
```

```
        AND psl.student_id = coppa_consent.user_id
    )
  );

-- Function: check if student has active COPPA consent
CREATE OR REPLACE FUNCTION has_coppa_consent(student_id uuid)
RETURNS boolean LANGUAGE sql STABLE SECURITY DEFINER AS $$
  SELECT EXISTS (
    SELECT 1 FROM coppa_consent
    WHERE user_id = student_id AND revoked_at IS NULL
  )
  OR NOT EXISTS (
    SELECT 1 FROM app_user
    WHERE id = student_id AND requires_coppa_consent = true
  );
$$;
```

The `has_coppa_consent()` function is called by RLS policies on content-access tables, ensuring that students under 13 without consent cannot access features that collect personal information — including AI feedback features.

The Audit Infrastructure

Hash Chain Verification

FERPA auditors need to verify that audit logs have not been tampered with. Echo implements a SHA-256 hash chain on the audit log: each entry's hash includes the hash of the previous entry, creating a cryptographic chain of custody.

```
-- From migration 00034_ferpa_hardening.sql
ALTER TABLE audit_log ADD COLUMN IF NOT EXISTS previous_hash text;
ALTER TABLE audit_log ADD COLUMN IF NOT EXISTS entry_hash text;
```

The application computes `entry_hash = SHA-256(action + target + timestamp + user_id + previous_hash)` on every write. Any modification to a historical entry breaks the chain — subsequent entries' hashes no longer verify.

An API endpoint (`GET /api/admin/audit-integrity`) verifies the entire chain and returns:

```
{ "verified": true, "totalEntries": 48291, "brokenEntries": 0 }
```

A FERPA auditor can call this endpoint and receive cryptographic proof that the audit trail has not been tampered with. This transforms compliance verification from a manual review process into an automated, API-accessible check.

Data Request Tracking

FERPA Section 99.10 requires institutions to respond to data access requests within 45 days. Echo tracks these requests as first-class database entities:

```
-- From migration 00034_ferpa_hardening.sql
CREATE TABLE data_request (
  id              uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  organization_id uuid NOT NULL REFERENCES organization(id),
  request_type    text NOT NULL CHECK (request_type IN
                  ('access', 'amendment', 'deletion')),
  requester_name  text NOT NULL,
  requester_email text NOT NULL,
  student_id      uuid NOT NULL REFERENCES app_user(id),
  status          text NOT NULL DEFAULT 'pending',
  due_date        timestamptz NOT NULL,
  resolution      text,
  created_at      timestamptz NOT NULL DEFAULT now()
);
```

Each request has a calculated `due_date` (45 days from creation), a status workflow (`pending -> in_progress -> completed`), and an assignment to a responsible administrator. RLS ensures only school and network administrators can view and manage requests for their organization.

Common Cartridge with Pedagogical Metadata

The Export Problem

IMS Global Common Cartridge 1.3 [5] specifies how to package educational content for portability between LMS platforms. The standard handles HTML content, QTI assessments, web links, and LTI tool configurations. What it does not handle is multi-outcome assessment metadata — which constructs map to which activities, which rubrics apply, at what weights.

A standard CC export from Echo would preserve the content but lose the pedagogical structure. A school that exports from Echo and imports into another Echo instance would get the assignments but not the multi-outcome grading framework — defeating the purpose.

The Solution: Extended Metadata

Echo's CC export extends the specification with custom metadata while remaining compliant with the 1.3 schema:

1. **LOM Classification Metadata.** Each activity's `<item>` in the manifest includes `<classification>` elements using Learning Object Metadata (LOM) that specify which learning constructs the activity assesses, with construct IDs and weights.

2. **Outcomes JSON.** A dedicated `outcomes/outcomes.json` file in the archive contains the full rubric framework: constructs, rubrics, dimensions, levels, and their relationships.
3. **Import Parser.** The import path reads `outcomes.json` and recreates `learning_construct`, `rubric`, `rubric_dimension`, `rubric_level`, and `activity_construct` records in the target database.

This means an NTN school that exports a course from Echo and imports it into another Echo instance gets the complete multi-outcome grading framework intact — constructs, rubrics, dimensions, levels, weights, and activity mappings. No other LMS preserves this structure during export.

Production Metrics

System Scale

Metric	Value
Lines of Code	136,301
Files	918
Application Pages	146
API Routes	78
UI Components	142
Database Tables	57
RLS Policies	155+
Database Migrations	131
Database Functions	20+
Database Indexes	124+
Test Cases	1,289 (761 E2E + 528 unit)
Total Commits	550
`as any` Casts	0 (hardened from 47)
Development Time	~100 days
CTO Assessment Score	9.85 / 10.0

CTO Assessment Arc

The platform underwent 8 independent technical assessments over 26 days, scoring progressively higher as each assessment's findings were systematically addressed:

Version	Score	Key Milestone
v1	6.2	Foundation exists, major gaps
v2	8.1	RLS policies, audit log
v3	8.7	Data integrity, test coverage
v4	9.1	COPPA compliance, AI feedback
v5	9.4	Email infrastructure, hardening
v6	9.5	Feature completeness, PWA
v7	9.7	127/127 production routes pass
v8	9.85	All gaps closed, type safety

The v8 assessment noted: "I'm running out of things to critique. 136,000 lines of TypeScript with zero `as any` casts. 1,289 test cases. 155 RLS policies verified by E2E tests. The 0.15 points to a perfect 10 are CSP nonce propagation (blocked on Next.js 16, not an application issue) and 22 fixme'd E2E tests (each with a documented blocker)."

Security Posture

Control	Implementation
Tenant isolation	RLS on every table + app scoping
Audit log	Append-only, SHA-256 hash chain
Audit verification	API endpoint, automated proof
Session management	30-min timeout + fingerprinting
Rate limiting	Upstash Redis sliding window
Input validation	Zod on all inputs
COPPA 2026	Age detection, consent flows
Encryption at rest	AES-256 (Supabase managed)

Control	Implementation
Encryption in transit	TLS 1.3 with HSTS
Type safety	Zero `as any` across codebase
Multi-tenant E2E	Browser-level isolation tests
Migration drift	CI gate blocks mismatched deploys

Cost Economics

Infrastructure Breakdown

Service	Monthly Cost
Supabase Pro (DB, Auth, Storage)	$25
Supabase Compute (2 GB RAM)	$15
Vercel Pro (hosting, CDN, serverless)	$20
Cloudflare (DNS, CDN, email routing)	$0
Resend (transactional email)	$0
Better Stack (uptime monitoring)	$0
Sentry (error tracking)	$0
Upstash Redis (rate limiting)	$0
Anthropic API (AI feedback)	~$5-15
Total	**~$65-75/mo**

For 22,000+ students, this is **$0.003/student/month**.

Competitive Comparison

Platform	Per-Student Cost	Multi-Outcome
Canvas	$4-8/student/yr	No

Platform	Per-Student Cost	Multi-Outcome
Schoology/PowerSchool	$2-5/student/yr	No
D2L Brightspace	$5-10/student/yr	No
Google Classroom	Free	No
Agilix Buzz (incumbent)	~$12/student/yr	Partial
Echo Reborn	**$0.04/student/yr**	**Native**

Echo is 300x cheaper than the incumbent vendor per student. The entire platform costs less per month than a single enterprise LMS implementation fee. This is possible because there are no sales teams, no account managers, no enterprise support tiers, and zero profit margin — the platform is a gift to the partner network.

Gross Margin at Scale

The infrastructure cost structure produces 95-98% gross margins at any scale:

Scale	Students	Monthly Rev	Infra Cost	Margin
Network only	22,000	$0	$100	N/A
First external	30,000	$4,400	$200	95.5%
Network deal	100,000	$25,000	$500	98.0%
Full scale	500,000	$125,000	$3,500	97.2%

This is possible because there are no per-seat licensing costs from any vendor. Supabase, Vercel, and all supporting services use flat-rate or usage-based pricing that scales sub-linearly with users.

Lessons Learned

Migration Drift Prevention

By v8 of the codebase (131 migrations), migration drift had introduced silent schema inconsistencies between the local development database and production. Code referenced columns that existed locally but not in production — and the type system could not catch this because the types were generated from the local schema.

The fix was two-fold: (1) regenerate TypeScript types from the *production* schema, which immediately surfaced 47 `as any` casts that had been hiding type mismatches, and (2) add a CI gate that compares the local migration count against production, blocking deploys when they diverge. This gate caught the exact class of bug that caused a critical incident in v7 — where code referenced LTI-related columns that only existed in local migrations.

Type Safety Hardening

The codebase started with 47 `as any` casts — shortcuts taken during rapid prototyping. Each cast was a lie to the compiler and a potential runtime bug. In a system handling student grades under FERPA, a type error that silently coerces a score is not a minor issue. Hardening all 47 casts to proper types — by regenerating types from the production schema — caught actual bugs and prevented new ones. TypeScript's type checker now covers the full path from database to UI.

RLS Complexity Compounds

The first 20 RLS policies are straightforward. By policy 100, the complexity compounds: cross-organization parent access (parents with children at different schools), security-definer function authorization, and regression risks where one migration accidentally drops guards from another. Four independent security audits caught issues that would have been data exposure incidents. Automated E2E testing of tenant isolation — verifying that User A literally cannot query User B's data through the UI — is the only way to maintain confidence at this scale.

The v7 to v8 Arc

The assessment arc from 6.2 to 9.85 over 22 days illustrates systematic gap closure. Each version's findings were addressed by the next. The gaps identified at v7 — Common Cartridge export, migration drift CI, CSP violation reporting — were all closed by v8. But v8 went further: closing a P0 multi-tenant scoping gap in the network-overview page, regenerating the type system, adding audit chain verification, and implementing multi-tenant E2E tests. The codebase did not just get bigger; it got more correct, more tested, and more secure with each iteration.

Future Work

AI Student Feedback with PII De-identification

Echo includes Claude API [18] integration for rubric-aligned formative feedback on student submissions. The architecture implements PII de-identification as a mandatory preprocessing step: student names, IDs, emails, school identifiers, and classmate names are stripped before any data reaches the AI provider. The AI receives only the submission text (with names replaced by tokens like `[CLASSMATE]`), the rubric criteria, and the learning outcome context. This is complemented by a contractual layer (DPA or BAA with the AI vendor) providing belt-and-suspenders compliance. COPPA consent gates ensure students under 13 cannot access AI features without parental authorization.

SIS Grade Passback

The LTI 1.3 Assignment and Grade Services specification [6] supports only single-score passback. Implementing grade passback for multi-outcome assessment requires either extending the LTI specification or building a custom SIS integration that maps construct scores to the SIS's native grade fields. We are exploring OneRoster 1.2 as a more flexible grade sync mechanism.

Native Mobile and Progressive Web App

The platform is mobile-responsive and deployed as a Progressive Web App (PWA) with offline capabilities. A native mobile application would provide push notification reliability, camera access for assignment submission, and offline-first gradebook viewing for teachers in schools with unreliable connectivity.

Clever and ClassLink SSO

Single sign-on integration with Clever and ClassLink — the two dominant SSO providers in K-12 education — is implemented via Supabase Auth SAML federation. Production deployment requires partnership agreements with each provider.

Conclusion

The gap between competency-based education's assessment requirements and commercial LMS architecture is structural, not incremental. Canvas, Schoology, and Google Classroom store grades as

`student x assignment -> single_score` because their schemas were designed for traditional grading. Retrofitting multi-outcome assessment would require rebuilding the gradebook, every report, every export, and every integration — a multi-year effort that no incumbent will undertake for a minority market segment.

Echo Reborn demonstrates that this problem is solvable with modern tools. The four-dimensional gradebook schema (`students x activities x constructs x rubric_dimensions`) natively supports multi-outcome assessment. Row Level Security with 155+ PostgreSQL policies enforces FERPA compliance at the database layer, not the application layer. The SHA-256 hash-chained audit log provides cryptographic proof of integrity. The Common Cartridge export preserves the pedagogical metadata that defines the assessment framework.

The platform serves 63 schools and 22,000+ students at $0.003/student/month, scored 9.85/10 on independent CTO assessment, and was built in approximately 100 days. The code is production-deployed, the security is independently audited, and the compliance infrastructure is built into the schema — not bolted on after the fact.

Multi-outcome assessment is not a niche requirement. It is the assessment model used by project-based learning networks, competency-based education programs, and International Baccalaureate schools worldwide. The only thing that has been niche is the technology to support it. That gap is now closed.

References

[1] U.S. Department of Education, "Family Educational Rights and Privacy Act (FERPA)," 20 U.S.C. §1232g; 34 CFR Part 99, 1974 (amended). Available: https://studentprivacy.ed.gov/ferpa

[2] Federal Trade Commission, "Children's Online Privacy Protection Act (COPPA) Rule," 15 U.S.C. §§6501-6506; 16 CFR Part 312, 1998 (updated 2024). Available: https://www.ftc.gov/legal-library/browse/rules/childrens-online-privacy-protection-rule-coppa

[3] Supabase Inc., "Row Level Security Documentation," Supabase Docs. Available: https://supabase.com/docs/guides
level-security

[4] The PostgreSQL Global Development Group, "Row Security Policies," PostgreSQL Documentation. Available: https://www.postgresql.org/docs/current/ddl-rowsecurity.html

[5] 1EdTech Consortium (formerly IMS Global), "Common Cartridge v1.3 Specification," 2017. Available: https://www.1edtech.org/activity/common-cartridge

[6] 1EdTech Consortium, "Learning Tools Interoperability (LTI) v1.3 Specification," 2019. Available: https://www.1edtech.org/activity/learning-tools-interoperability

[7] Vercel Inc., "Next.js App Router Documentation," 2024-2026. Available: https://nextjs.org/docs/app

[8] R. J. Marzano, *Classroom Assessment and Grading That Work.* Alexandria, VA: ASCD, 2006.

[9] N. L. Webb, "Criteria for Alignment of Expectations and Assessments in Mathematics and Science Education," Research Monograph No. 6, Council of Chief State School Officers, 1997. Available: https://files.eric.ed.gov/fulltext/ED414305.pdf

[10] Instructure Inc., "Canvas LMS REST API: Submissions," Canvas API Documentation. Available: https://canvas.instructure.com/doc/api/submissions.html

[11] Vercel Inc., "Vercel Deployment Documentation," 2024-2026. Available: https://vercel.com/docs

[12] R. Ross, V. Pillitteri, R. Graubart, D. Bodeau, and R. McQuaid, "Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations," NIST SP 800-171, Rev. 2, February 2020. DOI: 10.6028/NIST.SP.800-171r2

[13] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero Trust Architecture," NIST SP 800-207, August 2020. DOI: 10.6028/NIST.SP.800-207

[14] "PowerSchool Data Breach Exposes Student and Teacher Records from K-12 Districts," BleepingComputer, January 2025. Available: https://www.bleepingcomputer.com/news/security/powerschool-hack-exposes-student-teacher-data-from-k-12-districts/

[15] New Tech Network, "Learning Outcomes Framework," 2010-2026. Rubrics developed with Stanford Center for Assessment, Learning and Equity (SCALE). Available: https://newtechnetwork.org/resources/learning-outcomes/

[16] Wiley, *Competency-Based Education* (peer-reviewed journal), 2016-2026. Available: https://onlinelibrary.wiley.com

[17] C. Sturgis and K. Casey, "Making Mastery Work: A Close-Up View of Competency Education," CompetencyWorks / Aurora Institute, 2018. Available: https://aurora-institute.org/resource/making-mastery-work/

[18] Anthropic, "Claude API Documentation," 2024-2026. Available: https://docs.anthropic.com/

[19] Capgemini, Sogeti, and OpenText, "World Quality Report 2023-24," 15th Edition, 2023. Available: https://www.worldqualityreport.com/

About the Author

Erik Treviño is a Senior SDET and platform builder with 20+ years in software engineering spanning mainframes, fintech, entertainment, cybersecurity, and education technology. He is the creator of behavioral contract testing for AI-powered applications. Erik lives in Austin, TX. More at eriktrevino.ai.