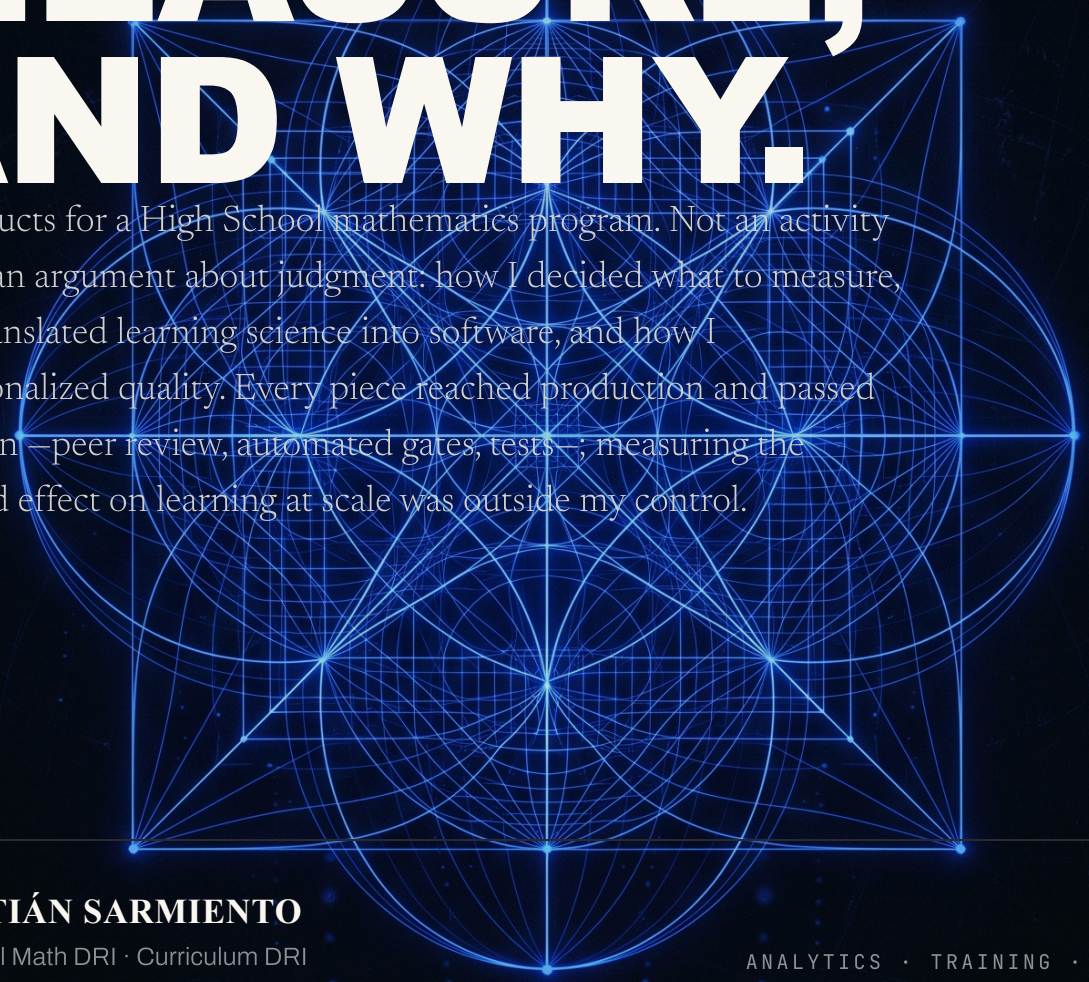
PRODUCT BOOK 2026

Engineering judgment for learning

WHAT TO MEASURE, AND WHY.

Six products for a High School mathematics program. Not an activity log, but an argument about judgment: how I decided what to measure, how I translated learning science into software, and how I institutionalized quality. Every piece reached production and passed validation —peer review, automated gates, tests—; measuring the sustained effect on learning at scale was outside my control.



SEBASTIÁN SARMIENTO

High School Math DRI · Curriculum DRI

DEC 2025 — JUN 2026

ANALYTICS · TRAINING · EVIDENCE

DESIGN, LEARNING SCIENCE AND QUALITY GATES

CONTENTS

THREE SECTIONS · SIX PRODUCTS

1 ANALYTICS & MONITORING

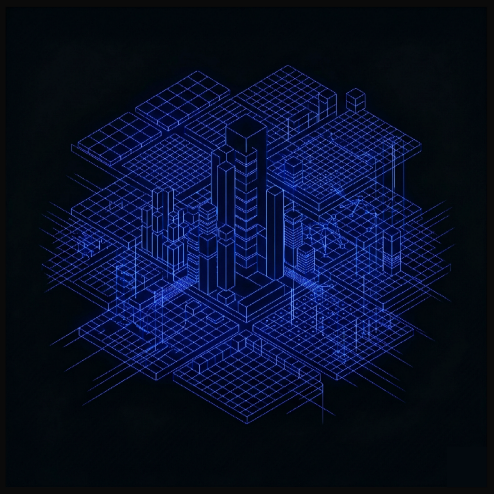
The systems that delivered visibility: seeing the risk of ~1,600 students before it became irreversible.

01 · ANALYTICS & MONITORING

CHAPTER 1.1

Visibility where there was none

MATH ANALYTICS COMMAND CENTER



The academic status of ~1,600 students in a single view — from manual, one-by-one inspection to an intervention decision in under thirty seconds.

STATUS	STACK	DATA	METRICS
Production	Next.js · Firebase	Practice API	5 risk axes

01 THE PROBLEM

A SYSTEM WITHOUT VISIBILITY

Student performance lived scattered across three platforms and could only be checked one student at a time. Knowing how a student was doing meant opening their profile, navigating to *settings* → *documentation* → *progress report* and reading it by hand. Multiplied by ~1,600 students, proactive monitoring was simply unworkable.

No mentor could answer *"how many of my students are at risk today?"* without going through the platform one by one.

There was no way to catch in time a student who was stalled, guessing, or declining in accuracy within the session.

Interventions were reactive — they arrived when a guide reported the problem, not before.

There was no common language: "doing well" or "doing poorly" were qualitative judgments, never measurable or comparable. Every week of undetected stalling was lost progress that wasn't recovered within the session's window.

02 THE OBJECTIVE

AN EVALUABLE STATEMENT

PRIMARY OBJECTIVE

Give every mentor real-time visibility into the academic risk of their students in under 30 seconds, replacing manual one-by-one inspection with a consolidated, actionable view.

As secondary objectives: turn raw API signals into interpretable pedagogical metrics (velocity, knowledge debt, accuracy decay, stability); prioritize automatically — have the system say *who to attend to first*, not just who exists; enable actionable outreach from the same tool; and operate as proactive monitoring without adding manual reporting load to the mentor.

Success was defined up front: answer "who is at risk today?" in under 30 seconds, with 100% coverage of students in a single view and latency resolved by automatic periodic sync.

03 THE ARTIFACT

THE TRIAGE VIEW

The dominant use is a **morning routine**: the mentor opens Triage at the start of the day, reads the red/yellow/green zones, and decides who to contact before the first session. The Risk Score orders the entire cohort by urgency; the color resolves the decision at a glance.

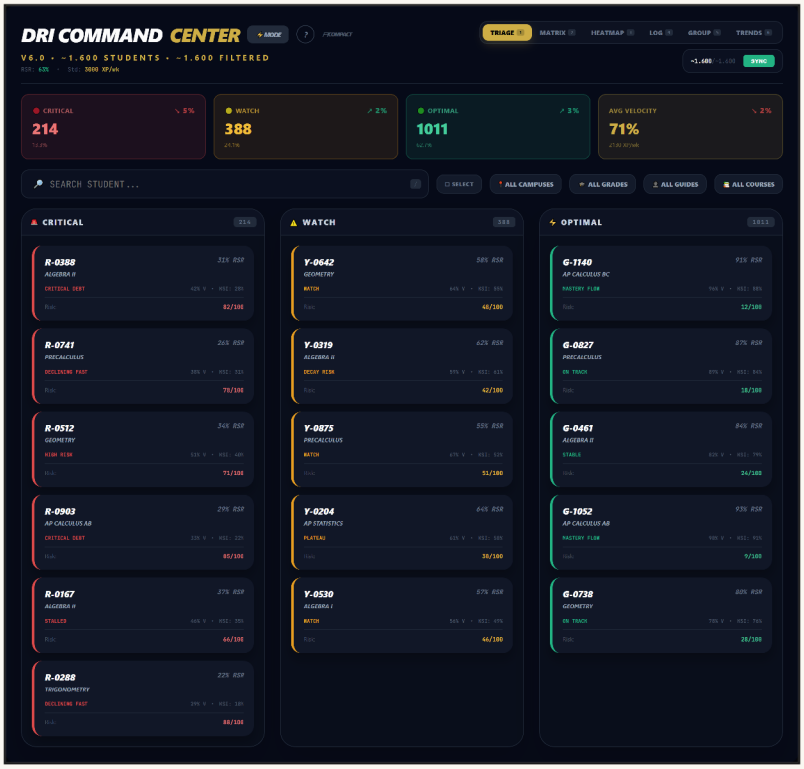


FIG. 1 — TRIAGE VIEW · ~1,600 STUDENTS SPLIT ACROSS THREE ZONES —CRITICAL / ATTENTION / OPTIMAL— AND ORDERED BY RISK SCORE WITHIN EACH ONE · THE COLOR DECIDES WHO TO ATTEND TO FIRST

04 FOUR VIEWS, FOUR QUESTIONS

THE SAME DATA, READ FOUR WAYS

Triage answers "who do I attend to now?". The other views answer different questions about the same cohort: **Matrix** groups by risk pattern —what *types* of student do I have?—; **Heatmap** shows the most critical topics at the cohort level —what *content* is everyone struggling with?—.

FIG. 2 — MATRIX VIEW · THE COHORT POSITIONED IN THE MASTERY (RSR) × STABILITY (KSI) SPACE, WITH THE UPPER-RIGHT QUADRANT AS MASTERY FLOW AND THE LOWER-LEFT AS CRITICAL DEBT

FIG. 3 — HEATMAP VIEW · THE 15 MOST CRITICAL KNOWLEDGE COMPONENTS CROSSED WITH EACH COURSE, PRIORITIZED BY LOW AVERAGE RSR

05 THE FOUNDATION

LEARNING SCIENCE, TURNED INTO METRICS

The dashboard doesn't invent arbitrary indicators: it operationalizes learning-science principles into five measurable metrics, derived from a proprietary calculation protocol documented separately.

PRINCIPLE	METRIC	WHAT IT CAPTURES
Mastery learning	**Velocity**	The student's pace against the standard of 125 XP per week.
Knowledge debt	**DER**	Percentage of K-8 topics an HS student is re-mastering — foundational gaps they carry.
Cognitive fatigue	**PDI**	Accuracy drop between the start and the end of the session.
Retention / stability	**KSI**	Stability of mastered knowledge over time.
Recent success	**RSR**	Sustained accuracy across the last ten tasks.

> *If mentors see risk in real time, they intervene preventively rather than reactively — and the student spends less time stalled within the session's window.*

So that no single metric would oversimplify, risk is computed as a **weighted score across five dimensions**, not as a single number. And so as not to distort motivation, the metrics are the mentor's instruments — never a public ranking of the student.

06 THE DESIGN

ARCHITECTURE AND THE DECISION THAT MATTERED MOST

A proprietary wrapper over the practice platform's API handles fault-tolerant incremental sync and feeds Firestore; the dashboard reads from there live, and the message generator pushes outreach to Slack. Stack: Next.js on TypeScript, Firebase/Firestore, Vercel.

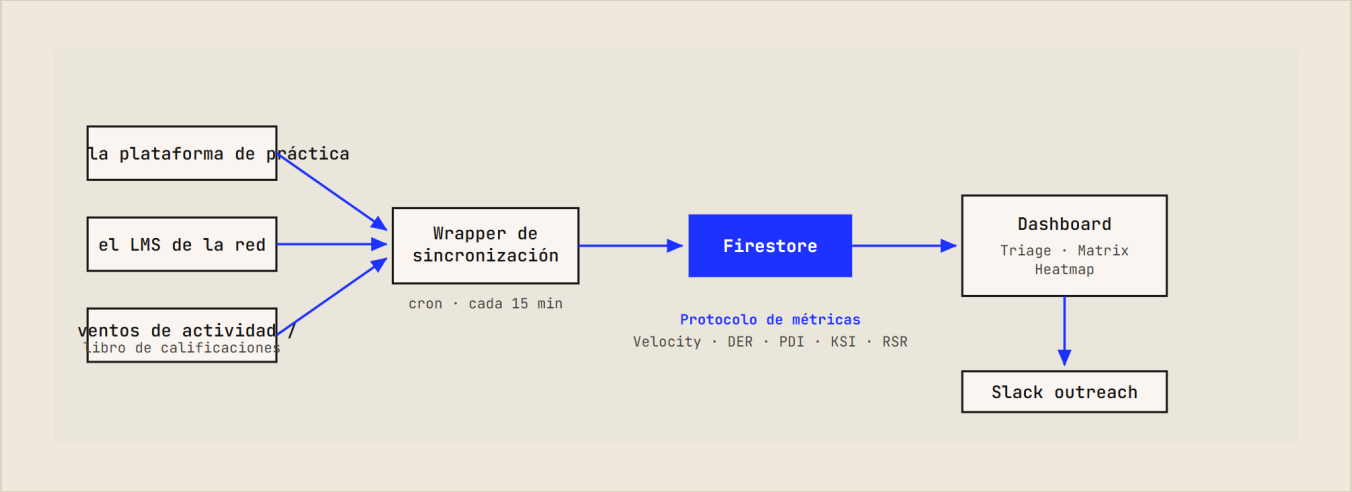


FIG. 4 — DATA FLOW: SOURCES → SYNC WRAPPER → FIRESTORE → DASHBOARD → OUTREACH

Risk as a weighted five-factor score — not a single metric

Context. A single number produced false positives (slow but solid students flagged red) and false negatives (fast but guessing students flagged green).

Decision. Risk Score = Debt Exposure 30% · Velocity 25% · Precision Decay 20% · Stability 15% · Stall 10%. Thresholds: ≥60 **RED** · ≥35 **YELLOW** · <35 **GREEN**.

Accepted trade-off. Harder to explain in one line, but far more faithful to the student's reality — and fidelity is what makes a mentor trust the color.

07 BUILD & VALIDATION

AI-FIRST PROCESS, PUT TO REAL DATA

Development ran with AI as the primary co-developer —architecture, implementation, and refactors—, preceded by a proprietary notebook that explored the API before building the production wrapper. The pattern was constant: **the decision of what to measure and why was always my own**; AI accelerated the *how to build it*. Built in three and a half weeks of sustained work, with thresholds centralized in a single configuration file to recalibrate without touching the logic.

The dashboard was connected to the full population —~1,600 students— and emitted **automated daily digests** of tier changes (one day it reported 79 changes; another, 43), confirming the live pipeline. It was presented to

the data platform team in a *data insights* session, and the calibration of the five factors was tuned against the cohort's observed behavior.

08 THE RESULT

STATUS AND LEGACY

In production, connected to the real population, with automated daily digests. The product didn't stay a demo: it became the visibility layer on which the next product in the portfolio was designed — the Student Performance Tracker Ch. 02, which adds pattern detection and automatic intervention.

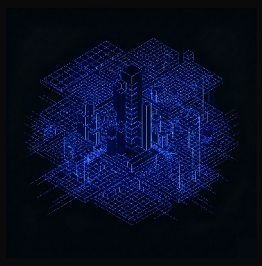
Students in coverage	**~1,600**
Build	AI-first · *what to measure* = my own decision · sustained pace (~3.5 wk)
Integrations	Practice API · Firebase · Slack · Vercel
Views	Triage · Matrix · Heatmap · Log · Student Modal
Status	**In production**

LEARNINGS

- **Technical.** The bottleneck of a live dashboard isn't the interface — it's the robust wrapper over the third-party API: incremental sync, retries, fault tolerance.
- **Product.** A well-weighted composite metric beats five loose ones. The user doesn't want five numbers; they want to know who to attend to first.
- **Pedagogical.** Operationalizing learning science into metrics makes actionable what used to be intuition — and leaves a traceable decision trail.

From seeing risk to acting on it

STUDENT PERFORMANCE TRACKER



The Command Center showed who was at risk. The Tracker decides who to contact today, why, and leaves the draft ready — early intervention turned into an automatic routine.

STATUS	STACK	MODEL	BASE
Production	Next.js · Firebase · Slack	6 triggers	reuses Command Center

01 THE PROBLEM

VISIBILITY WASN'T ACTION

The Command Center Ch. 01 solved the *seeing*: in a single view, the risk of the whole cohort ordered by urgency. But between seeing the red and doing something there was still a manual chasm. Each intervention required the mentor to open the student's profile, read the history, decide on the message and write it by hand — and repeat it, student by student, every morning.

Risk detection was a static snapshot: it said *who* was struggling today, not *what pattern* had led them there nor whether anyone had already intervened before.

No memory of the intervention remained. A mentor could reach out twice about the same thing, or forget to follow up on a case left open the week before.

Prioritizing under time pressure was done by eye: with dozens of students in yellow, which five do you write to before the first session?

And outreach —the step that actually changes the student's trajectory— was precisely the one with the most friction. The bottleneck was no longer information; it was the manual work of turning it into a sent message.

02 THE OBJECTIVE

AUTOMATE DETECTION AND OUTREACH

PRIMARY OBJECTIVE

Move from passive visibility to automatic early intervention: have the system detect risk patterns, prioritize the students who most need contact today, and leave outreach one click away.

As secondary objectives: break risk down into **nameable patterns** —not an opaque score, but six concrete triggers a mentor recognizes—; keep an *intervention history* per student so that no action is duplicated or lost; reduce the morning routine to a single artifact readable in minutes; and draft the first version of the outreach message automatically, leaving the mentor the final judgment.

Success was defined up front: each morning the mentor opens a single view, sees the five priority students with the explicit why, and can trigger contact without writing from scratch again.

03 THE ARTIFACT

THE "MORNING COFFEE" DASHBOARD

The product was designed around a single moment of use: morning coffee. The mentor opens **Morning Coffee**, sees the **five priority students** of the day with the *trigger* that fired each case, a **Quality Score** that measures the student's recent health, and a **Slack draft** already written and ready to send. What used to be dozens of profiles opened by hand is now resolved on a single screen.

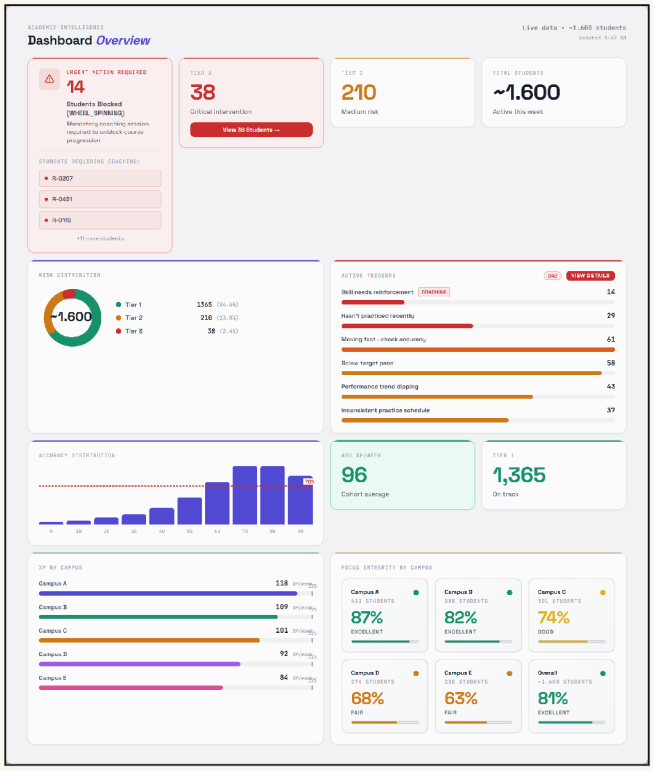


FIG. 1 — "MORNING COFFEE" DASHBOARD · TOP-5 PRIORITY + TRIGGER + QUALITY SCORE + SLACK DRAFT · THE MORNING ROUTINE IN A SINGLE VIEW

04 THE STUDENT CARD

MEMORY OF THE INTERVENTION

If Morning Coffee answers "who do I attend to now?", the student card answers "what has happened to them and what have we done?". Each student —identified only by their `R-/Y-/G-` code, with no personal data— has an **intervention timeline**: which trigger fired, which outreach was sent, what happened afterward. The memory that used to live in the mentor's head is now recorded and queryable.

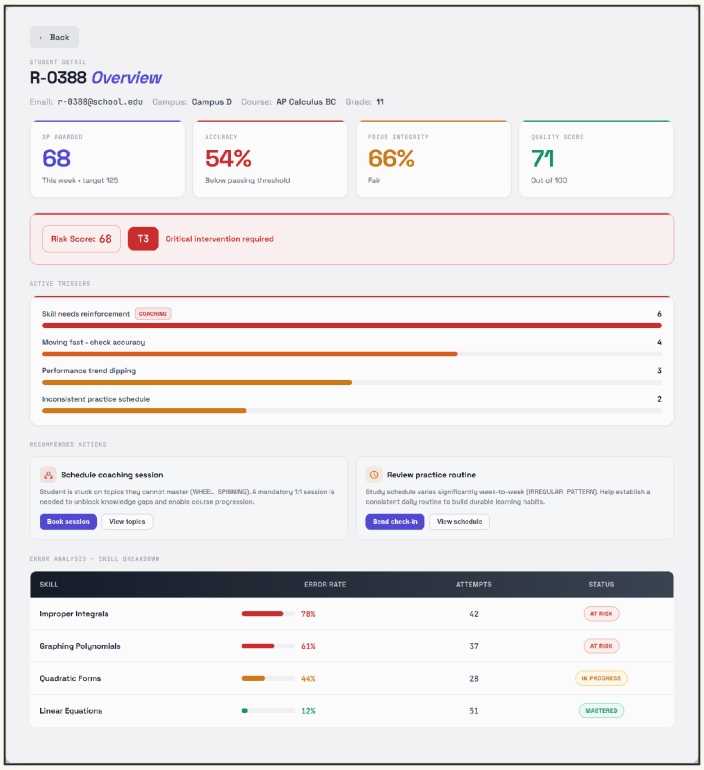


FIG. 2 — STUDENT CARD · INTERVENTION HISTORY IN A TIMELINE · NO CONTACT IS DUPLICATED OR LOST

05 THE MODEL

SIX TRIGGERS, ONE RISK SCORE, ONE OUTREACH TIER

The heart of the Tracker is a pattern-detection model. Six independent *triggers* watch distinct signals in each student's activity; when one or more fire, they feed a **Risk Score** that classifies the student into a **tier**, and the tier decides the form of the **outreach** — from a light touch to an immediate-contact alert.

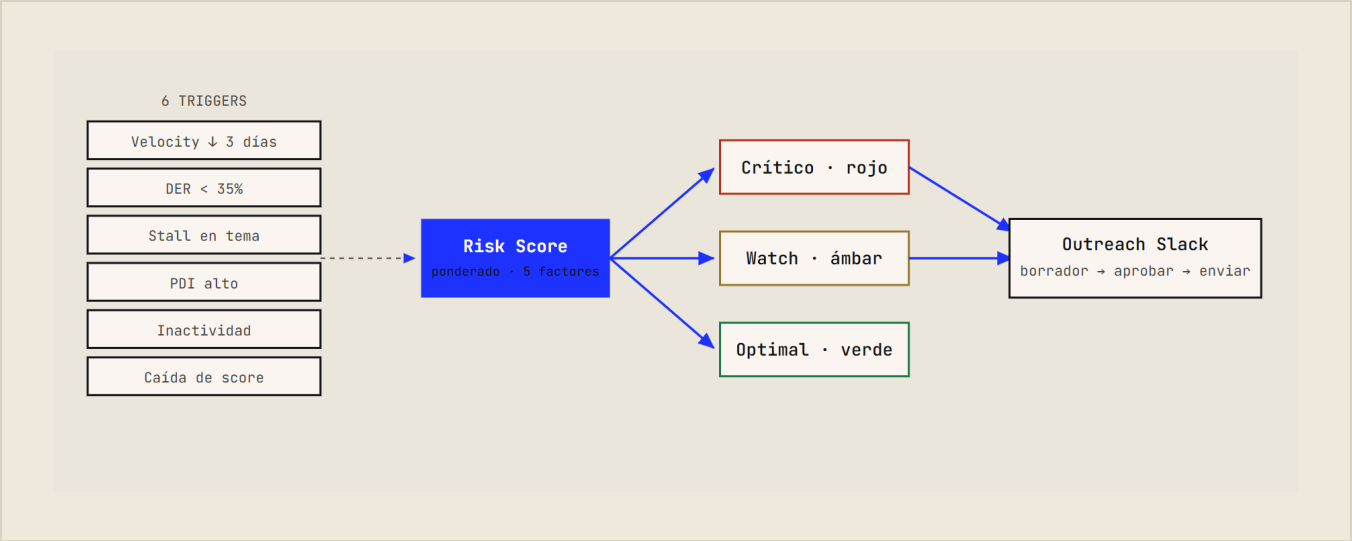


FIG. 3 — MODEL FLOW: SIX TRIGGERS → RISK SCORE → TIER → OUTREACH

TRIGGER	WHAT PATTERN IT DETECTS
Stall	The student stops advancing — no measurable progress for several days in a row.
Velocity drop	The rate of advancement falls below the standard expected for their cohort.
Precision decay	Accuracy collapses within the session — a signal of fatigue or guessing.
Guessing	A pattern of fast, erratic answers that betrays random responses.
Knowledge debt	Accumulation of unmastered foundational topics the student carries.
Disengagement	A sustained drop in activity — less time and fewer tasks than their baseline.

> *It's not an opaque number: each top-5 case arrives with the trigger that fired it, so the mentor understands the why before writing the first word.*

The six triggers are the model's fixed structure; the exact thresholds of each are calibrated against the cohort's observed behavior and live centralized so they can be recalibrated without touching the detection logic.

06 BUILD

AI-FIRST PROCESS ON AN EXISTING BASE

The Tracker didn't start from scratch: it was built **on top of** the Command Center's visibility layer, reusing its sync pipeline and its Firestore. Development ran with AI as the primary co-developer —trigger-model architecture, implementation, refactors—, while the decision of **what signals to measure and how to translate them into outreach** was always my own. Carried through to production on that base, with the Slack integration to push the message drafts directly to the mentor's channel.

The draft generator was the piece that closed the loop: taking the trigger and the student's context and producing an editable first message resolved the friction the Command Center had left open. The mentor keeps the final judgment —edits and sends—, but no longer starts from a blank page.

07 THE RESULT

STATUS AND LEGACY

In production, connected to the real cohort, emitting its priority top-5 each morning with drafts ready. The Tracker completed the arc the Command Center had begun: from *seeing* risk to *acting* on it without repeated manual work.

Detection model	`6 triggers → Risk Score → tier`
Build	AI-first on an existing base · reuse of sync + Firestore
Integrations	Practice API · Firebase · Slack · Vercel
Artifacts	Morning Coffee · Student card · Outreach draft
Status	**In production**

LEARNINGS

- **Product.** Visibility isn't value until it becomes action. The leap from the Command Center to the Tracker was closing the last meter: from visible risk to a sent message.
- **Design.** A nameable pattern —six triggers the mentor recognizes— builds more trust than a perfect but opaque score. The why matters as much as the who.
- **Process.** Building on a previous portfolio product —reusing its sync and its Firestore— made it possible to reach production by reusing infrastructure instead of starting over.

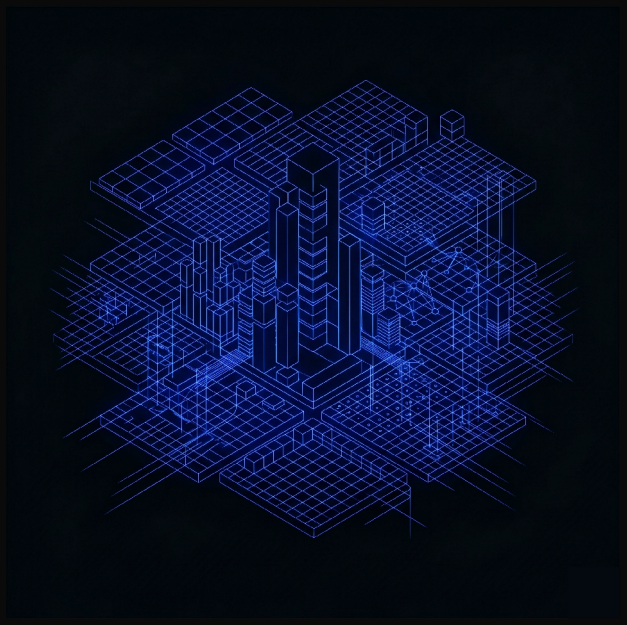
01 · ANALYTICS & MONITORING

CHAPTER 1.3

Only what changes the decision of the day

STUDENT ACTIVITY MONITOR

A bot that watches a student's learning pace every five minutes and pings via Telegram only when it's worth a look.



STATUS	STACK	DATA	DELIVERY
Production	Node.js 18 · 0 deps	Platform API · Telegram	Telegram · low latency

01 THE PROBLEM

THE DATA EXISTED; NO ONE SAW IT IN TIME

Tracking a student on an adaptive practice platform was, by default, a manual student-by-student review: someone had to log into the dashboard, open the profile, read the day's XP, skim the topics worked on, and infer whether the pace was healthy or stalling.

For a Middle School student carrying accumulated conceptual gaps, that tracking could not be weekly nor depend on someone remembering to look.

The concrete problem was one of *latency* and *attention*: when a session went off the rails — repeated low accuracy on the same topic, little progress by mid-afternoon, early abandonment — the data existed in the API, but no one saw it in time to intervene the same day.

Checking the platform by hand every five minutes was unworkable.

And receiving all the raw detail would have been useless noise. The challenge was not accessing the data, but distilling from it only the signals that warrant an action.

02 THE OBJECTIVE

A DEFINITION OF SUCCESS IN SIGNAL-TO-NOISE TERMS

PRIMARY OBJECTIVE

Close the distance between the data and the intervention: detect actionable signals in near real time and push only those signals — not the noise — to a channel where whoever supports the student sees them effortlessly.

I defined success in advance and in signal-to-noise terms: a day of monitoring that produced between **six and ten meaningful messages** — a briefing at the start, progress milestones, critical alerts, session close — instead of the fifteen to twenty trivial pings a naive system would generate, or the total silence of a manual review.

The system had to run on its own, without anyone turning it on, within school hours.

03 THE USERS

THE ALERT ARRIVES WHERE THE PERSON ALREADY IS

The direct user is the person who supports the student's learning: they need to know, without opening any dashboard, whether today's session is on track and where to step in.

The monitored subject is a Middle School student with prior gaps — anonymized as **R-0388** in this chapter — whose recovery plan depended on close, daily support.

The chosen channel was **Telegram** precisely because it lives on the supporter's phone: alerts arrive where the person already is, not on a dashboard they have to remember to visit.

Later the same engine went on to track several students in parallel, each with their own daily XP goal — without touching the core logic.

That choice of channel is not cosmetic: it defines the product's contract. If the notification interrupts, it had better be worth the interruption.

04 THE ARTIFACT

A FULL DAY IN THE TELEGRAM FEED

The product is, for the person using it, a conversation. The bot opens the day with a **morning briefing** that situates the day against the week, marks goal milestones as they are crossed, fires a red alert when a topic genuinely gets stuck, and closes with the session summary. Between six and eight bubbles that tell the story of the day without anyone opening a dashboard.

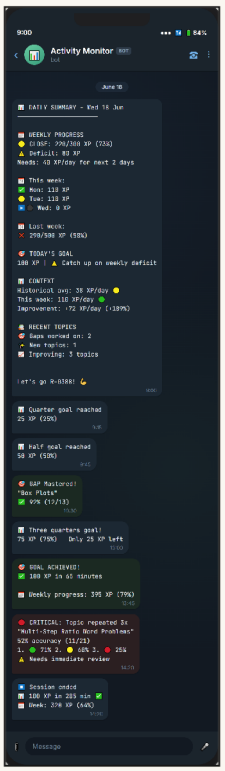


FIG. 1 — A DAY'S TELEGRAM THREAD FOR R-0388 · BRIEFING, XP MILESTONES, CRITICAL ALERT AND CLOSE · RAW DETAIL DISTILLED INTO ACTIONABLE SIGNALS

05 THE FOUNDATION

AN ALERT THAT ALWAYS ARRIVES STOPS BEING AN ALERT

The notification design is anchored in a principle of cognitive load and relevance. The system does not report continuous progress but **significant milestones** — goal quarters at 25, 50, 75 and 100 percent of XP — that correspond to moments when progress changes phase, not to arbitrary increments.

PRINCIPLE	SIGNAL	WHEN IT FIRES
Relevance / cognitive load	`Goal milestone`	On crossing 25, 50, 75 and 100% of the day's XP — phases of progress, not increments.
Deliberate practice	`Critical topic`	Same topic repeated in the day with accuracy below 60% over ≥8 questions — the pattern, not the isolated error.
Deep failure	`Hard alert`	Accuracy below 30% on a topic, or a marked regression against history.
Temporal context	`Briefing`	Start of day situated against the weekly trend and the historical average.
Cycle close	`Session end`	Summary on detecting 30 min of inactivity within school hours.

> *Detecting problem topics rests on deliberate practice: what matters is not an isolated error, but the pattern that recurs with sustained low accuracy.*

The morning briefing situates today against the weekly trend and the historical average, because a metric without context does not allow a decision: **29 XP mean different things** depending on whether the week is running behind or ahead.

06 THE DESIGN

AN AUSTERE ARCHITECTURE AND THE THREE DECISIONS THAT DEFINE IT

The architecture is deliberately austere. The monitor is a single Node.js process **with not a single external dependency** — only native modules — that on each run queries the platform API, compares the freshly read activity against a state persisted on disk, and derives from that difference which notifications are warranted. State lives in JSON files, which keeps the system database-free and trivially inspectable.

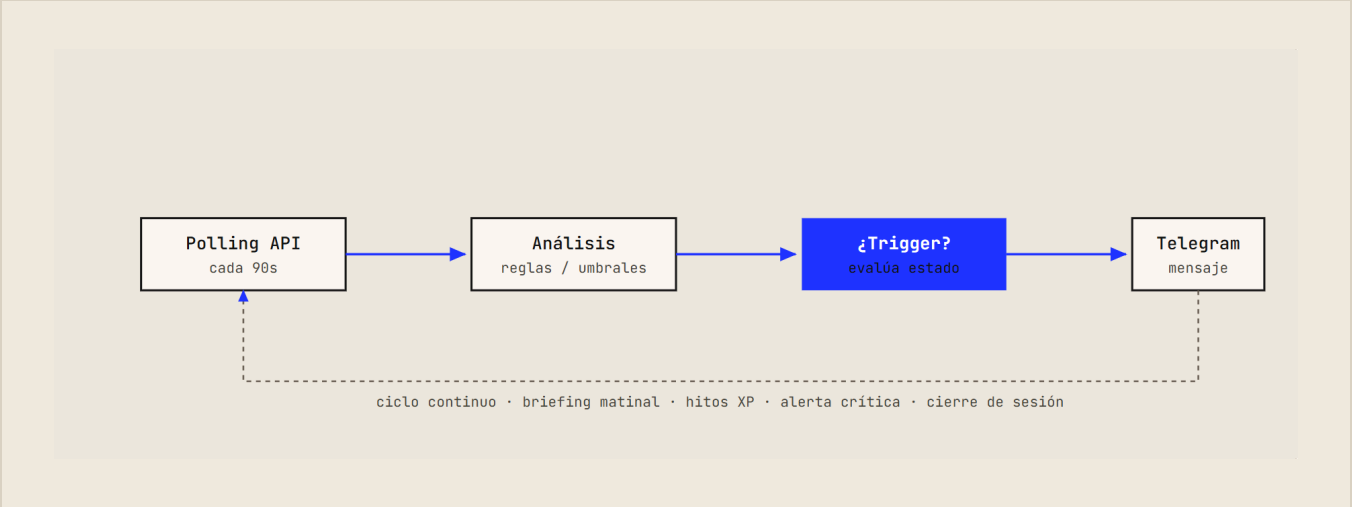


FIG. 2 — THE CYCLE EVERY 5 MIN: CRON-RUNNER → API POLLING → DIFF AGAINST STATE → ANALYSIS → ALERT TO TELEGRAM

Three architecture decisions that define the product

Separate scheduler from monitor. A cron-runner keeps an HTTP server with a health endpoint and fires the monitor every five minutes; that *health check* is what keeps the hosting platform from sleeping the service for inactivity.

Idempotent notifications per day. Flags in the state prevent spam structurally — a milestone once sent is not resent, a critical alert is not repeated — rather than through fragile filters.

Fit within school hours. Monitoring between 9:00 and 16:10 Campus B time, with its own computation of the EST/EDT transition, so as not to generate noise outside class. The network layer uses retries with exponential backoff that distinguishes client errors — which are not retried — from server errors.

07 CONSTRUCTION

TWELVE COMMITS, TWO LESSONS THAT HURT

I built the monitor in an AI-first flow, leaning on Claude to iterate quickly over the notification logic and the API handling. The bulk of the work concentrated in **twelve commits between March 12 and May 21, 2026**. Thresholds and triggers live centralized, so recalibrating relevance does not require touching the logic.

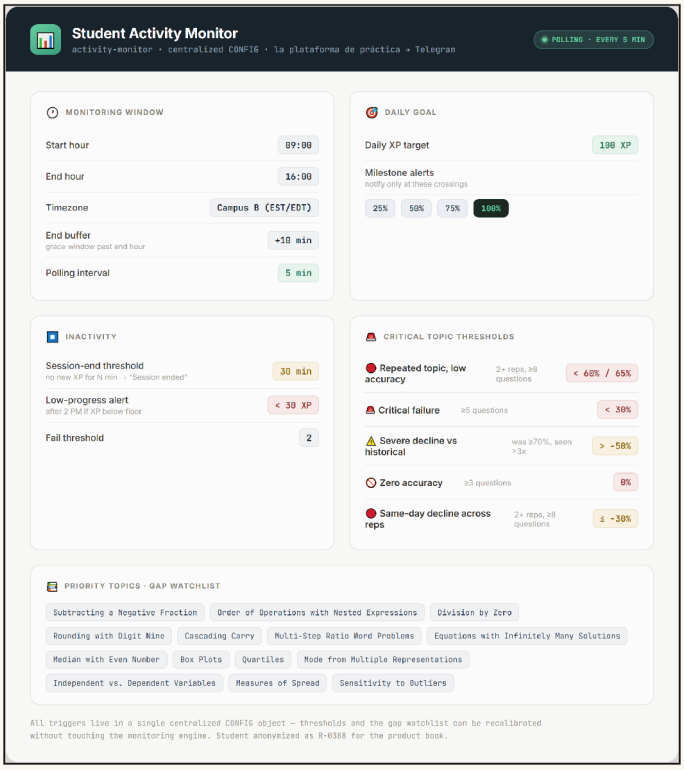


FIG. 3 — TRIGGER CONFIGURATION · MILESTONES, CRITICAL-TOPIC THRESHOLDS AND SCHOOL-HOURS WINDOW CENTRALIZED TO RECALIBRATE WITHOUT TOUCHING THE ENGINE

The first real obstacle was the API itself: requests returned a persistent 400 error until I identified that the *Accept* and *Content-Type* headers were missing and that the base URL had to point directly to the version prefix of the practice platform; I reused the correct configuration already validated in an earlier tracking project. The second lesson came from the persisted state: when migrating the service to the cloud it inherited a local state with all flags already set, so the monitor ran flawlessly but generated **zero notifications** because it believed it had already sent them. That forced me to treat state as a first-class citizen, with a clean reset per instance. The notification system reached a version 4.0 that cut the volume from fifteen-to-twenty pings down to six-to-eight messages with context.

08 VALIDATION

IN PRODUCTION, AGAINST THE EDGE CASES

I validated the system in real production, not in a test environment. The activity log shows the cycle working end to end: query to the API, reading of the day's XP and questions, comparison against state, and effective delivery

of the message via Telegram. I explicitly tested the edge cases that break this kind of bot — running outside hours, critical-pattern detection, days with no activity, API inconsistencies returning totals from different timestamps — with a battery of dedicated scripts before trusting the automatic flow.

The measure of success I cared about was not technical but **signal-to-noise**: that a full day produced the right number of actionable messages. The system proved it could sustain the five-minute cycle within school hours and deliver the morning briefing, the XP milestones, the critical-topic alerts and the inactivity session-close as differentiated and timely messages.

09 THE RESULT AND THE LEGACY

FROM ONE STUDENT TO SEVERAL, WITHOUT TOUCHING THE ENGINE

An individual learning monitor in production that turns the API of an adaptive practice platform into low-latency, high-relevance support. Whoever supports the student stops reviewing dashboards by hand and receives on their phone only what changes the decision of the day.

Monitored student	`R-0388 · Middle School`
Construction	AI-first · no external dependencies · centralized thresholds
Integrations	the practice platform's API · Telegram Bot API
Signal volume	6-8 messages/day (down from 15-20)
Deployment	Railway, migrated to Render
Status	**In production**

The same engine scaled from one student to several in parallel — each with their own daily goal, state file, history and log — without touching the core logic, which confirmed that the dependency-free, per-student state-oriented architecture was the right decision. The service survived a change of hosting platform while keeping the health check that keeps it awake. It rests on the same visibility layer that underpins the analytics portfolio `Ch. 01` `Ch. 02`.

LESSONS

- **Product.** The value of an alert system lies in what it decides NOT to send. Designing relevance — milestones instead of increments, patterns instead of isolated errors, weekly context instead of loose figures — was harder and more important than connecting the API.
- **Technical.** In a stateful monitor, persisted state is part of the design, not an implementation detail: the inherited-flags bug — a system that ran flawlessly while producing zero output — showed that the silent errors live there.
- **Architecture.** The payoff of deliberate simplicity: zero dependencies, JSON files and a health check were enough for a service that runs on its own every school day — and that austerity was what made it trivial to scale from one student to several.

2 TRAINING PLATFORMS

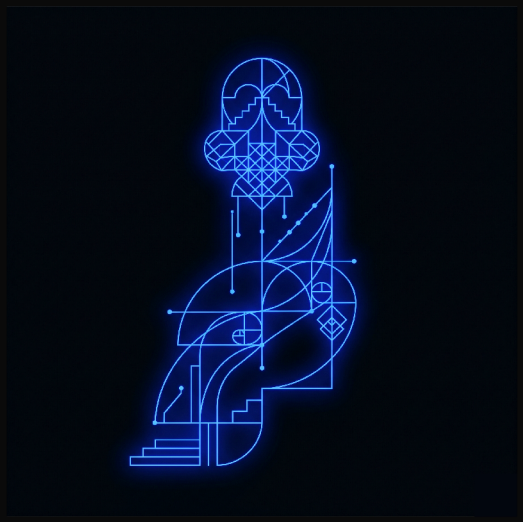
The instruments that produce change: tackling the most expensive difficulty leaps, from the SAT's 650→800 to the AP's 3→5.

02 · TRAINING CHAPTER 2.1

From the minute of algebra to the twenty-second click

DESMOS SAT TRAINING PLATFORM

Turning the Digital SAT's graphing calculator into a trainable skill — the jump from 650 to 800 wasn't about math, it was about fluency with the tool.



STATUS	STACK	DATA / AI	QUALITY
Production · access for 14	React · Vite · Firebase	Firestore · AWS Bedrock	9 build gates

01 THE PROBLEM

A FLUENCY GAP, NOT A MASTERY GAP

The cohort analytics showed a sharp pattern: students hovering around 650 points in the Math section of the Digital SAT were not failing for lack of content. They mastered Heart of Algebra and Passport to Advanced Math — close to 70% of the exam — in their regular courses. What cost them the jump toward 800 was operational: they didn't know how to use the Desmos graphing calculator that the exam itself embeds.

A problem an expert student solves in twenty seconds of clicking took them four minutes of manual algebra, with the risk of error that drags along under time pressure.

The math curriculum they used taught how to solve, not how to use the tool the exam makes available.

It was a structural fluency gap, not one of mathematical mastery.

And no one was training it systematically — the concept was already there; what was missing was automating the procedure with the tool until it freed up working memory.

02 THE OBJECTIVE

INSTALL A REFLEX, NOT TEACH MATH

PRIMARY OBJECTIVE

Close the fluency gap: teach the student to recognize, faced with a SAT-type problem, which Desmos move solves it fastest, and to execute it without syntax hesitation.

The aim was not to teach new math, but to install a reflex: faced with a linear equation, graph it and read the root; faced with a system, find the intersection; faced with a scatter of data, write a one-line regression.

The product goal was defined in advance: that a student who enters the platform with the math already known leaves able to operate the real exam's calculator with **expert speed**.

03 THE USERS

STUDENTS IN THE 650–800 BAND

The primary users are high school students from **a K-12 school network** of AI-accelerated personalized learning, preparing for the Digital SAT and sitting in the 650 to 800 band. The production version gave access to **14 real students**; no learning outcome was measured. To anonymize tracking, each one is identified by a code — for example R-0388 — never by name.

The secondary users are instructors and administrators: through a dashboard they observe each student's progress, completion rates and the error reports that students themselves submit from each question. Role-based access control — student, admin, superadmin — keeps everyone's data isolated.

04 THE FOUNDATION

CONCEPTUAL KNOWLEDGE VS. PROCEDURAL FLUENCY

The design starts from a learning-science distinction: the difference between conceptual knowledge and procedural fluency. The high-band student already has the concept; what they lack is automating the procedure with the tool until working memory is freed.

That is why the entry unit is not mathematical but motor — **Unit 0, keystroke fluency** — to build muscle memory over fractions, exponents, subscripts and the regression tilde before loading any content. On that base, scaffolding fade was applied, the gradual withdrawal of supports as performance rises: each keystroke pattern goes through three levels — scaffold with the literal syntax in view, collapsed reminder, and blank exam like the real Bluebook environment of the SAT — and the level advances on a streak of consecutive correct answers, not on an arbitrary timer.

> *The progression is organized by cohorts according to prior score, so that each student enters at the difficulty rung that fits them and does not repeat what they already master.*

05 THE ARTIFACT

THE SPLIT VIEW: PROBLEM AND DESMOS, CO-VISIBLE

The central screen is a **split view** (split-screen): the problem on the left, the embedded and persistent Desmos calculator on the right. The decision that defines the product was recorded as an ADR in the component's own code: an app whose purpose is to teach how to use Desmos must *show* Desmos at all times. The student reads the prompt, sees the suggested move and executes it in the calculator without switching context.

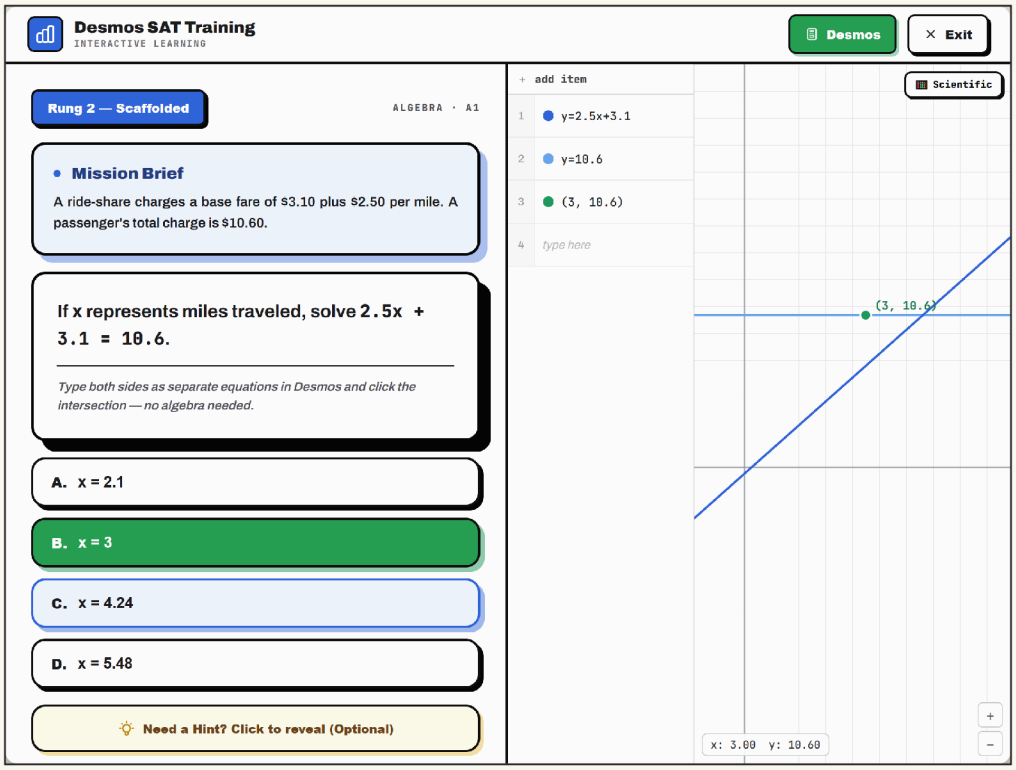


FIG. 1 — SPLIT TRAINING VIEW · SAT-TYPE PROMPT ON THE LEFT, EMBEDDED DESMOS ON THE RIGHT · THE TOOL LIVES CO-VISIBLE WITH THE PROBLEM, NEVER HIDDEN

06 THE CURRICULUM STRUCTURE

FROM UNIT 0 TO THE SAT MIX, RUNG BY RUNG

The content is structured in five layers: **Unit 0** of keystrokes; **Units 1 to 4** aligned with the exam's real domains and weights — Algebra ~35%, Advanced Math ~35%, Data Analysis ~15%, Geometry & Trig ~15%; a final timed **BOSS** that mixes the four domains with medal tiers (Platinum to Bronze, passing 4/6); and a repeatable **SAT Mix** for continuous practice.

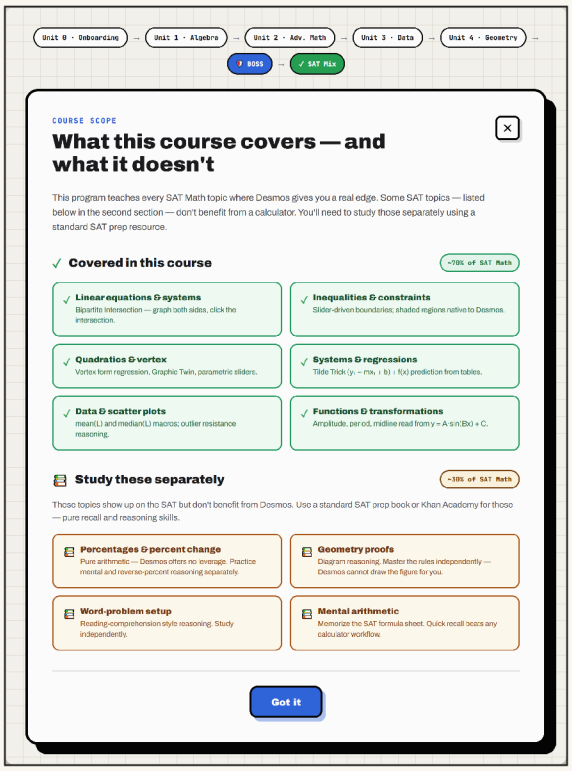


FIG. 2 — CURRICULUM MAP · THE PATH UNIT 0 → UNITS 1-4 → BOSS → SAT MIX, WITH DIFFICULTY SUB-RUNGS PER COHORT

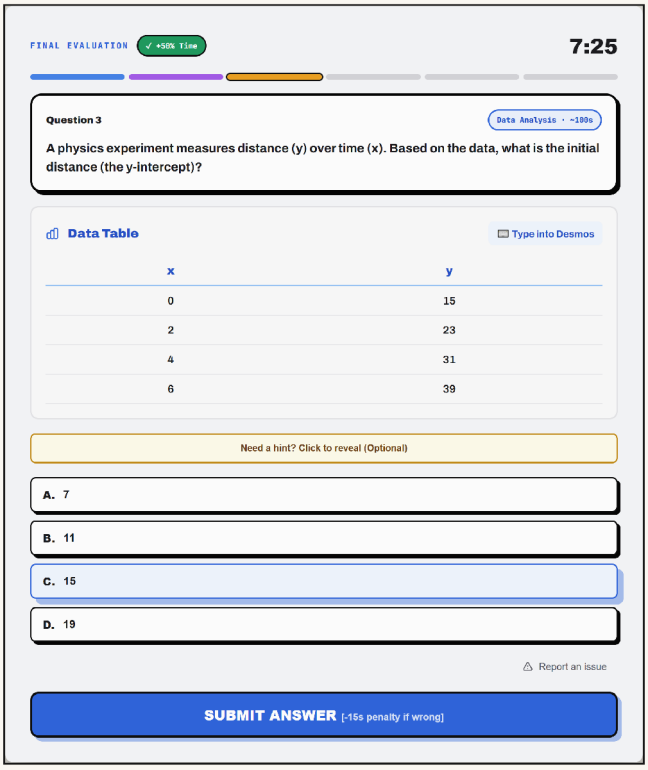


FIG. 3 — SAT MIX MOCK · REPEATABLE PRACTICE THAT REPRODUCES THE REAL BLUEBOOK ENVIRONMENT OF THE SAT — THE "BLANK EXAM" LEVEL OF THE FADE

07 THE DESIGN

CURRICULAR PROGRESSION AND INTEGRATION PIPELINE WITH THE LMS

Global state is handled with persistent Zustand and progress is synced to Firestore with a per-unit progress schema and per-student metrics. The question corpus was generated and analyzed leaning on a pipeline of research scripts that process reference SAT PDFs with models via AWS Bedrock, and is served from per-unit item banks typed in TypeScript. The platform integrates with **the network's LMS** through activity events and gradebook writes.

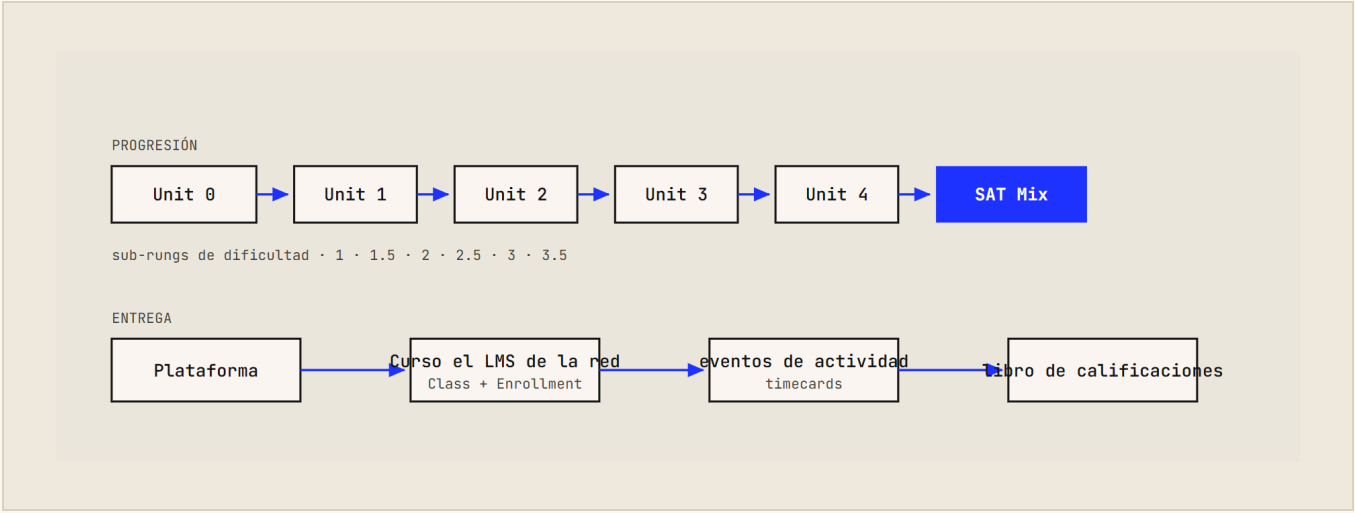


FIG. 4 — CURRICULAR PROGRESSION (UNIT 0 → SAT MIX) AND INTEGRATION PIPELINE WITH THE LMS VIA ACTIVITY EVENTS + GRADEBOOK

Desmos lives embedded and persistent — not hidden behind a floating button

Context. The initial version hid the calculator behind a floating button that covered the content. An app that teaches how to use a tool cannot hide it.

Decision. After an instructional audit it was redesigned so that Desmos lives embedded and persistent in the right panel, co-visible with the problem, inheriting the pattern of Unit 0.

Accepted trade-off. Less horizontal space for the prompt, but the tool stays always in view — and showing the tool *is* the product's pedagogy.

08 CONSTRUCTION AND VALIDATION

QUALITY INSTITUTIONALIZED AS BUILD GATES

Development ran with an AI-first flow sustained over about three and a half weeks. Content quality does not rest on manual review: it was institutionalized as a set of automatic gates that run before each build — a fidelity check that recomputes the answer from the item's data, strict skill coverage, option ordering, LaTeX render, valid Desmos syntax, keystroke patterns, and correspondence between question type and move. If any one fails, the build fails.

Validation combined three fronts: **mathematical peer review** with two high school experts, who verified the correctness of the items and the pedagogy of the moves; **instructional QC** that audited the learning experience end to end; and the battery of **automatic gates** as a permanent safety net. Deployment is to Vercel with auto-deploy from the main branch and an invitation system with superadmin approval. The students themselves feed an error-report channel from each question.

09 THE FOUNDATION, TURNED INTO A TABLE

THE PRODUCT'S EVIDENCE

Students with access	14 · no outcome measurement
Construction window (v2)	May 25 → Jun 18 2026
Commits (v2)	346
Content units	Unit 0 (keystrokes) + Units 1-4 + BOSS + SAT Mix
Cohorts by prior score	foundational <600 · intermediate 600-740 · advanced 750-800
Quality gates at build	9 (fidelity, coverage, ordering, render, desmos, keystrokes, repetition, move, move coverage)
Scaffolding fade levels	3 (scaffold · reminder · exam)
Weight of SAT domains covered	Algebra ~35% · Adv Math ~35% · Data ~15% · Geo&Trig ~15%
Institutional integration	the network's LMS — activity events + gradebook writes
Validation	mathematical peer review (2 experts) + instructional QC
Status	In production

10 THE RESULT

FROM AN ANALYTICS INTUITION TO AN OPERATIONAL PRODUCT

The platform went to production with open access for 14 students, covering the full path from Unit 0 to BOSS plus the SAT Mix, with LMS integration active. Validation was of judgment, not of outcome: measuring the sustained effect on the

real score would have required a rollout and a window I did not control. What was measured was product quality — and the peer review was unequivocal: one reviewer described having learned a lot about Desmos with the system itself, and another rated it a very solid course.

Beyond the deployment, the underlying result was turning an analytics intuition Ch. 1 — the 650 to 800 gap is one of Desmos, not of math — into an operational product with a curriculum of verified moves, a scaffolding-fade engine and a set of quality gates that ensure no incorrect item reaches the student.

11 LESSONS

WHAT THE PROJECT LEFT BEHIND

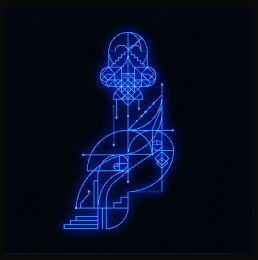
- **Design.** An app that teaches a tool has to display that tool, not hide it; the audit that moved Desmos from a floating overlay that covered content to a persistent embedded panel changed the experience at its root.
- **Trust in the content.** Machine-verified does not equal fully verified — after regenerating content it pays to look at it rendered, because the eye catches legibility problems and clones the gate does not see.
- **Architectural.** Encapsulating the data layer well pays off: having Firestore isolated behind a service layer made it possible to later evaluate and execute a backend migration with a clean cutover.
- **Process.** Institutionalizing quality as gates that fail the build, instead of human checklists, is what sustains a pace of hundreds of commits without the content degrading.

02 · TRAINING

CHAPTER 2.2

Teaching how to prove, not just how to compute

AP MATH JUSTIFICATION TRAINER



From the empirical argument —"I tried it with three numbers and it works"— to the formal justification that an AP exam rewards with a 5. The CERC framework makes mathematical argumentation trainable and measurable.

STATUS	STACK	DATA	MODEL
Beta	Next.js 14 · TS · KaTeX	Mock KV ↔ LMS	CERC scoring

01 THE PROBLEM

KNOWING HOW TO DO THE MATH, NOT HOW TO DEFEND IT

Advanced math students had mastered the procedure but were losing points where they count most: in the written justification. The diagnosis was clear and quantified. On the adaptive practice platforms the network used, multiple-choice performance hovered around **82%**; on the Free Response Questions –where the examiner grades not the result but the reasoning that supports it– it dropped to **67%**, and only a minority were truly on track toward the target score.

The existing procedural tool was solid for automating computation, but it did not train what a real AP exam measures: written mathematical argumentation.

A student could solve an integral flawlessly and still fail the question for not stating the condition that enables the theorem they used.

That gap —knowing how to do the math but not how to defend it— was invisible to the existing system.

And it was, exactly, the difference between a 3 and a 5.

02 THE GOAL

JUSTIFICATION AS A TRAINABLE SKILL

PRIMARY GOAL

Move the student from the empirical argument to the formal justification that cites the theorem's hypothesis and verifies its conditions before invoking it — treating mathematical justification as a trainable, measurable skill, not a diffuse talent.

To make this operational, an explicit scaffold was adopted: the **CERC** framework —Claim, Evidence, Reasoning, Conditions— which breaks every proof down into four visible, required moves.

The goal was not for the student to write more, but to write *complete*: that none of the four elements stay implicit, because it is precisely the implicit that an examiner cannot reward.

03 THE USERS

THE STUDENT AND THE TUTOR

The primary user is the advanced math student who already masters the calculation but loses points on the written answer. The platform was designed for three parallel courses —Calculus AB, Calculus BC and Statistics— because the structure of the argument is the same even when the content changes: a claim, its evidence, the principle that connects them, and the conditions that enable it.

The second user is the tutor or academic coordinator, who operates through an admin panel with visualization of each student's reasoning state (R-0388 and its peers always appear anonymized), per-unit progress tracking, and manual practice triggers.

> *A deliberate privacy decision: the tutor panel isolates sensitive data and never exposes to the student the pedagogical "traps" of each problem, so as not to contaminate the exercise.*

04 THE ARTIFACT

THE SPLIT-SCREEN SESSION

The central interface is a **split-screen** session: on the left, the problem statement with the notation rendered in KaTeX and the theorem box — name, statement, hypotheses—; on the right, the CERC form, four stacked fields —Claim, Evidence, Reasoning, Conditions— each with its description and its sentence frame where applicable. The completeness counter and the progress bar give real-time visual feedback as the student fills in each field.

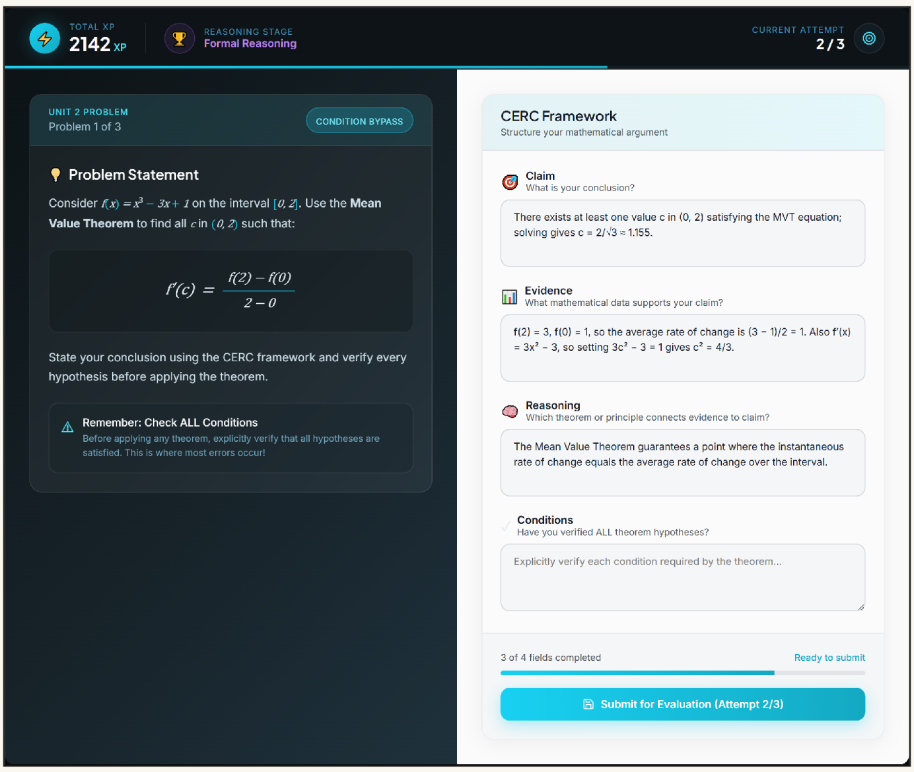


FIG. 1 — CERC SESSION · CALCULUS FRQ ON THE LEFT, CLAIM/EVIDENCE/REASONING/CONDITIONS FORM ON THE RIGHT · "3 OF 4 COMPLETE" SHOWS THE LIVE COMPLETENESS SCORING

05 THE FEEDBACK

COMPLETENESS SCORING, NOT SEMANTIC CORRECTNESS

After submission, the system evaluates **deterministically** whether the four elements are present and non-empty, and projects it onto the AP 1-to-5 rubric. It does not judge whether the argument is "good" with a language model: the first habit to install is not eloquence, but the structural integrity of the argument —that the condition is written, that the evidence exists, that the reasoning names the theorem.

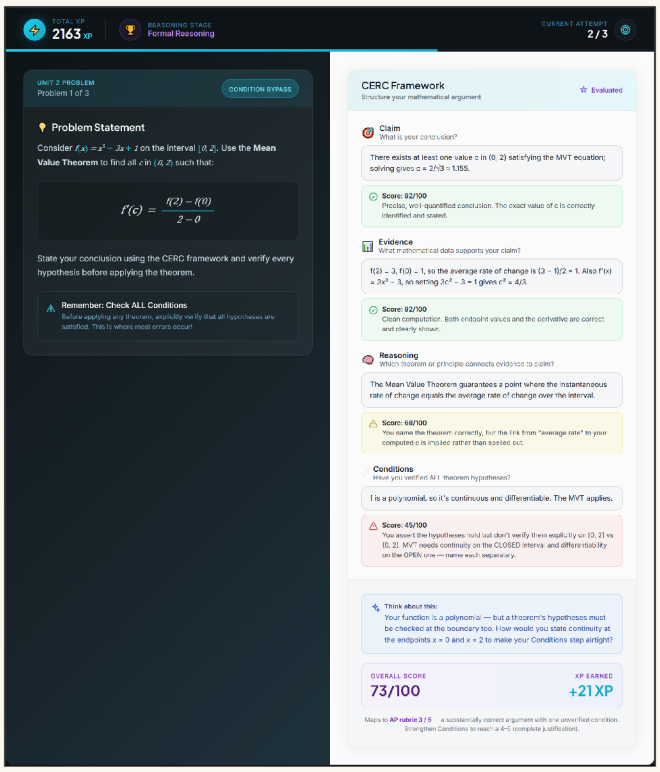


FIG. 2 — POST-SUBMISSION FEEDBACK · CERC COMPLETENESS MAPPED TO THE AP 1–5 RUBRIC · A SEMANTIC EVALUATOR WOULD HAVE BEEN MORE IMPRESSIVE AND FAR LESS HONEST ABOUT WHAT IT MEASURED

06 THE FOUNDATION

THREE STAGES, FOUR COGNITIVE UNITS

The design rests on a model of mathematical reasoning development with three successive stages, coded literally into the application's data type: **empirical** (the student is convinced by examples), **generic** (generalizes the pattern but without rigor), and **formal** (proves by invoking the structure). The course is built to force that transition.

UNIT	COGNITIVE FOCUS	WHAT IT TRAINS
U1	Breaking the empirical illusion	Problems designed so that intuition fails: the student experiences why seeing three cases is not enough.
U2	Condition verification	Verifying ALL of a theorem's conditions, with no shortcuts — the most expensive mistake on the real exam.
U3	Synthesis without scaffolding	Multi-concept synthesis and communicative precision, now without sentence frames.
U4	Timed FRQs	Individual Free Response under exam conditions.

The catalog classifies each problem by the type of error it provokes — `CONDITION_BYPASS`, `LOCAL_ONLY_ARGUMENT` or `CER_BREAKDOWN`— and pairs them with sentence frames (*sentence frames*) that are withdrawn unit by unit: a direct application of the *fading* principle, where the scaffolding fades away as competence consolidates.

07 THE DESIGN

THE CERC MODEL AND THE DECISION THAT MATTERED MOST

CERC breaks every proof down into four moves: the assertion (Claim), the evidence that backs it (Evidence), the principle or theorem that connects them (Reasoning), and the conditions that enable it (Conditions). A three-level hint system —where the flaw is, which CERC element is broken, the explicit correction— supports the student without solving the problem for them. Gamification adds XP per unit and unlocks badges with GSAP animations.

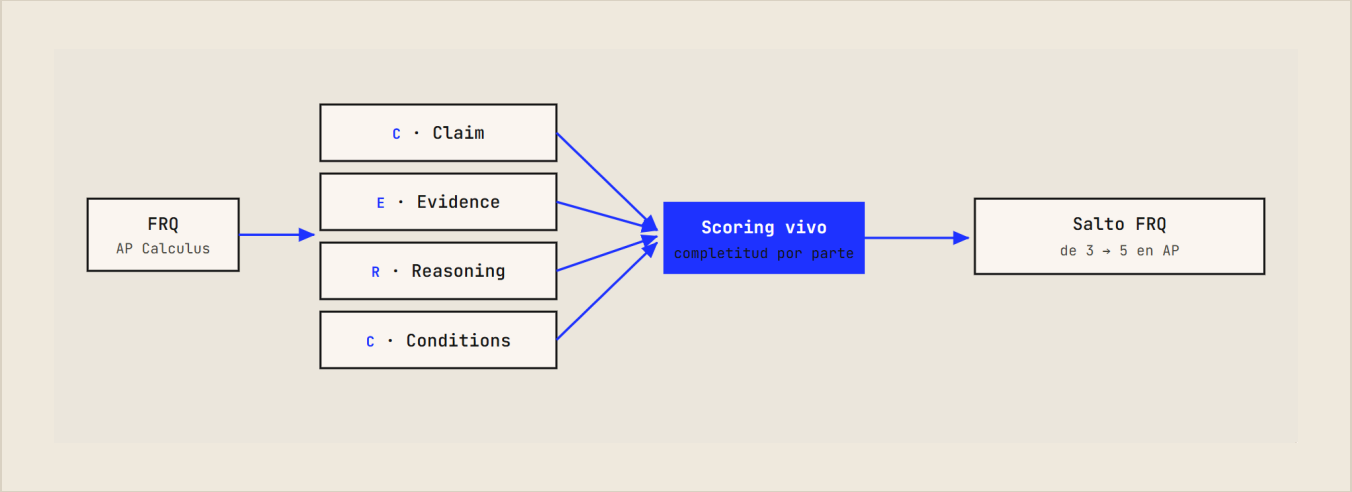


FIG. 3 — CERC MODEL: CLAIM → EVIDENCE → REASONING → CONDITIONS, THE FOUR VISIBLE AND REQUIRED MOVES OF EVERY PROOF

Completeness scoring — not semantic AI grading

Context. A semantic evaluator that judged whether the argument is “good” with a language model would have been more impressive, but dishonest about what it actually measured.

Decision. The system evaluates deterministically whether the four CERC elements are present and non-empty, with real-time visual feedback. The first habit to install is the structural integrity of the argument, not eloquence.

Accepted trade-off. It measures less “quality” and more “completeness” — but it attacks the right habit first and measures exactly what it claims to measure.

The second structural decision was the data layer with an **adapter** pattern: a mock adapter for development backed by Vercel KV, and an LMS adapter for production that preserves the correct shapes of the open educational interoperability standards.

08 BUILD AND VALIDATION

AI-FIRST, WITH SECURITY AS A DESIGN CONSTRAINT

It was built with an **AI-first** flow, assisted by Claude, at a deliberately high pace over about six weeks (March 17 to April 30), across 29 pages in Next.js 14's App Router. The stack settled on strict TypeScript, Tailwind, KaTeX for the notation and GSAP for the gamification.

Security was treated as its own front, not as an add-on: authentication with JWTs signed via *jose*, route protection by middleware with role-based access control (student / administrator), sanitization with DOMPurify, rate limiting and mandatory CSRF on every authenticated mutation. An external audit in early April detected hardening opportunities in authentication and PII handling; they were resolved by reinforcing the session model and the data isolation of the admin panel.

Validation was of two kinds. On the technical side, a test battery covered the integrity of the course data, the prerequisite logic between units, and the full flow of a session, plus the production build of all 29 pages. On the pedagogical side, the platform was prepared for a **pilot** with a small group of Calculus BC and Statistics students —anonymously identified— with their own login backed by the network's LMS real roster.

> *The measurement is not a subjective grade but the student's trajectory through the three reasoning stages and their growing CERC completeness as the scaffolding is withdrawn between Unit 1 and Unit 4.*

09 THE RESULT

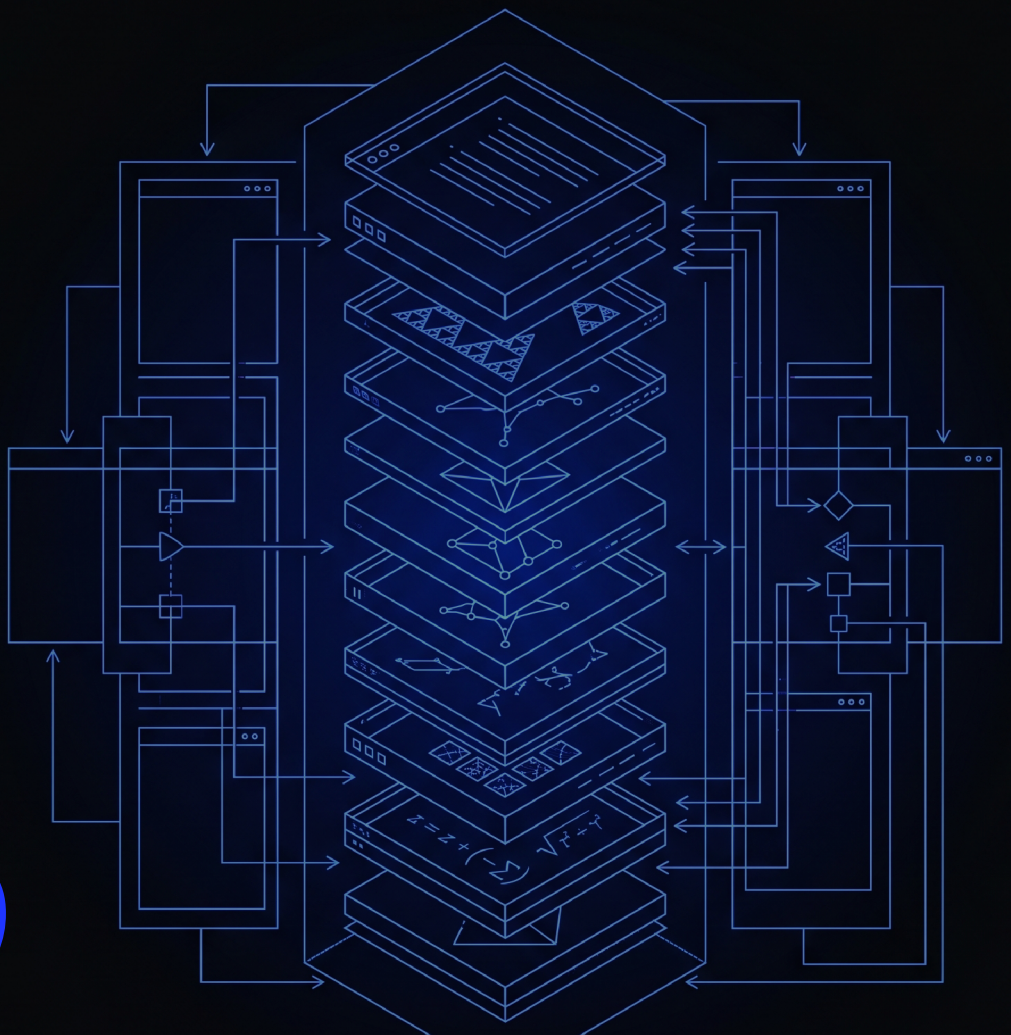
STATUS AND LEGACY

A functional platform in Beta status: the four units operational with their problem catalog, the three courses, the completeness scoring engine, the XP and badge system, the admin panel with data isolation, and the layer of interchangeable adapters. The production integration with the LMS was left as a deliberate stub, faithful to the shapes of the open educational interoperability standards, ready to connect when the production environment allowed it. The success criterion set by academic leadership was explicit: the work was approved if it brought the student to a **5** on the AP exam — the outcome, not the activity.

Pedagogical framework	**CERC** — Claim · Evidence · Reasoning · Conditions
Units · problems	**4 units · 7 / 7 / 7 / 4**
Courses covered	Calculus AB · Calculus BC · Statistics
Reasoning stages	empirical · generic · formal (data type)
Data layer	Mock (Vercel KV) ↔ the network's LMS (open educational interoperability standards)
Security	JWT (jose) · role-based middleware · CSRF · DOMPurify · rate limiting
Build	AI-first · 29 pages · interchangeable adapter layer
Status	**Beta** · the LMS as a stub faithful to the APIs

LESSONS LEARNED

- **Measurement honesty.** Resisting the temptation of the semantic AI evaluator and choosing deterministic completeness scoring: it measured exactly what it claimed to measure and attacked the right habit first —structure before eloquence.
- **Theory encoded in the types.** When the reasoning stage is a first-class data type and the units withdraw the scaffolding on a schedule, the pedagogical design stops being intention and becomes verifiable.
- **Adapter pattern.** A low-cost, high-return decision: iterating at full speed with a persistent mock without coupling to the production API, keeping its shapes intact.
- **Security as a design constraint.** On a platform with minors' data, authentication, CSRF and data isolation are not a final phase but a constraint from the first line. the AP/SAT curricular intersection

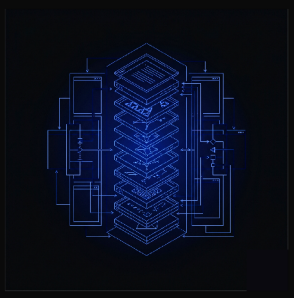


3 KNOWLEDGE INFRASTRUCTURE

The layer that grounds them all: from academic literature to research with evidence validation.

Evidence you can answer for

RESEARCH INTELLIGENCE PLATFORM



An evidence operating system that turns academic literature into traceable institutional decisions — one claim at a time, anchored to the study that supports it.

STATUS	STACK	AI	VALIDATION
Production	Next.js 16 · React 19 · Firebase	AWS Bedrock · Claude	ESSA + 26 suites

01 THE PROBLEM

CONCLUSIONS NO ONE COULD ANSWER FOR

A network of AI-accelerated personalized-learning K-12 schools makes curriculum and instruction decisions daily: which sequence to adopt, which intervention to scale, which practice to retire. Those decisions relied on AI-generated summaries that blended, in a single operation, two things that should never go together: the faithful extraction of what a study reported and the pedagogical interpretation of what that study meant for the classroom.

The result was impossible to audit. There was no way to tell what a paper's author had measured from what the model had inferred, nor to trace a recommendation back to the specific study that supported it.

The conclusions arrived as fluid, unanchored narrative: orphan sentences that sounded authoritative but that no one could verify.

On top of this was an identity problem: the previous system referenced studies by their position in an array, and every reordering silently corrupted those references.

For an institution that defines itself as evidence-based, the bottleneck was not producing text, but producing text you could answer for.

02 THE GOAL

CLAIM-LEVEL TRACEABILITY

PRIMARY GOAL

Build an evidence operating system, not a summary generator: that **every publishable conclusion had claim-level traceability**. No sentence in the final document could exist without a structured mapping to one or more source studies identified by a stable `refId`.

Alongside that, three concrete goals: irreversibly separate extraction from interpretation, apply a deterministic and reproducible methodological quality framework, and maintain a living, versioned corpus capable of incorporating new studies or re-evaluations without rebuilding entirely.

The product had to close the full cycle, from the raw paper to the documented institutional decision, so that a human reviewer could intervene at any point without breaking the chain of evidence. It is an internal admin tool for two people: the pipeline operator and a researcher from the learning science team brought in as a review peer.

03 THE SYSTEM

AN AUDITABLE NINE-STAGE PIPELINE

The system is organized as a chained nine-stage pipeline, where each stage has its own agent, its input and output contract, and a persisted, versioned artifact. The central decision, documented as **ADR-001**, was to forbid extraction, evaluation, clustering, synthesis and writing from happening in a single model call: the separation of layers is what makes the system auditable.

Ingestion→Extraction→Clustering→Synthesis→Triangulation→Claims→Chapters→Final→Editorial

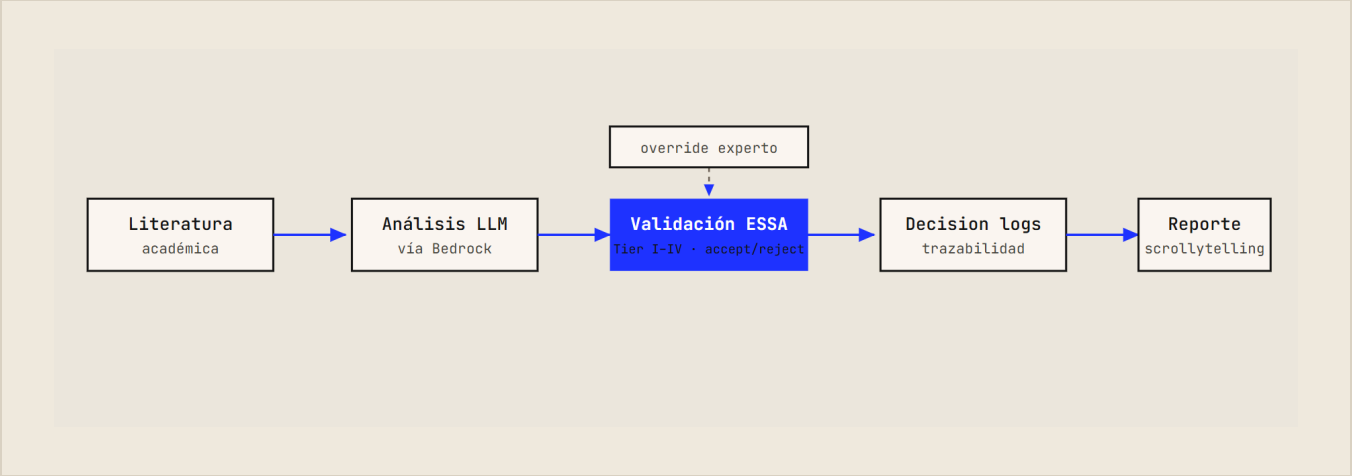


FIG. 1 — PIPELINE LITERATURE → LLM → ESSA VALIDATION → DECISION LOGS → REPORT · EACH STAGE PERSISTS A VERSIONED ARTIFACT WITH IDENTITY BY IMMUTABLE REFID, NEVER BY POSITIONAL INDEX

Separate extraction from interpretation — never in the same model call

Context. The previous system blended in one operation what the study measured and what it meant for the classroom, and referenced studies by their position in an array — every reordering corrupted the references.

Decision. The quality framework (ADR-012) forks the *appraisal* by study type —ESSA for quantitative, CASP for qualitative, both for mixed— and encodes the ESSA decision tree as explicit deterministic rules in the prompt: `randomAssignment + controlGroup + adequate sample → Strong`. The agent does not infer the tier freely.

Accepted trade-off. More stages and more artifacts than a one-step summarizer — but it is what makes every conclusion traceable back to the study that supports it. The 21 decisions were recorded as ADRs.

04 THE FOUNDATION

THREE SYNTHESIS STANDARDS, TURNED INTO SOFTWARE

The architecture translates two recognized evidence-synthesis standards into the software domain, adding a third for the qualitative. The guiding principle –**fidelity first**– comes straight from the discipline of systematic review: represent the study before interpreting it.

FRAMEWORK	WHAT IT BRINGS	HOW IT OPERATES IN THE SYSTEM
ESSA	Strength of evidence in four tiers	Deterministic decision tree encoded in the prompt; classifies every reference.
PRISMA 2020	Transparency of systematic reviews	Automatic 12-item audit against the system's artifacts at closing.
CASP	Appraisal of qualitative evidence	Forks the analysis so that quantitative, qualitative and mixed are not forced into a single logic.

The four ESSA tiers structure the entire corpus, from the strongest to the lowest empirical evidence:

TIER 1 · STRONG — RCT TIER 2 · MODERATE — quasi-experimental

TIER 3 · PROMISING — correlational

TIER 4 · RATIONALE — logic model

> *The quantitative is synthesized by effects and moderators; the qualitative by themes and mechanisms; the mixed is explicitly integrated through triangulation. Represent the study before interpreting it.*

05 THE VALIDATION

THE FUNCTIONAL CORE: HUMAN REVIEW REFERENCE BY REFERENCE

Validation happens in **/research** through an explicit human review flow. After running ESSA validation, the researcher and the operator reviewed each reference one by one, with three possible verdicts —**Accepted**, **Warning** or **Rejected**— on the tier the agent had assigned according to the deterministic tree. The system allows a human *override*: the model's judgment is a proposal, not a verdict.

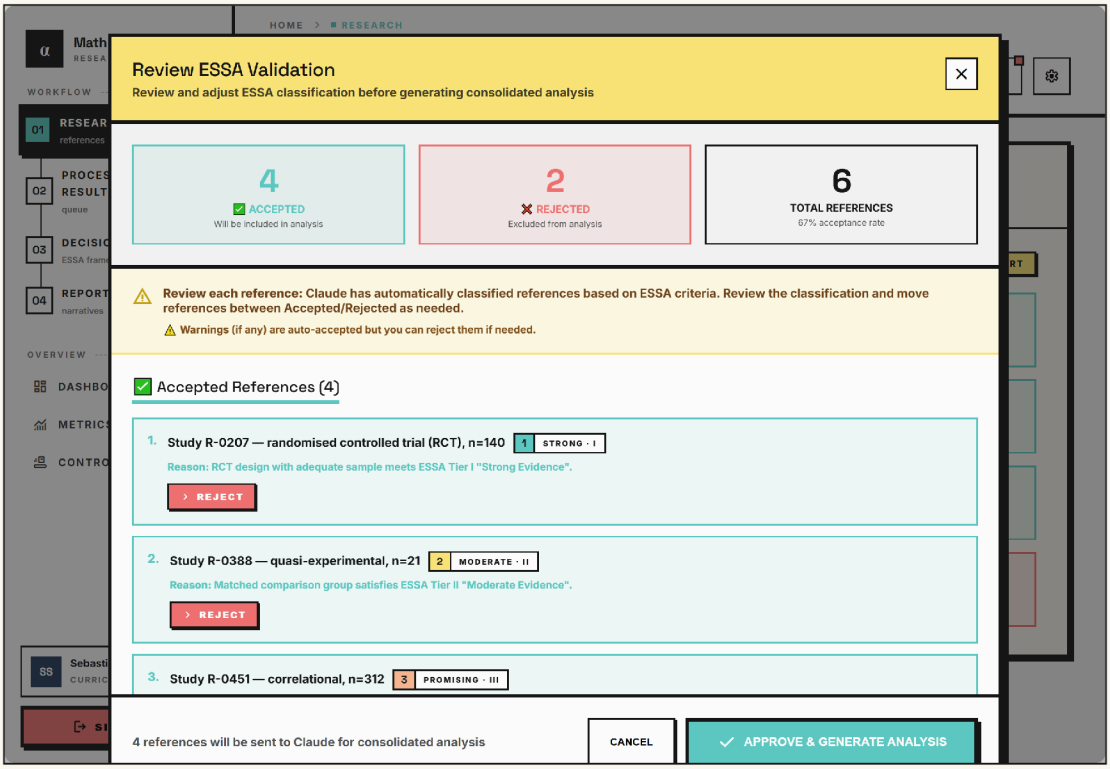


FIG. 2 — /RESEARCH WITH ESSA VALIDATION · EACH REFERENCE SHOWS ITS PROPOSED TIER AND THE ACCEPTED / WARNING / REJECTED OVERRIDE · NO SELECTION IS APPROVED WITHOUT PASSING THE HUMAN FILTER

06 THE SECOND FILTER

AUTOMATIC PRISMA AUDIT AND DECISION LOGS

The closing of the final document adds a second layer of automatic control, the **PRISMA audit**, which verifies twelve items of the standard against the system's artifacts —from eligibility criteria and per-study risk of bias to synthesis results and limitations— and reports total, partial or non-compliant adherence. Thus, no research is published without passing the double filter: human review over the evidence and automatic verification over the report. The **/decision-logs** view records each institutional decision with its chain of evidence intact.

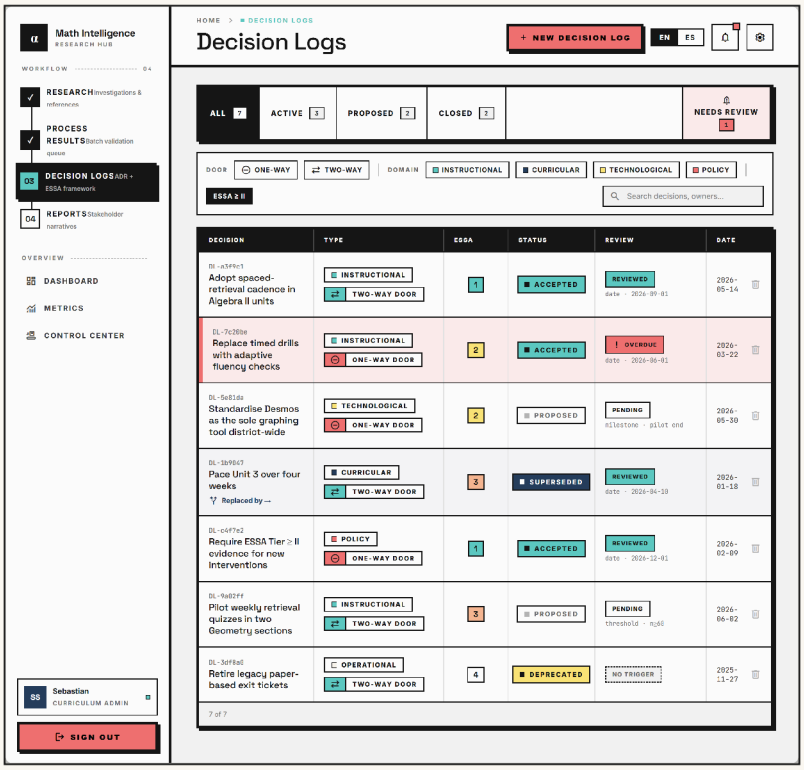


FIG. 3 — /DECISION-LOGS · EACH DECISION IS ANCHORED TO THE RESEARCH AND TO THE STUDIES THAT SUPPORT IT · TRACEABILITY FROM THE PAPER TO THE DOCUMENTED DECISION

07 FLAGSHIP CASE

MEASURING THE 2X FACTOR

Measuring the 2x Factor: Speed and Mastery Depth Metrics for Adaptive HS Math. The thesis: the claim that adaptive math platforms can double learning speed **cannot be verified or refuted** with the available evidence as it is structured today —

no study operationalizes speed and mastery depth simultaneously against an elite benchmark.

FINDINGS, WITH EFFECT SIZES

d 0.10–0.68 · g 0.37 — Adaptive platforms produce moderate positive effects on math performance, but no study measured them against elite benchmarks like 800 on SAT Math or 5 on AP.
speed ≠ depth — Advancing quickly within a platform can produce shallow understanding; speed metrics must be separated from conceptual mastery metrics.
d 0.30–1.34 — Interleaved and spaced practice produce the most robust effects on long-term retention; distributed practice doubles or triples retention versus massed practice.
depth > acceleration — Deep precalculus mastery predicts performance in college calculus with more than double the impact of simply having taken the course.
wheel-spinning — Extensive practice without achieving mastery is predictable from early indicators; time on task and number of exercises are insufficient to evaluate real learning.

08 FLAGSHIP CASE

THE PROPOSED FRAMEWORK AND ITS EVIDENCE BASE

DUAL-AXIS SPEED-MASTERY MEASUREMENT FRAMEWORK

(1) a valid speed metric distinct from time on task; (2) a mastery depth metric anchored in verified foundational competencies; and (3) a calibrated elite-performance benchmark.

The unresolved tension. The speed–depth contradiction remains open because no study operationalized both metrics at once against an elite benchmark; the effects for high-performing students under extreme pressure remain undocumented. The implication is direct: any platform that claims acceleration («2x») should empirically demonstrate the operationalization of its metrics and report effects on external assessments before declaring itself effective. It is the framework that makes the model's central promise *falsifiable*.

Evidence clusters	29
Mapped claims	45
Sources	45 · all STRONG
Report confidence	Moderate

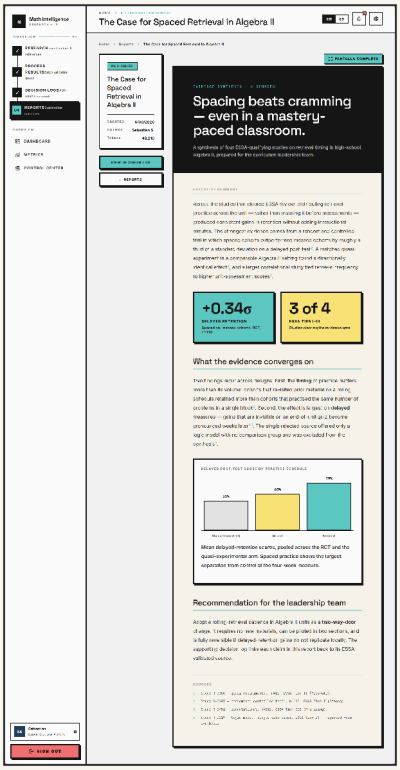


FIG. 4 — GENERATED SCROLLYTELLING REPORT · A PIECE OF RESEARCH TURNED INTO NAVIGABLE NARRATIVE, WITH EACH CLAIM ANCHORED TO ITS SOURCES · EXPORTABLE TO PDF

09 V0 GENESIS

SEVEN MANUAL STUDIES THAT JUSTIFIED THE PLATFORM

Before the platform there was a proof of concept: **seven studies synthesized by hand with Gemini**. They proved that turning literature into actionable research was valuable — and that doing it by hand neither scaled nor left an auditable trace. That v0 is what justified building the evidence operating system.

1. **Academic Audit Report.** Longitudinal audit of high-school math performance (SAT Math and AP) against reference institutions. ch. 07
2. **Cognitive Architecture of the SAT.** The 650→800 gap as structural fluency and arbitrage with Desmos. ch. 08
3. **AP/SAT Curricular Intersection.** The algebraic root system that supports advanced Calculus and Statistics. ch. 09
4. **Automation Threshold Roadmap.** The automaticity threshold as a bridge to accelerated Calculus. ch. 10
5. **MS Persistence vs SAT Stamina.** From 7th-grade persistence to the cognitive stamina of the 1550. ch. 11
6. **Elite Freshman Profiles.** Recursive audit of SAT Math compression and STEM readiness. ch. 12
7. **Technical Calculation Protocol.** Proprietary metrics for the daily math sprint. ch. 13

The seven reappear as their own chapters in this book: the manual v0 was the origin of the corpus that the platform later scaled and formalized.

10 THE CORPUS, AT A GLANCE

THE PLATFORM'S OUTPUT AT SCALE

From seven manual studies to a validated, living, versioned body of research. This is what the platform produced:

19
Research pieces
542
Studies synthesized
465
References

FIG. 5 — CORPUS AT A GLANCE · EACH FIGURE IS THE PRODUCT OF THE NINE-STAGE PIPELINE WITH A DOUBLE FILTER · THE PREVIOUS MANUAL PROTOTYPE WITH GEMINI STARTED FROM 7 STUDIES

Each of these research pieces went through the ESSA decision tree, the human review reference by reference, and the automatic twelve-item PRISMA audit. The validation interface with human override and the PRISMA audit are the two guarantees that distinguish this system from a summary generator: every conclusion that comes out is traceable back to the studies that support it. It is, in practice, the infrastructure that feeds the research corpus of the following chapters.

11 APPENDIX

CORPUS INDEX — THE 19 RESEARCH PIECES

RESEARCH	STUDIES	REFS.	DATE
Measuring the 2x Factor: Speed and Mastery Depth Metrics for Adaptive HS Math	211	195	2026-05-07
Critical Variables for Big Bang Adaptive Math Curriculum Implementation in High School	58	53	2026-05-07
Change Management and Leadership Resistance in Non-Traditional School Reform	28	14	2026-05-07
Educational Data Mining and Learning Analytics in Mathematics	23	20	2026-04-06
Advanced Placement Mathematics Achievement Factors	23	13	2026-04-06
Mastery-Based Learning and Spaced Practice in Mathematics	19	11	2026-04-06
Interleaved Practice and Problem-Type Discrimination	18	8	2026-04-06
From Computation to Proof: Pedagogical Transitions for Advanced Adolescents in Self-Directed Environments	17	17	2026-04-21
Computational Fluency and Mathematical Modeling Integration in Advanced Secondary Curricula	16	16	2026-04-21
Self-Regulated Learning in Digital Mathematics Environments	16	11	2026-04-06
Beyond AP and SAT: Assessment Frameworks for Mathematical Maturity — International Models and Admissions Signaling	15	15	2026-04-21
Sequencing Advanced Mathematics for Adolescent STEM Readiness: International Comparative Analysis	15	15	2026-04-21
Conceptual vs. Procedural Knowledge Development in Mathematics	15	8	2026-04-06
Mathematics Anxiety and Emotional Regulation in Digital Learning	14	9	2026-04-06
Sustaining Mathematical Motivation: Wellbeing, Coach Support, and Resilience in Accelerated Adolescent Math Learning	11	19	2026-04-21
Large Language Models and Conversational Tutoring in Mathematics	11	11	2026-04-06
Cognitive Load Management in Autonomous Digital Learning	11	9	2026-04-06

Intelligent Tutoring Systems Effectiveness in Secondary Mathematics	11	11	2026-04-06
Projected Learning Gains of LLM-Based Conversational Tutoring in Secondary Mathematics: A Monte Carlo Simulation Based on AutoTutor Evidence	10	10	2026-04-12

12 BUILD AND RESULT

AI-FIRST, SUBJECTED TO DATA CONTRACTS

I built the platform AI-first and at high cadence, weekends included, on a deliberately modern stack: Next.js 16 with App Router, React 19 and Tailwind v4, taking on their breaking changes. The inference engine is AWS Bedrock: **Claude Sonnet 4.6** for all classification and publishing stages, and **Opus** for the interactive assistants. The discipline that sustains confidence in the output is validation with Zod at every data edge, on read and on write, against centralized contracts: an artifact that does not validate does not enter the corpus.

Each agent's runners are decoupled from the Bedrock client through an injectable interface, which makes it possible to run the full pipeline against a mock in the tests. CI runs typecheck, lint, tests and build on every push, with **26 suites** across contracts, integration, regression and unit tests. The platform shipped to production as an admin app, serving the full cycle from the paper to the documented decision.

Commits	~773 · Mar 31 → Jun 5 2026
Pipeline stages	9 · ingestion → editorial
Architecture decisions	21 ADRs recorded
Quality frameworks	ESSA + CASP + PRISMA 2020 (12 items)
Data validation	Zod on read and write, at every edge
Test suites in CI	26 · contracts, integration, regression, unit
Admin users	2 · operator + learning science researcher
Status	In production

LESSONS LEARNED

- **On architecture.** Auditability is not added later, it is designed from the first layer: separating extraction from interpretation and anchoring identity in an immutable refId resolved an entire class of bugs at the root that no later patch would have contained.
- **On AI and expert judgment.** Encoding the ESSA tree as deterministic rules inside the prompt, instead of letting the model reason out the tier freely, made the result reproducible and put judgment where it belongs: in the human override reference by reference.
- **On design.** Working the validation flow alongside a learning science researcher tuned the interface more than any written specification — watching someone walk through the review live showed exactly where traceability helped and where it got in the way. And the 21 ADRs proved their worth when operating on a stack with breaking changes.

What the portfolio demonstrates, taken together

JUDGMENT IS THE PRODUCT.

Six pieces for a single mathematics program. Seen from a distance, they are not six apps: they are six exercises of the same muscle — deciding what to measure, translating learning science into software, and sustaining quality without letting it erode.

01 THE ARC

FROM SEEING, TO TRAINING, TO GROUNDING

The portfolio spans an entire program, not an isolated feature. First **seeing**: an analytics layer that made the risk of ~1,600 students legible and turned it into morning action with no manual work Sec. 1. Then **training**: two platforms that tackle the exam's most expensive difficulty leaps —the SAT's calculator fluency and the AP's argumentation— translating learning-science distinctions into software mechanics Sec. 2. And finally **grounding**: an infrastructure that turns academic literature into traceable curriculum decisions, every claim anchored to its source Sec. 3.

Each product solved its own problem and, at the same time, enabled the next: the analytics surfaced the gap the training attacked; the need to ground those decisions drove the evidence platform.

It is not a catalog of demos: it is a system designed in layers, where the decision of what to build always came from a diagnosis, not from a feature list.

02 THE COMMON THREAD

WHAT RECURS ACROSS THE SIX PIECES

TRACEABLE JUDGMENT

Every structural decision was documented with its *why* and its accepted trade-off – from the embedded, persistent Desmos to deterministic scoring by completeness. The artifact shows the judgment, not just the result.

LEARNING SCIENCE TURNED SOFTWARE

CERC as a data type, the five risk metrics, scaffolding fading by streak, the reasoning stages. Learning principles turned into measurable mechanics, not slogans.

INSTITUTIONALIZED QUALITY

Gates that fail the build, a double human filter + evidence validation, data contracts in CI. Quality as a property of the system, not a manual review that erodes under pressure.

DESIGN HONESTY

Choosing the defensible over the flashy: honest scoring over an impressive but opaque semantic evaluator; a stub faithful to its contracts before a faked integration.

03 WHAT WAS MEASURED, AND WHAT WASN'T

THE HONEST FRAMING OF IMPACT

Every piece reached production and passed validation: mathematical and instructional peer review, automated gates, test batteries and data contracts. **That was measured — product quality.**

> *Measuring the sustained effect on learning at scale would have required a rollout and an evaluation window I did not control. There is, therefore, no student-outcome metric; and faking one would be exactly the dishonesty the rest of the work avoids.*

What this book can defend, under any line of questioning, is the judgment: how it was decided what to measure, how theory was translated into software, and how quality was sustained. It is what remains when the servers go dark — and it is, precisely, what these products were designed to demonstrate.

04 CODA

BUILT IN SIX MONTHS, WITH AI AS LEVERAGE

Everything was built within a six-month window using an AI-first development model: AI accelerated the *how to build it*; the decision of **what to measure and why** was always my own. The portfolio is the evidence of that division of labor — and that, well directed, this leverage lets a single person reason and deliver at the scale of a team.