What We Learned Building AI Features For TalentLMS

Yannis Rizos | Epignosis | Mindstone Practical AI Meetup | Athens | 18-06-2025

The Town Planner's View

AI is not the product - it's one component in a larger system

The system, not the model, is what users interact with

AI development is 5% model behavior, 95% engineering effort

At the scale of TalentLMS, you can't just experiment and hope for the best

What We've Built

TalentCraft

AI-powered authoring (full units from single prompt)

Inline Authoring Tools

embedded in standard course editor

Course Translation

entire courses across languages

Skills

AI-powered skill definition and content linking

Learning Coach

AI assistant for learners (soon)

Learning Paths Personalization

adaptive course sequences (soon)

LLMs Are Different

Remote

external API calls, not local libraries

Probabilistic

same input may not yield same output

Expensive

measurable monetary cost for every call

Non-deterministic

behavior varies based on subtle inputs

Unpredictable latency

especially under load

What This Actually Means

Need timeouts, retries, isolation, fallback planning

Must handle probabilistic nature of responses

Hallucinations cannot be eliminated - they must be contained

Reasoning-heavy or multi-step tasks suffer from high latency

Avoid LLMs in critical path unless absolutely necessary

Cache results, batch jobs, defer low-priority processing

Who Owns What

AI team

Prompt handling, API surfaces, infrastructure

Logging and observability, recovery logic

Cost and performance control

Product teams

Do not call LLMs directly

Interact with stable, well-defined interfaces

Focus on user-facing value, not prompt logic and LLM internals

Prompts Are Not Strings

Prompts are engineering assets

Centralized

Versioned

Tested

Tightly constrain retrieval and response scope

Logic must never be encoded into prompts

Control flow and workflows implemented in code

Prompts are purely content-generating inputs

Failure Is Expected

System designed for failure

Retry logic, circuit breakers, graceful degradation

Precomputed fallback outputs

Human-in-the-loop mechanisms

Offline modes required

Product must function if model unavailable

Cached responses or degraded workflows

Avoid hard failures in core flows

Good UX mitigates bad AI

Metrics From Day One

Every AI job logs structured events

Events visualized and exposed to Product

Feedback loop: Errors, latency, user behavior

Same event infrastructure powers rate limiting

Nothing ships without observability

Architecture Drives Decisions

Testability

Observability

Latency

Explainability

Cost control

Version stability

Architecture Drives Decisions

Model selection is deliberate

Smaller, faster, cheaper models for narrow/frequent tasks

Larger, expensive models for quality-intensive work

Modular provider, model and prompt configurations

Concurrency is critical

AI jobs run in parallel when safe

Use async primitives, avoid sequential execution

one more slide ;)

The Big Picture

Focus on the system, not the model

Engineering discipline matters more than AI experimentation

Design for failure first, not as afterthought

Make decisions based on metrics, not intuition

Organizational boundaries matter

Path from prototype to production is 95% engineering

Thank you!