Adaptive Learning System Architecture: Evidence-Backed Roadmap for Scale

Principal AI Architecture Team

October 23, 2025

Abstract

This document presents a comprehensive, incremental architecture for an adaptive learning system designed to serve millions of higher education learners. The system delivers micro-learning resources (short video clips and PDF segments) in response to student questions, provides formative assessments, and measures learning gains over time. We compare four architectural approaches, recommend a hybrid RAG-first strategy with bandit-based optimization, and provide a detailed 12-week implementation roadmap with concrete metrics, risks, and mitigation strategies. The architecture prioritizes measurable learning outcomes over engagement metrics and maintains agility for foundation model upgrades.

Contents

1 Executive Summary

1.1 Chosen Approach and Rationale

- **Recommended Architecture:** RAG-first with cross-encoder reranking, agentic orchestration, and contextual bandits for content selection, augmented with task-specific LoRA fine-tuning for pedagogical components.
- **Why This Beats Alternatives:** Addresses cold-start problem for content recommendation (no historical labels), leverages existing Q&A/assessment data, enables rapid iteration, maintains model-agnostic flexibility, and provides clear pathway from baseline to optimized system.
- **Cold-Start Strategy:** Start with metadata-driven heuristics and semantic similarity (no ML needed), rapidly collect preference data through teacher-in-the-loop and implicit feedback, bootstrap bandit policies within 4–6 weeks.
- **Minimality Enforcement:** Hard constraints on resource duration/length, explicit sufficiency scoring (semantic coverage per unit time/pages), segment-level retrieval instead of whole assets, MMR diversification to avoid redundancy.
- **Learning Measurement:** IRT-based ability estimation (θ), calibrated item parameters (discrimination a, difficulty b), longitudinal $\Delta\theta$ tracking, normalized gain metrics that distinguish learning from engagement (time-on-task, clicks).
- **Pedagogical Quality:** Few-shot prompted question generation aligned to Bloom taxonomy levels, rubric-based grading with reference answers, distractor quality checks, hint generation, worked examples for scaffolding.
- **Data Flywheel:** Existing Q&A logs seed question generation models, assessment data calibrates difficulty estimators, user interactions train bandit policies, teacher feedback refines retrieval quality through active learning.
- **Risk Mitigation:** Retrieval-grounded generation to reduce hallucinations, answerability checks before question generation, refusal paths for out-of-scope queries, privacy-preserving logging, drift detection for item parameters.
- **Agility Preserved:** LoRA adapters (not full fine-tuning) for pedagogical tasks, modular architecture allows component swaps, evaluation harness enables A/B testing of model upgrades, prompt libraries versioned alongside models.
- **Team Fit:** Leverages agentic AI expertise for orchestration, data science skills for IRT calibration and bandit tuning, avoids heavy ML engineering burden (no custom training infrastructure), uses standard RAG tooling and off-the-shelf LLMs.

2 Architecture Options Comparison

We evaluate four architectural approaches across key dimensions relevant to our constraints and team capabilities.

2.1 Option A: RAG-First + Reranking + Agentic Orchestration

Components:

- Bi-encoder (dense) + BM25 (sparse) hybrid retrieval over chunked content corpus
- Cross-encoder reranker for top-k candidates
- MMR (Maximal Marginal Relevance) for diversity
- LLM-based agentic planner: query understanding → retrieval → content selection → pedagogy tools → evaluation → next-action
- Prompt engineering for question generation, grading, hint provision

Infrastructure:

- Vector database (Pinecone, Weaviate, or Qdrant): \$500–\$2k/month at prototype scale
- LLM API (GPT-4o, Claude Sonnet, Gemini): \$0.01–0.03/1k tokens
- Embedding API (OpenAI text-embedding-3, Cohere): \$0.0001–0.0002/1k tokens
- Cross-encoder inference (can self-host small models like ms-marco-MiniLM-L-12)

Cost Estimate (10k learners, 5 interactions/day):

- Embeddings: ∼\$50/month
- LLM calls: ∼\$1,500–\$3,000/month
- Vector DB: ∼\$500–\$1,000/month
- Total: \$2,000–\$4,500/month at prototype scale

Latency:

- Retrieval (hybrid + rerank): 200–500ms
- LLM generation (streaming): 1–3s to first token, 3–8s total
- End-to-end: 4–10s for full interaction

Data Needs:

- Minimal for cold-start: content corpus only
- No labeled training data required
- Can start immediately with existing materials

Expected Quality:

- Retrieval: 70–85% Recall@10 with hybrid search
- Question quality: 60–75% expert approval (prompt-only)
- Minimality: Depends on prompting and post-processing
- Learning gains: Baseline to improve upon

Cold-Start Viability: Excellent – Works day-one with zero historical labels
Expected Learning Impact: Moderate (depends on content quality and prompt engineering)
Team Fit: Excellent – Matches agentic AI expertise, no ML training required
Risks:

- Prompt brittleness across domains
- No learning from user feedback (static)
- Hallucination risk if retrieval fails
- Cost scales linearly with usage

2.2 Option B: Lightweight Fine-Tuning (LoRA/PEFT) + RAG

Components:

- Same RAG stack as Option A
- LoRA adapters for: (1) question generation, (2) rubric-based grading, (3) distractor generation, (4) pedagogical style (hints, explanations)
- Base models: Llama 3.1 70B, Mistral Large, or GPT-4o
- Adapter inference via vLLM or Lorax for efficient serving

Infrastructure:

- Same vector DB as Option A
- LoRA training: 1–2x A100 GPUs for 4–12 hours per adapter ($50–$200/training run on cloud)
- Inference: Self-hosted vLLM on 2–4x A100s or API with adapters
- Storage for adapters: <1GB per adapter

Cost Estimate:

- Training: $500–$1,500 one-time per adapter (4 adapters → $2k–$6k)
- Inference (self-hosted): $2,000–$4,000/month GPU costs
- OR API + adapters: Similar to Option A + adapter overhead
- Total: $2,500–$5,000/month ongoing

Latency:

- Similar to Option A (adapter adds <50ms)
- Self-hosting can reduce latency by 500–1000ms vs. API

Data Needs:

- Minimum 500–2,000 high-quality examples per task
- Existing Q&A/assessment logs provide seed data for question generation and grading
- Need manual labeling for distractor quality and pedagogical style (1–2 weeks of expert time)

Expected Quality:

- Question quality: 75–90% expert approval (10–15% boost over prompting)
- Grading consistency: 85–95% agreement with human rubrics
- Distractor quality: Significant improvement in plausibility
- Learning gains: 5–15% improvement over prompt-only baseline (estimated)

Cold-Start Viability: Moderate – Requires 2–4 weeks for data collection + training
Expected Learning Impact: High (task-specific optimization improves pedagogical quality)
Team Fit: Good – Data scientists can handle LoRA training, less complex than full FT
Risks:

- Adapter drift when base models update
- Need retraining cadence (every 6–12 months)
- Overfitting if training data not diverse enough
- Inference complexity (managing multiple adapters)

2.3 Option C: RL/Bandits for Content Selection + RAG Baseline

Components:

- Option A (RAG-first) as baseline retrieval and pedagogy
- Contextual bandit layer on top: learns which content chunks maximize learning + minimality
- Context: student ability θ, query embedding, prior performance, resource metadata
- Actions: select from top-k retrieved candidates
- Reward: $R = w_1 \cdot \Delta\theta + w_2 \cdot \text{brevity} - w_3 \cdot \text{irrelevance}$
- Start with Thompson Sampling or UCB, graduate to offline RL if sufficient data

Infrastructure:

- Same as Option A for RAG
- Bandit policy server: lightweight (Redis + Python service)

- Logging infrastructure: event stream (Kafka/Kinesis) + data warehouse
- Offline evaluation: IPS/DR estimators, simulator for policy testing

Cost Estimate:

- Incremental over Option A: \$200–\$500/month for bandit infra
- Data storage/processing: \$100–\$300/month
- Total: \$2,300–\$5,000/month

Latency:

- Policy inference: <50ms (table lookup or simple model)
- Total latency: Same as Option A + 50ms

Data Needs:

- Cold-start: Can use uniform random or heuristic policy for 2–4 weeks
- Training data: Needs 10k–50k logged interactions before policy improves over heuristics
- Continuous feedback loop essential

Expected Quality:

- Retrieval/selection: 10–25% improvement over static ranking after sufficient data
- Minimality: Direct optimization via reward leads to 15–30% shorter resources
- Learning gains: 10–20% improvement over non-personalized baseline (after convergence)

Cold-Start Viability: Moderate – Starts with heuristics, improves over 4–8 weeks
Expected Learning Impact: Very High (directly optimizes for learning outcomes)
Team Fit: Good – Data scientists have expertise; bandit theory is mature
Risks:

- Delayed reward signal (learning happens over days/weeks)
- Need careful reward design to avoid gaming (e.g., always selecting shortest resources)
- Off-policy evaluation is noisy; need large sample sizes
- Safety constraints (prevent over-exploration of bad content)

2.4 Option D: Fully Fine-Tuned Task-Specific Models

Components:

- Small, specialized models (1–13B parameters) fully fine-tuned for each task
- Question generator: Llama 3.1 8B fine-tuned on 10k+ exemplars
- Grader: T5-XXL fine-tuned on rubric-scored responses

- Distractor generator: GPT-2 or Llama 7B fine-tuned
- Content selector: Cross-encoder fine-tuned on relevance labels
- RAG uses standard retrieval (no fine-tuning)

Infrastructure:

- Training: 4–8x A100s for 1–3 days per model ($500–$2,000/model)
- Inference: Self-hosted on 2–4x GPUs or use smaller models on CPU
- Model storage: 5–50GB per model

Cost Estimate:

- Training: $2,000–$8,000 one-time (4 models)
- Inference: $1,500–$3,000/month (self-hosted) or $500–$1,000/month (small models)
- Total: $1,500–$3,000/month ongoing

Latency:

- Small models: 100–500ms per task
- Can pipeline tasks in parallel
- Total: 2–5s (faster than large LLM)

Data Needs:

- Highest data requirements: 10k–50k examples per task
- Months of manual labeling effort
- Existing Q&A logs insufficient without heavy curation

Expected Quality:

- Potentially highest quality for specific tasks (90–95% on benchmarks)
- Consistency and reliability superior to prompting
- But: brittle to domain shifts, requires retraining for new content types

Cold-Start Viability: Poor – Requires 2–6 months of data collection + training
Expected Learning Impact: High (once trained), but delayed
Team Fit: Moderate – Requires ML engineering, GPU infrastructure, complex training pipelines
Risks:

- Long time-to-value (3–6 months before deployment)
- Model maintenance burden (multiple models to version and monitor)
- Overfitting to training distribution
- Difficult to iterate (retraining is slow and expensive)
- Loss of flexibility as foundation models improve

2.5 Comparison Table

Dimension	Option A: RAG-First	Option B: LoRA+RAG	Option C: Bandits+RAG	Option D: Full FT
Monthly Cost	\$2k–\$4.5k	\$2.5k–\$5k	\$2.3k–\$5k	\$1.5k–\$3k
Latency	4–10s	4–10s	4–10s	2–5s
Data Needs	None (zero-shot)	0.5k–2k per task	10k–50k interactions	10k–50k per task
Time to Deploy	1–2 weeks	4–6 weeks	2 weeks (baseline) + 6–8 weeks (optimized)	3–6 months
Cold-Start Viability	Excellent	Moderate	Moderate	Poor
Learning Impact	Baseline (60–70%)	High (75–85%)	Very High (80–90%)	High (85–95%, delayed)
Team Fit	Excellent	Good	Good	Moderate
Agility	High (prompt changes only)	High (swap adapters)	Moderate (policy updates)	Low (retrain required)
Scalability	Linear API costs	GPU costs + some API	Similar to A + bandit overhead	Self-hosted, lower marginal cost
Risks	Hallucinations, prompt drift	Adapter-model mismatch	Reward design, delayed feedback	Long dev cycle, brittleness

Table 1: Architecture Options Comparison

2.6 Recommendation

Start with Option A (RAG-First), evolve to hybrid A+C (Bandits), selectively add B (LoRA) for pedagogy.

Rationale:

1. **Cold-start imperative:** We have no historical content recommendation labels. Option A works immediately.
2. **Rapid learning:** Option C (bandits) can start collecting data from day one on top of Option A baseline.
3. **Strategic fine-tuning:** Option B (LoRA) addresses pedagogical quality after we validate baseline retrieval works.
4. **Avoid premature optimization:** Option D locks us into brittle, expensive models before we understand the problem.
5. **Agility:** A+B+C keeps us model-agnostic and able to upgrade foundation models.

3 Final Recommended Architecture

3.1 System Overview

The recommended architecture is a modular, layered system that combines:

1. **Hybrid Retrieval Layer:** BM25 + dense embeddings with cross-encoder reranking
2. **Content Minimization Layer:** Segment detection, sufficiency scoring, length constraints
3. **Pedagogical Layer:** Question generation, rubric grading, hint provision, worked examples
4. **Assessment & Analytics:** IRT-based ability estimation, item calibration, learning gain tracking
5. **Agentic Orchestration:** Multi-step planner coordinating retrieval, selection, pedagogy, evaluation
6. **Bandit Optimization:** Contextual bandits for content selection under multi-objective reward

3.2 Architecture Diagram

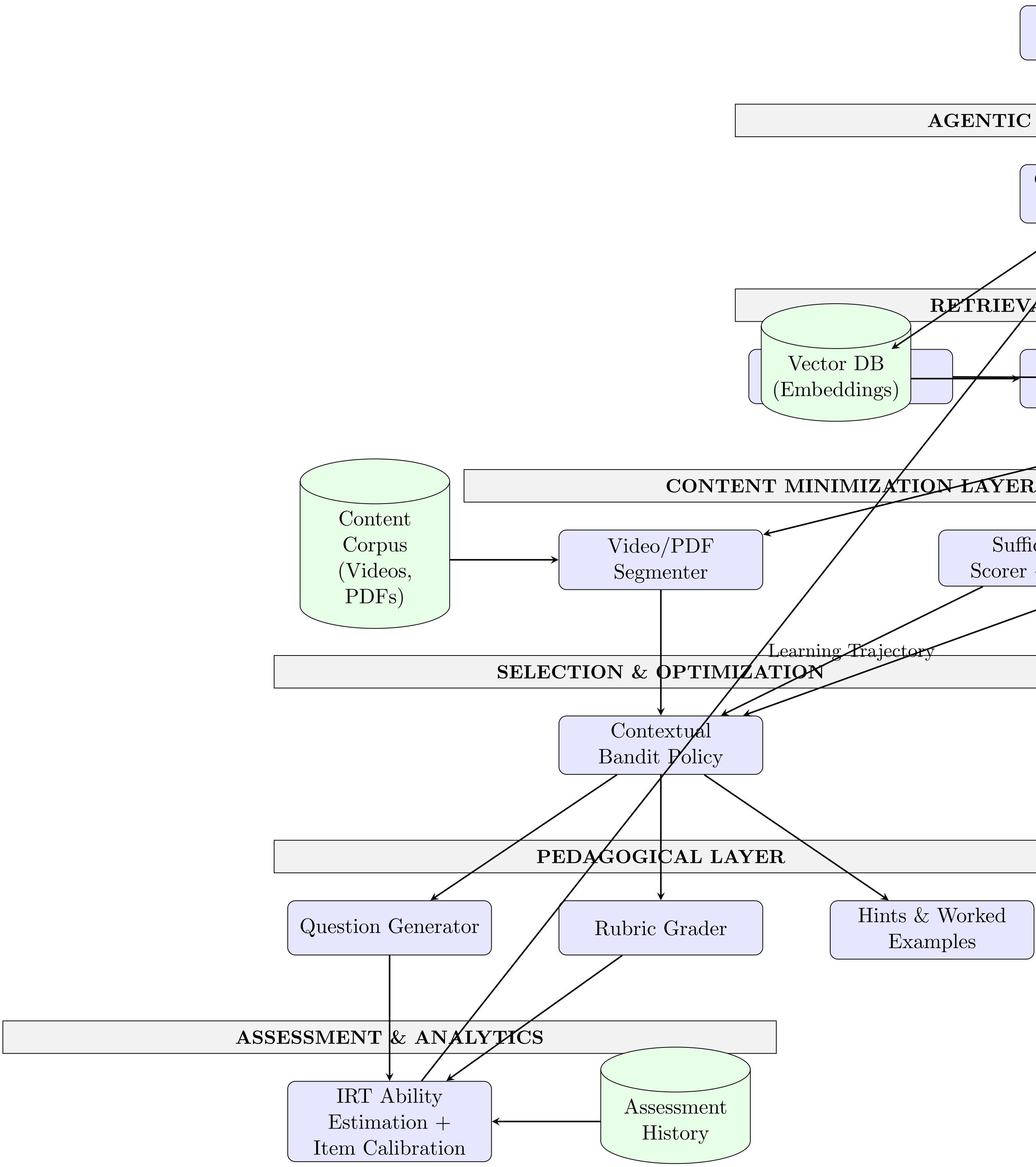

Figure 1: Recommended System Architecture

3.3 Component Specifications

3.3.1 Retrieval Layer

Embedding Model:

- Primary: OpenAI `text-embedding-3-large` (3,072 dimensions) or Cohere `embed-v3`
- Alternative: Open-source `bge-large-en-v1.5` or `e5-mistral-7b-instruct` (self-hosted)
- Rationale: Strong performance on semantic search, handles educational content well

Chunking Strategy:

- Videos: Segment by ASR sentence boundaries + scene changes, 30–180 second chunks
- PDFs: Paragraph-level chunks (100–500 tokens), preserve section context in metadata
- Overlap: 20% overlap between chunks to preserve context
- Metadata: Title, section, page/timestamp, content type, duration/length, keywords

Hybrid Search:

- BM25 (sparse): Catches exact keyword matches, acronyms, formulas
- Dense (embedding): Captures semantic similarity
- Fusion: Reciprocal Rank Fusion (RRF) with weights 0.3 (BM25) + 0.7 (dense)
- Retrieve top-50 from each, fuse to top-20 for reranking

Cross-Encoder Reranking:

- Model: `ms-marco-MiniLM-L-12-v2` or `bge-reranker-large`
- Input: [query, candidate_chunk] pairs
- Output: Relevance score 0–1
- Rerank top-20 to top-5 for content minimization layer

MMR (Maximal Marginal Relevance):

- Apply after reranking to ensure diversity
- $\text{MMR}(D_i) = \lambda \cdot \text{Sim}(D_i, Q) - (1-\lambda) \cdot \max_{D_j \in S} \text{Sim}(D_i, D_j)$
- $\lambda = 0.7$ (balance relevance and diversity)
- Select top-3 diverse candidates for presentation

3.3.2 Content Minimization Layer

Video Segmentation:

- ASR: Whisper (OpenAI) or AssemblyAI for transcription
- Scene detection: PySceneDetect or TransNetV2 for visual boundaries
- Semantic chunking: Combine ASR sentence boundaries + scene changes + silence detection
- Target: 30–180 second clips (hard maximum: 3 minutes)
- Timestamp alignment: Map chunks to video timecodes for clip extraction

PDF Section Detection:

- Parsing: PyMuPDF or Apache PDFBox for structured extraction
- Section headers: Regex + font size analysis to identify hierarchy
- Paragraph boundaries: Whitespace and formatting cues
- Target: 0.5–2 page segments (hard maximum: 3 pages)
- Extract images/figures with captions

Sufficiency Scoring:

- **Semantic Coverage:** $\text{Coverage}(R, Q) = \frac{\text{cosine}(\text{embed}(R),\text{embed}(Q))}{\text{duration}(R) \text{ or pages}(R)}$
- **Redundancy Penalty:** Penalize chunks with high overlap to already-selected content
- **Clarity Score:** LLM-based (few-shot) assessment of pedagogical clarity (0–10 scale)
- **Final Score:** $S = 0.5 \cdot \text{Coverage} + 0.3 \cdot \text{Clarity} - 0.2 \cdot \text{Redundancy}$
- Select top-1 to top-3 segments with highest sufficiency scores

Summarization to Micro-Nuggets:

- For longer segments (¿2 min or ¿1 page), provide extractive or abstractive summary
- Key points: 3–5 bullet points capturing main ideas
- Learner can choose: (1) summary + link to full segment, or (2) full segment directly

3.3.3 Pedagogical Layer

Question Generation:

- **Prompt-based (Milestone 3):** Few-shot examples aligned to Bloom taxonomy (Remember, Understand, Apply, Analyze, Evaluate, Create)
- **LoRA fine-tuned (Milestone 5):** Llama 3.1 8B with 500–2k exemplars from existing Q&A logs
- **Inputs:** Learning resource content, student query, desired Bloom level, prior performance

- **Outputs:** Question text, answer key, distractor options (for MCQ), rubric (for open-ended)
- **Validation:** Answerability check (can question be answered from resource?), factuality check (grounded in content?)

Rubric-Based Grading:

- **Rubric design:** 3–5 levels (Novice, Developing, Proficient, Advanced), explicit criteria per level
- **Grading prompt:** Chain-of-thought reasoning with rubric, reference answer, and student response
- **LoRA fine-tuning (optional):** On 1k–2k graded examples with expert annotations
- **Confidence scoring:** Model outputs confidence 0–1; defer to human if confidence < 0.7

Hints & Worked Examples:

- **Progressive hints:** Graduated scaffolding (conceptual hint $\rightarrow$ procedural hint $\rightarrow$ partial solution)
- **Worked examples:** Step-by-step solution to similar problem with annotations
- **Trigger:** Provide hints after 1–2 incorrect attempts or explicit student request

Distractor Generation (MCQ):

- **Quality criteria:** Plausible but incorrect, target common misconceptions, vary in difficulty
- **Generation:** Prompt or fine-tuned model with misconception database
- **Validation:** Expert review (10% sample), pilot testing with learners

3.3.4 Assessment & Analytics Layer

IRT (Item Response Theory) Ability Estimation:

- **Model:** 3PL (3-Parameter Logistic): $P(\theta, a, b, c) = c + \frac{1-c}{1+e^{-a(\theta-b)}}$
- θ: Learner ability, a: Item discrimination, b: Item difficulty, c: Guessing parameter
- **Estimation:** Maximum Likelihood Estimation (MLE) or Expected A Posteriori (EAP) for θ
- **Item calibration:** Marginal Maximum Likelihood (MML) using response data from all learners

Question Difficulty Calibration:

- **Initial estimate:** Weak labels from expert judgment (1–10 scale) or readability scores
- **Pilot phase:** Administer to diverse learners (n=50–200), collect response patterns
- **Calibration:** Run IRT parameter estimation (EM algorithm) on pilot data
- **Update cadence:** Recalibrate every 500 responses per item or quarterly

Learning Gain Measurement:

- **Primary metric:** $\Delta\theta = \theta_{\text{post}} - \theta_{\text{pre}}$ over study session or week
- **Normalized gain:** $g = \frac{\theta_{\text{post}} - \theta_{\text{pre}}}{\theta_{\text{max}} - \theta_{\text{pre}}}$ (Hake gain)
- **Mastery progression:** % of items at target proficiency level (e.g., $P(\theta) > 0.7$)
- **Longitudinal tracking:** Plot $\theta(t)$ over weeks/months, detect plateaus or regression

Distinguish Engagement from Learning:

- **Engagement metrics:** Time-on-task, click-through rate, completion rate (log but don't optimize for)
- **Learning metrics:** $\Delta\theta$, quiz accuracy uplift, downstream task performance
- **Correlation analysis:** Monitor correlation; high engagement + low $\Delta\theta$ signals ineffective content

3.3.5 Agentic Orchestration

Planner:

- **Input:** Student query, current θ estimate, prior interaction history, available tools
- **Output:** Execution plan: (1) parse query intent, (2) retrieve candidates, (3) select best resource, (4) deliver + formative assessment, (5) evaluate response, (6) plan next resource
- **Implementation:** ReAct-style prompting or LangGraph for multi-step orchestration

Tool Inventory:

- `retrieve_content(query, filters)`: Hybrid search + reranking
- `segment_resource(resource_id, target_length)`: Video/PDF chunker
- `score_sufficiency(segments, query)`: Sufficiency scorer
- `select_content(candidates, policy)`: Bandit policy or heuristic
- `generate_question(resource, bloom_level)`: Question generator
- `grade_response(question, response, rubric)`: Rubric grader
- `provide_hint(question, attempt_history)`: Hint generator
- `update_ability(learner_id, response_data)`: IRT updater
- `plan_next_resource(learner_state, goals)`: Next-action planner

Execution Flow:

1. Planner receives query, retrieves learner state (θ, history)
2. Calls `retrieve_content` $\rightarrow$ top-20 candidates

3. Calls `segment_resource` + `score_sufficiency` → top-3 segments
4. Calls `select_content` (bandit) → 1 segment
5. Delivers resource to learner
6. Calls `generate_question` → formative question
7. Learner responds
8. Calls `grade_response` → correctness + feedback
9. Calls `update_ability` → new θ estimate
10. Calls `plan_next_resource` based on $\Delta\theta$ and mastery gaps

3.3.6 Bandit Optimization Layer

Contextual Bandit Setup:

- **Context:** $x = [\theta, \text{query\_embedding}, \text{prior\_performance}, \text{resource\_metadata}]$
- **Actions:** $A = \{\text{segment}_1, \text{segment}_2, \ldots, \text{segment}_k\}$ (top-k from retrieval)
- **Reward:** $R = w_1 \cdot \Delta\theta + w_2 \cdot \text{brevity\_bonus} - w_3 \cdot \text{irrelevance\_penalty} - w_4 \cdot \text{latency\_cost}$
- **Policy:** $\pi(a|x)$ maps context to action (content selection)

Reward Components:

- $\Delta\theta$: Change in ability after learning session (proxy: quiz accuracy if θ not yet calibrated)
- Brevity bonus: $\max(0, \frac{T_{\text{target}} - T_{\text{actual}}}{T_{\text{target}}})$ where T is duration/length
- Irrelevance penalty: $1 - \text{cosine}(\text{resource}, \text{query})$ (semantic alignment)
- Latency cost: $\frac{\text{response\_time}}{\text{10 seconds}}$ (normalized)
- **Weights:** $w_1 = 1.0$, $w_2 = 0.3$, $w_3 = 0.5$, $w_4 = 0.1$ (tune via Pareto frontier analysis)

Algorithm:

- **Cold-start (Weeks 1–4):** Thompson Sampling with Beta priors, uniform exploration
- **Warm-up (Weeks 5–8):** LinUCB or Neural UCB with context features
- **Maturity (Weeks 9+):** Offline RL (DQN or BCQ) trained on logged data, online fine-tuning

Off-Policy Evaluation:

- Inverse Propensity Scoring (IPS): $\hat{V}(\pi) = \frac{1}{n}\sum_{i=1}^{n} \frac{\pi(a_i|x_i)}{\pi_0(a_i|x_i)} r_i$
- Doubly Robust (DR): Combines IPS with model-based estimates for lower variance
- Simulator: Replay historical interactions under new policy for what-if analysis

Safety Constraints:

- Minimum exploration rate: 10% (always try random actions to discover new content)
- Blacklist: Flag and exclude resources with negative feedback (thumbs-down, low $\Delta\theta$)
- Per-resource quotas: Limit exposure to any single resource to avoid overfitting policy
- Teacher override: Human-in-the-loop to veto policy decisions during pilot

3.4 Guardrails & Safety

Retrieval-Grounded Generation:

- All LLM outputs cite source chunks (IDs, timestamps)
- Factuality check: Verify claims against retrieved content using NLI model
- Contradiction detection: Flag if LLM output contradicts source material

Answerability Checks:

- Before generating questions, verify: "Can this question be answered from the provided resource?"
- Use LLM self-critique: "Given resource R, is question Q answerable? Yes/No + reasoning"
- If unanswerable, regenerate or defer

Refusal Paths:

- Detect out-of-scope queries (off-topic, harmful, non-educational)
- Polite refusal: "I can help with [list of topics]. Your question about [X] is outside my scope."
- Fallback to human tutor if available

JSON Schema Validation:

- All structured outputs (questions, rubrics, plans) validated against JSON schemas
- Reject malformed outputs, retry with schema instructions

Privacy & PII Protection:

- No learner PII (names, emails) in LLM prompts or logs
- Use anonymized IDs (learner_id = hash(email))
- Role-based access: Only educators/admins see identifiable data
- On-prem deployment option for institutions with strict privacy requirements

3.5 Telemetry & Monitoring

Per-User Dashboards:

- Learning trajectory: $\theta(t)$ over time with confidence intervals
- Mastery map: Heatmap of topics $\times$ proficiency levels
- Engagement metrics: Time-on-task, session frequency
- Resource exposure: Types and durations of consumed content

System-Level Metrics:

- Retrieval quality: nDCG@5, Recall@10, mean reciprocal rank (MRR)
- Minimality: Median resource length, overkill rate (% over target length)
- Question quality: Expert approval rate, factuality score
- Grading consistency: Inter-rater agreement (Krippendorff's α)
- IRT stability: θ standard error (SE), item parameter drift
- Bandit diagnostics: Exploration rate, policy entropy, regret bounds
- Safety incidents: Hallucination rate, refusal accuracy, complaint rate

Alerts:

- θ SE > 0.5 (low confidence in ability estimate)
- Item difficulty drift > 0.2 units/month (needs recalibration)
- Bandit policy entropy < 0.5 (exploitation too aggressive)
- Hallucination rate $> 5\%$ (content quality issue)
- P95 latency > 15 seconds (performance degradation)

Stakeholder Views:
Educator Dashboard:

- Class-level: Average θ, $\Delta\theta$, mastery rates per topic
- Individual learner drill-down: Trajectory, struggling topics, intervention recommendations
- Content analytics: Which resources are most/least effective (by $\Delta\theta$)
- Question bank review: Flag low-discrimination or misaligned items

Learner Explanations:

- "Why this resource?": "This 2-minute video segment covers [concept] at your current proficiency level."
- "Why this question?": "This question assesses [skill] at the Apply level of Bloom's taxonomy."
- "What's next?": "Based on your performance, I recommend: [next topic/skill] to build on [current mastery]."

4 Data Plan: Cold-Start to Flywheel

4.1 Challenge: No Historical Content Recommendation Labels

We have:

- Content corpus (videos, PDFs) with metadata (title, topic, duration/length)
- Historical Q&A logs (student questions, instructor answers)
- Historical assessment data (questions, responses, correctness)

We lack:

- Explicit labels: "For query Q, resource R is the best/shortest/most relevant"
- Implicit feedback: clicks, dwell time, learner ratings on recommended content

4.2 Cold-Start Strategy (Weeks 1–4)

4.2.1 Heuristic Baseline

Step 1: Metadata Filters

- Tag content with keywords/topics (manual or LLM-based auto-tagging)
- Extract query intent → match to content tags
- Filter: Duration $\leq$ 3 min (videos), Length $\leq$ 2 pages (PDFs)
- Rank by: (1) tag overlap, (2) brevity, (3) popularity (if view counts available)

Step 2: Semantic Coverage (Zero-Shot)

- Embed query and all candidate chunks
- Compute cosine similarity: $\text{sim}(q, c_i)$
- Coverage-per-unit: $\frac{\text{sim}(q,c_i)}{\text{duration}(c_i)}$ or $\frac{\text{sim}(q,c_i)}{\text{pages}(c_i)}$
- Select top-3 by coverage-per-unit

Step 3: ASR + Sectionizer

- Videos: ASR transcript → chunk by sentences/scenes → embed chunks → retrieve top-k
- PDFs: Extract sections → embed paragraphs → retrieve top-k
- Sufficiency check: LLM judges "Does this segment answer the query? Yes/No + confidence"
- If confidence < 0.7, try next candidate

Expected Performance:

- Retrieval Recall@10: 60–75% (decent but not great)
- Minimality: 70–80% under target length (heuristics enforce hard caps)
- User satisfaction: Baseline to improve upon

4.2.2 Weak Labels from Existing Data

Source 1: Q&A Logs

- Historical: Student asked Q, instructor provided answer A
- Heuristic: Find content chunks that match A (high similarity)
- Weak label: "For query Q, chunk C (matching A) is relevant"
- Caveat: Instructors may cite external resources not in our corpus
- Validation: Sample 100 cases, manually verify label quality

Source 2: Assessment Data

- Historical: Question q assesses concept c
- Weak label: "Content chunks explaining concept c are relevant for queries about c"
- Build concept → content mapping via topic modeling or LLM classification

Source 3: Content Transcripts + Question Overlap

- If video transcript or PDF text contains terms/phrases from historical questions, likely relevant
- TF-IDF or BM25 to find high-overlap chunks

Dataset Size: 5k–20k weakly labeled (query, relevant_chunk) pairs

4.2.3 Teacher-in-the-Loop Labeling

Active Learning Protocol:

1. System retrieves top-5 candidates for a query using heuristics
2. Present to expert educator: "For query Q, rank these 5 resources by relevance + minimality"
3. Expert provides: (1) ranking, (2) binary label (good/bad), (3) optional notes
4. Prioritize uncertain cases: Low confidence, low coverage, diverse query types

Labeling Sprint (Week 1–2):

- Goal: 500–1,000 high-quality labels
- Team: 2–3 educators, 2–4 hours/day
- Interface: Simple web form with query, candidates, ranking inputs
- Quality control: 10% overlap for inter-rater agreement (target Krippendorff's $\alpha > 0.7$)

Rubric for "Best Minimal Resource":

1. **Relevance:** Directly answers the query (5 = perfect, 1 = off-topic)
2. **Minimality:** Shortest duration/length that covers the question (5 = minimal, 1 = excessive)

3. **Clarity:** Pedagogically clear, well-explained (5 = excellent, 1 = confusing)
4. **Correctness:** Factually accurate (5 = correct, 1 = errors/misleading)

Inter-Rater Agreement:

- Measure: Krippendorff's α or Fleiss' κ on overlap subset
- Target: $\alpha > 0.7$ (acceptable), $\alpha > 0.8$ (good)
- If $\alpha < 0.7$: Refine rubric, provide calibration examples, re-train raters

4.3 Rapid Dataset Creation (Weeks 3–6)

4.3.1 Offline Labeling Campaign

Scale Up:

- Recruit 5–10 educators (internal or contract)
- Labeling platform: Label Studio, Prodigy, or custom React app
- Batch size: 50–100 queries per educator per week
- Target: 2,000–5,000 labels in 4 weeks

Sampling Strategy:

- Stratify by: (1) query complexity (simple fact → complex concept), (2) topic area, (3) resource type (video/PDF)
- Over-sample edge cases: Ambiguous queries, multiple plausible answers, very short/long resources
- Include negative examples: Clearly irrelevant resources (for classifier calibration)

Quality Assurance:

- Weekly calibration sessions: Discuss disagreements, update rubric
- Random audits: Senior educator reviews 10% of labels for correctness
- Automatic checks: Flag outliers (e.g., label contradicts semantic similarity)

4.3.2 Synthetic Data Augmentation

LLM-Generated Queries:

- For each content chunk, generate 3–5 plausible student questions
- Prompt: "Given this video segment, what questions would a student ask to learn this content?"
- Validation: Expert review 20% sample for realism and relevance
- Synthetic pairs: (generated_query, source_chunk) → 10k–50k pairs

Paraphrasing Existing Queries:

- Take historical Q&A queries, generate paraphrases via LLM or back-translation
- Preserves intent, increases query diversity
- Expands dataset by 2–3x

Caveat: Synthetic data has distribution shift risk. Use for augmentation, not replacement.

4.4 Using Existing Q&A/Assessment Logs

4.4.1 Question Generation Training Data

Source: Historical assessment database (10k–100k questions)

Preprocessing:

- Filter: Remove low-quality (typos, ambiguous), off-topic, or obsolete questions
- Annotate: Bloom level (manual or LLM-based classification)
- Map to content: Link question to content chunk(s) that explain the answer

Training Corpus:

- Format: (content_chunk, bloom_level, question, answer, distractors)
- Size: 1k–5k high-quality exemplars (post-filtering)
- Use for: Few-shot prompting (Milestone 3), LoRA fine-tuning (Milestone 5)

4.4.2 Grading Rubric Training Data

Source: Historical student responses + instructor grades/feedback

Preprocessing:

- Extract: (question, student_response, score, rubric_level, instructor_feedback)
- Standardize rubrics: Map to common scale (Novice/Developing/Proficient/Advanced)
- Clean: Remove responses with insufficient instructor feedback

Training Corpus:

- Format: (question, rubric, student_response, ground_truth_level, rationale)
- Size: 2k–10k graded responses
- Use for: Prompt engineering with exemplars, LoRA fine-tuning

4.4.3 Difficulty Estimation Calibration

Source: Historical response data (learner_id, question_id, correct/incorrect, response_time)
IRT Calibration:

- Run MML estimation (EM algorithm) on historical data
- Obtain: (a, b, c) parameters for each item, θ for each learner
- Validate: Holdout test set, check model fit (RMSE, AIC, BIC)

Cold-Start for New Items:

- Expert judgment: Rate difficulty 1–10, map to b estimate
- Pilot: Administer to 50–200 learners, update parameters
- Anchor items: Include calibrated items in each test to link scales

4.5 Data Flywheel (Weeks 7+)

4.5.1 Implicit Feedback Collection

User Actions:

- Click-through: Learner selects a recommended resource
- Dwell time: Duration spent on resource (proxy for relevance)
- Skip: Learner skips to next resource (negative signal)
- Thumbs up/down: Explicit feedback on resource quality
- Quiz performance: Correctness on formative questions (learning outcome proxy)

Logging:

- Event: (timestamp, learner_id, query, recommended_resources, selected_resource, dwell_time, quiz_score, feedback)
- Storage: Event stream (Kafka) → Data warehouse (Snowflake, BigQuery)
- Retention: 1 year minimum for analysis

4.5.2 Bandit Policy Training

Data Preparation:

- Context: $x = [\theta, \text{query\_embedding}, \text{resource\_metadata}]$
- Action: $a = \text{selected\_resource\_id}$
- Reward: $r = w_1 \cdot \text{quiz\_score\_uplift} + w_2 \cdot \text{brevity\_bonus} - w_3 \cdot \text{skip\_penalty}$
- Propensity: $\pi_0(a|x)$ from baseline heuristic policy

Training Cadence:

- Week 1–4: Collect data under heuristic policy (10k–50k interactions)
- Week 5: Train initial bandit (Thompson Sampling with historical data)
- Week 6–8: Deploy bandit, collect more data (exploration rate = 20%)
- Week 9+: Retrain weekly with cumulative data, reduce exploration to 10%

4.5.3 Retrieval Quality Improvement

Relevance Feedback:

- Positive: Learner clicks + high dwell time + thumbs-up → boost rank
- Negative: Skip + thumbs-down → demote rank
- Retraining: Fine-tune cross-encoder on (query, clicked_resource, label) pairs every month

Active Learning:

- Identify uncertain cases: Low cross-encoder score but high user engagement (or vice versa)
- Send to expert for labeling: "Is resource R relevant for query Q?"
- Retrain with new labels → improved reranker

4.5.4 Question Quality Refinement

Expert Review Loop:

- Weekly: Sample 50–100 generated questions
- Educators rate: Clarity, alignment, factuality (1–5 scale)
- Low-rated questions → analyze failure modes (ambiguous, off-topic, factually wrong)
- Update: Refine prompts or add to LoRA training set

Learner Feedback:

- Flag button: "This question is confusing/incorrect"
- Aggregate flags → prioritize for review
- Retire low-quality questions, replace with improved versions

4.6 Data Plan Summary

Phase	Data Sources	Methods	Timeline
Cold-Start (W1–4)	Content metadata, ASR, weak labels from Q&A logs	Heuristics, semantic similarity, teacher-in-the-loop (500–1k labels)	Weeks 1–4
Rapid Dataset (W3–6)	Offline labeling (2k–5k), synthetic queries (10k–50k)	Active learning, LLM augmentation, quality checks	Weeks 3–6
Existing Logs (W1–6)	Historical Q&A (10k–100k), assessments (10k–100k)	IRT calibration, question/rubric corpus creation	Weeks 1–6
Flywheel (W7+)	Implicit feedback (clicks, dwell, quiz scores), explicit feedback (thumbs)	Bandit training, retrieval fine-tuning, question refinement	Weeks 7+ (ongoing)

Table 2: Data Plan Timeline

5 Metrics & Evaluation

All metrics must be **executable per milestone** with concrete measurement protocols.

5.1 Retrieval & Selection Quality

5.1.1 nDCG@k (Normalized Discounted Cumulative Gain)

Definition:

$$\text{nDCG@k} = \frac{\text{DCG@k}}{\text{IDCG@k}}, \quad \text{DCG@k} = \sum_{i=1}^{k} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)}$$

Measurement:

- Expert labels: For 200–500 test queries, rate top-10 retrieved resources (relevance 0–3)
- Compute nDCG@5, nDCG@10
- **Baseline target:** nDCG@5 > 0.6, nDCG@10 > 0.65
- **Optimized target:** nDCG@5 > 0.75, nDCG@10 > 0.80

5.1.2 Recall@k

Definition: Fraction of relevant resources retrieved in top-k results.
Measurement:

- For each test query, identify all relevant resources in corpus (exhaustive or via expert judgment on random sample)
- Recall@k $= \frac{\#\text{ relevant in top-k}}{\text{total }\#\text{ relevant}}$
- **Baseline target:** Recall@10 > 0.70
- **Optimized target:** Recall@10 > 0.85

5.1.3 Coverage

Definition: Percentage of unique content chunks recommended across all queries.
Measurement:

- Log all recommended resources over 1 week
- Coverage $= \frac{\#\text{ unique chunks recommended}}{\text{total }\#\text{ chunks in corpus}}$
- **Target:** Coverage $> 50\%$ (avoid over-recommending popular content)

5.1.4 Time-to-First-Useful-Resource

Definition: Latency from query submission to first resource deemed useful by learner.
Measurement:

- Implicit: Time to first click + dwell time > 30 seconds
- Explicit: Thumbs-up on first resource
- **Target:** P50 < 5 seconds, P95 < 10 seconds

5.2 Minimality Metrics

5.2.1 Median Resource Length

Measurement:

- Videos: Median duration (seconds) of recommended segments
- PDFs: Median length (pages) of recommended segments
- **Target:** Videos < 90 seconds, PDFs < 1.5 pages

5.2.2 Overkill Rate

Definition: Percentage of recommendations exceeding target length thresholds.
Measurement:

- Set thresholds: Videos > 3 minutes, PDFs > 2 pages
- Overkill Rate $= \frac{\text{\# recommendations over threshold}}{\text{total \# recommendations}}$
- **Target:** Overkill Rate $< 15\%$

5.2.3 Compression Ratio

Definition: Ratio of recommended segment length to full resource length.
Measurement:

- For each recommended segment, compute $\frac{\text{segment length}}{\text{full resource length}}$
- Average across all recommendations
- **Target:** Compression ratio < 0.3 (segments are $< 30\%$ of full resource)

5.3 Question Quality Metrics

5.3.1 Expert Rubric Scores

Rubric Dimensions:

1. **Clarity:** Is the question unambiguous and well-phrased? (1–5)
2. **Alignment:** Does it assess the intended concept/skill? (1–5)
3. **Bloom Level:** Does it match the target cognitive level? (1–5)
4. **Factuality:** Is it grounded in provided content? (1–5)

Measurement:

- Sample 100–200 generated questions per milestone
- 2–3 experts rate each question independently
- Aggregate: Mean score per dimension, overall score (average of 4 dimensions)
- **Target:** Overall score > 4.0 (good), > 4.5 (excellent)

5.3.2 Pass@k on Canonical Answers

Definition: Percentage of generated questions for which the canonical answer (from reference material) receives a passing grade.

Measurement:

- Generate questions from content with known answers
- Grade canonical answer using rubric grader
- Pass@k = $\frac{\text{\# questions where canonical answer passes}}{\text{total \# questions}}$
- **Target:** Pass@1 $> 90\%$

5.3.3 Factuality via Reference-Grounded Checks

Method:

- NLI (Natural Language Inference) model: Check if question + answer entailed by source content
- Labels: Entailment (factual), Contradiction (hallucination), Neutral (unverifiable)
- **Target:** Entailment rate $> 95\%$, Contradiction rate $< 2\%$

Tools:

- Models: DeBERTa-v3-large-mnli, RoBERTa-large-mnli
- Thresholds: Entailment probability > 0.8 (confident), < 0.5 (reject question)

5.4 Assessment Quality Metrics

5.4.1 Item Discrimination (a)

Definition: Slope of the item characteristic curve; higher a means item better distinguishes high- from low-ability learners.

Measurement:

- Extract from IRT calibration: a parameter for each item
- **Target:** Median $a > 1.0$ (acceptable), > 1.5 (good), > 2.0 (excellent)
- **Flag:** Items with $a < 0.5$ (low discrimination) for review/retirement

5.4.2 Item Difficulty (b)

Definition: Ability level at which 50% probability of correct response.

Measurement:

- Extract from IRT: b parameter for each item
- **Target distribution:** $b \in [-2, 2]$ (covers range of abilities), roughly normal
- **Flag:** Items with $|b| > 3$ (extreme difficulty) for review

5.4.3 Guessing Parameter (c)

Definition: Lower asymptote of item characteristic curve; probability of correct response by random guessing.

Measurement:

- Extract from 3PL model: c parameter
- **Target:** For 4-option MCQ, $c \approx 0.20$–0.30 (slightly above chance)
- **Flag:** $c > 0.4$ (too easy to guess) or $c < 0.1$ (implausible)

5.4.4 Test Information and Ability Standard Error

Test Information: $I(\theta) = \sum_{i=1}^{n} I_i(\theta)$ where $I_i(\theta) = a_i^2 \cdot P_i(\theta) \cdot Q_i(\theta)/P_i'(\theta)^2$

Standard Error: $SE(\theta) = \frac{1}{\sqrt{I(\theta)}}$

Measurement:

- Compute for each learner after each test/session
- **Target:** $SE(\theta) < 0.4$ (acceptable precision), < 0.3 (good precision)
- Adaptive testing: Administer items near learner's θ to maximize information

5.4.5 Test-Retest Reliability

Definition: Correlation of θ estimates across two independent test administrations.

Measurement:

- Pilot: Administer parallel forms to 50–100 learners, 1 week apart
- Compute Pearson correlation: $\rho(\theta_{\text{test}}, \theta_{\text{retest}})$
- **Target:** $\rho > 0.80$ (acceptable), > 0.85 (good), > 0.90 (excellent)

5.5 Learning Outcome Metrics

5.5.1 $\Delta\theta$ Over Time

Definition: Change in ability estimate from session t to session $t + 1$.

Measurement:

- For each learner, compute $\Delta\theta_t = \theta_{t+1} - \theta_t$
- Aggregate: Mean $\Delta\theta$ across learners per week
- **Target:** Mean $\Delta\theta > 0.1$ per week (noticeable improvement)
- **Benchmark:** Compare to control group or historical baseline (if available)

5.5.2 Normalized Gain (Hake Gain)

Definition: $g = \frac{\theta_{\text{post}} - \theta_{\text{pre}}}{\theta_{\text{max}} - \theta_{\text{pre}}}$
Measurement:

- Pre-test: Assess θ_{pre} at start of course/unit
- Post-test: Assess θ_{post} at end
- θ_{max}: Theoretical maximum (e.g., 3 standard deviations above mean)
- **Interpretation:** $g < 0.3$ (low), $0.3 \leq g < 0.7$ (medium), $g \geq 0.7$ (high)
- **Target:** Mean $g > 0.4$ (medium gain)

5.5.3 Mastery Progression

Definition: Percentage of learners achieving target proficiency on each topic/skill.
Measurement:

- Define mastery threshold: $P(\text{correct}|\theta, b) > 0.70$ for items of target difficulty
- Compute: Mastery Rate $= \frac{\text{\# learners above threshold}}{\text{total \# learners}}$
- Track progression over time: Weekly mastery rate by topic
- **Target:** 70% mastery rate by end of unit

5.5.4 Downstream Task Performance

Definition: Performance on external assessments (e.g., final exams, standardized tests) after using the system.
Measurement:

- Compare: Users of adaptive system vs. control group (traditional instruction)
- Metrics: Mean score, pass rate, effect size (Cohen's d)
- **Target:** Effect size $d > 0.3$ (medium effect), ideally $d > 0.5$ (large effect)

5.6 Distinguishing Engagement from Learning

5.6.1 Engagement Metrics (Track but Do Not Optimize For)

- **Click-Through Rate (CTR):** % of recommended resources clicked
- **Dwell Time:** Average duration on resource (minutes)
- **Session Frequency:** # sessions per week
- **Completion Rate:** % of suggested activities completed

5.6.2 Learning Metrics (Primary Optimization Target)

- **$\Delta\theta$:** Ability growth per session/week
- **Quiz Accuracy Uplift:** Pre-resource vs. post-resource correctness
- **Mastery Rate:** % of topics at proficiency level
- **Downstream Performance:** External exam scores

5.6.3 Correlation Analysis

Method:

- Compute: $\rho(\text{engagement}, \Delta\theta)$ for each metric pair
- Scatter plots: Engagement (x-axis) vs. Learning (y-axis) per learner
- **Red flag:** High engagement + low $\Delta\theta$ (indicates ineffective content)
- **Green zone:** High engagement + high $\Delta\theta$ (effective and engaging)

Action:

- If $\rho < 0.3$: Engagement is decoupled from learning $\rightarrow$ audit content quality
- If negative correlation: Engaging content may be distracting $\rightarrow$ investigate

5.7 Safety & Accuracy Metrics

5.7.1 Hallucination Rate

Definition: Percentage of LLM outputs that contradict or are unsupported by retrieved content.
Measurement:

- Sample 200–500 generated questions/explanations per milestone
- NLI check: Entailment with source content
- Expert review: 10% random sample for factual errors
- Hallucination Rate $= \frac{\text{\# contradictions or unsupported claims}}{\text{total \# outputs}}$
- **Target:** Hallucination rate $< 3\%$

5.7.2 Refusal/Deferral Accuracy

Definition: Correctness of system's decision to refuse or defer to human when uncertain.
Measurement:

- Test set: 100 in-scope queries + 50 out-of-scope queries
- Metrics: Precision (true refusals / all refusals), Recall (true refusals / should-refuse cases)
- **Target:** Precision > 0.90, Recall > 0.85

5.8 Evaluation Protocol per Milestone

Milestone	Metrics to Evaluate	Acceptance Criteria
M1 (RAG Baseline)	nDCG@5, Recall@10, Median length, Overkill rate	nDCG@5 > 0.6, Recall@10 > 0.70, Median < 90s, Overkill $< 20\%$
M2 (Reranking + Segmentation)	nDCG@5, Compression ratio, Time-to-first	nDCG@5 > 0.70, Compression < 0.3, Time < 6s (P95)
M3 (Pedagogy v1)	Expert rubric score, Pass@1, Hallucination rate	Overall score > 4.0, Pass@1 $> 85\%$, Hallucination $< 5\%$
M4 (Bandits)	$\Delta\theta$, Normalized gain, Bandit regret	Mean $\Delta\theta > 0.08$/week, $g > 0.35$, Regret decreasing
M5 (LoRA FT)	Expert rubric score, Grading consistency, $\Delta\theta$	Overall score > 4.3, $\kappa > 0.85$, $\Delta\theta > 0.10$/week
M6 (Production)	All above + Safety metrics, P95 latency, Cost/learner	Hallucination $< 3\%$, Refusal precision > 0.90, Latency < 10s, Cost $<$ \$0.50/session

Table 3: Milestone Evaluation Criteria

6 Stepwise Roadmap (12 Weeks)

6.1 Milestone 1 (Weeks 1–2): RAG Baseline with Minimality Constraints

6.1.1 Objectives

- Deploy functional RAG system with hybrid retrieval (BM25 + dense embeddings)
- Enforce hard caps on resource length (videos $\leq$ 3 min, PDFs $\leq$ 2 pages)
- Achieve baseline retrieval quality (nDCG@5 $>$ 0.6, Recall@10 $>$ 0.70)
- Establish evaluation harness and red-team test cases

6.1.2 Tasks

1. **Content Ingestion:**
   - Parse videos (ASR via Whisper), PDFs (PyMuPDF)
   - Chunk by semantic boundaries (30–180s for video, paragraphs for PDF)
   - Extract metadata (title, topic, duration/length, keywords)
2. **Embedding & Indexing:**
   - Embed chunks using OpenAI `text-embedding-3-large`
   - Store in vector DB (Pinecone or Weaviate)
   - Build BM25 index (Elasticsearch or custom)
3. **Hybrid Retrieval:**
   - Implement RRF fusion (0.3 BM25 + 0.7 dense)
   - Retrieve top-50 from each, fuse to top-20
4. **Minimality Filtering:**
   - Hard filter: Remove chunks $>$ 3 min (video) or $>$ 2 pages (PDF)
   - Rank by: Semantic similarity / duration (or pages)
   - Return top-3 to user
5. **Evaluation Harness:**
   - Collect 200–300 test queries (diverse topics, complexities)
   - Expert labeling: Relevance ratings for top-10 results
   - Compute: nDCG@5, nDCG@10, Recall@10, median length, overkill rate
6. **Red-Team Test Cases:**
   - Adversarial queries: Ambiguous, out-of-scope, edge cases
   - Test refusal logic (placeholder: ”I don’t have information on that”)
   - Document failure modes

6.1.3 Deliverables

- Functional RAG API: `POST /retrieve` (query → top-3 resources)
- Evaluation report: Metrics table, example retrievals, failure analysis
- Jupyter notebook: Reproducible eval pipeline
- Data card: Content corpus stats (videos, PDFs, total duration/pages)

6.1.4 Acceptance Criteria

- nDCG@5 > 0.6, Recall@10 > 0.70
- Median resource length < 90 seconds (videos), < 1.5 pages (PDFs)
- Overkill rate $< 20\%$
- P95 latency < 8 seconds

6.2 Milestone 2 (Weeks 3–4): Cross-Encoder Reranking & Video/PDF Segmentation

6.2.1 Objectives

- Improve retrieval precision with cross-encoder reranker
- Implement video segmentation (ASR + scene detection) and PDF section extraction
- Generate JSON-structured outputs for downstream consumption
- Achieve nDCG@5 > 0.70, compression ratio < 0.3

6.2.2 Tasks

1. **Cross-Encoder Reranking:**
   - Deploy `ms-marco-MiniLM-L-12-v2` (self-hosted or HuggingFace Inference API)
   - Input: Top-20 candidates from M1 retrieval
   - Output: Reranked top-5 by relevance score
2. **Video Segmentation:**
   - ASR: Integrate Whisper API (or self-host Whisper-large-v3)
   - Scene detection: PySceneDetect on video files
   - Merge: Align ASR sentence boundaries with scene changes
   - Extract: 30–180s clips with timestamps
3. **PDF Section Detection:**
   - Parse: Identify headers (font size, formatting)
   - Segment: 0.5–2 page chunks preserving section context
   - Extract: Include images/figures with captions

4. **Sufficiency Scoring:**
   - For each segment, compute: $\frac{\text{similarity}(q,s)}{\text{duration/pages}(s)}$
   - LLM clarity check (optional few-shot prompt): "Rate pedagogical clarity 0–10"
   - Select top-1 segment with highest sufficiency score
5. **JSON Output Schema:**
   - Schema: {`query, resources: [`{`id, title, type, url, timestamp/page_range, duration/length, relevance_score, sufficiency_score`}`]`}
   - Validate: JSON schema validator (Pydantic or jsonschema)
6. **MMR Diversification:**
   - Apply MMR ($\lambda = 0.7$) to top-5 reranked results
   - Return top-3 diverse segments

6.2.3 Deliverables

- Enhanced API: `POST /retrieve_v2` with reranking + segmentation
- Video processing pipeline: ASR + scene detection + segmenter
- PDF processing pipeline: Section extractor
- Evaluation report: nDCG@5 (improved), compression ratio, segment examples
- Model card: Cross-encoder specs, performance, latency

6.2.4 Acceptance Criteria

- nDCG@5 > 0.70 (10% improvement over M1)
- Compression ratio < 0.3 (segments are < 30% of full resources)
- P95 latency < 6 seconds (including segmentation)
- No malformed JSON outputs (100% schema compliance)

6.3 Milestone 3 (Weeks 5–6): Pedagogy Tools v1

6.3.1 Objectives

- Deploy prompt-based question generator aligned to Bloom levels
- Implement rubric-based grader with chain-of-thought reasoning
- Add hint generator for scaffolding
- Achieve expert approval > 80% for generated questions

6.3.2 Tasks

1. **Question Generation:**
   - Prompt design: Few-shot examples (10–20 exemplars per Bloom level)
   - Input: Resource content + desired Bloom level (Remember/Understand/Apply/Analyze)
   - Output: {`question, answer_key, distractors (if MCQ), bloom_level`}
   - Answerability check: LLM self-critique ("Can this be answered from resource?")
2. **Rubric-Based Grading:**
   - Rubric template: 4 levels (Novice, Developing, Proficient, Advanced) with explicit criteria
   - Prompt: Chain-of-thought with rubric + reference answer + student response
   - Output: {`level, score, rationale, confidence`}
   - Confidence threshold: Defer to human if confidence < 0.7
3. **Hint Generation:**
   - Progressive hints: (1) Conceptual ("Think about X"), (2) Procedural ("Try Y approach"), (3) Partial solution
   - Trigger: After 1–2 incorrect attempts or explicit request
   - Output: {`hint_level, hint_text`}
4. **Worked Examples:**
   - Generate: Step-by-step solution to similar problem
   - Annotate: Explain each step's rationale
   - Deliver: After learner completes (or struggles with) question
5. **Validation:**
   - Sample 200 generated questions
   - Expert review: Clarity, alignment, Bloom level, factuality (1–5 scale per dimension)
   - Factuality check: NLI model (DeBERTa-mnli) for grounding

6.3.3 Deliverables

- Pedagogy API: `POST /generate_question`, `POST /grade_response`, `POST /get_hint`
- Prompt library: Versioned few-shot exemplars for each tool
- Evaluation report: Expert rubric scores, Pass@1, hallucination rate
- Example outputs: 50 question/grade/hint triplets with expert annotations

6.3.4 Acceptance Criteria

- Overall expert rubric score > 4.0 (out of 5)
- Pass@1 on canonical answers > 85%
- Hallucination rate < 5% (via NLI + expert review)
- Grading consistency: $\kappa > 0.75$ (agreement with human raters on 100 test cases)

6.4 Milestone 4 (Weeks 7–8): Bandit/RLHF for Content Selection

6.4.1 Objectives

- Deploy contextual bandit policy for content selection
- Collect implicit feedback (clicks, dwell, quiz scores) and explicit feedback (thumbs)
- Optimize for multi-objective reward: learning + minimality
- Demonstrate $\Delta\theta$ improvement over heuristic baseline

6.4.2 Tasks

1. **Bandit Infrastructure:**
   - Policy server: Thompson Sampling with Beta priors (initial)
   - Context: $x = [\theta, \text{query\_embedding}, \text{resource\_metadata}]$
   - Actions: Select from top-5 reranked candidates (from M2)
   - Exploration rate: 20% (uniform random)
2. **Reward Function:**
   - $R = 1.0 \cdot \Delta\theta + 0.3 \cdot \text{brevity\_bonus} - 0.5 \cdot \text{irrelevance\_penalty} - 0.1 \cdot \text{latency\_cost}$
   - $\Delta\theta$ proxy: Quiz correctness post-resource (short-term)
   - Brevity: $\max(0, \frac{180-\text{duration}}{180})$ for videos
   - Irrelevance: $1 - \text{cosine}(\text{resource}, \text{query})$
   - Latency: Normalized response time
3. **Logging & Feedback:**
   - Log: (timestamp, learner_id, query, context, action, reward, propensity)
   - Storage: Kafka → Data warehouse (BigQuery or Snowflake)
   - Volume target: 10k–50k interactions in Weeks 7–8
4. **Policy Training:**
   - Week 7: Collect data under heuristic policy (cold-start)
   - Week 8: Train bandit on logged data, deploy with exploration=20%
   - Off-policy eval: IPS/DR estimators to predict performance before deployment

5. **IRT Ability Tracking:**
   - Implement: EAP estimation for θ after each quiz
   - Calibrate: Initial item parameters from M3 question difficulty estimates
   - Update: Recalibrate parameters weekly using cumulative response data
6. **A/B Test:**
   - Control (50%): Heuristic policy from M2
   - Treatment (50%): Bandit policy
   - Metrics: $\Delta\theta$, normalized gain, minimality, user satisfaction
   - Duration: 2 weeks (Weeks 7–8)

6.4.3 Deliverables

- Bandit service: API for policy inference + logging
- IRT module: Ability estimation + item calibration
- A/B test report: Comparison of control vs. treatment on all metrics
- Decision memo: Proceed to LoRA fine-tuning (M5) if $\Delta\theta$ improves by $\geq 10\%$
- Telemetry dashboard: Real-time bandit diagnostics (exploration rate, reward trends, regret)

6.4.4 Acceptance Criteria

- Mean $\Delta\theta$ per week > 0.08 (treatment group)
- Normalized gain $g > 0.35$ (medium Hake gain)
- Bandit policy outperforms heuristic by $\geq 10\%$ on $\Delta\theta$
- Median resource length stable or reduced (< 90 seconds)
- Off-policy evaluation: Predicted reward matches actual reward within 10% (model calibration check)

6.5 Milestone 5 (Weeks 9–10): Optional LoRA Fine-Tuning & A/B Test

6.5.1 Objectives

- Fine-tune task-specific LoRA adapters for pedagogy tasks
- Compare fine-tuned vs. prompt-only versions via A/B test
- Achieve expert approval $> 85\%$ and grading consistency $\kappa > 0.85$
- Decision gate: Proceed only if fine-tuning shows measurable gains

6.5.2 Tasks

1. **Training Data Preparation:**
   - Question generation: 500–2k exemplars from M3 expert-reviewed + historical Q&A logs
   - Rubric grading: 1k–2k graded responses with rubric levels + rationale
   - Distractor generation: 300–500 MCQs with plausible distractors + quality ratings
   - Pedagogical style: 200–500 hint/explanation exemplars
2. **LoRA Fine-Tuning:**
   - Base model: Llama 3.1 70B or Mistral Large (or GPT-4o if API supports adapters)
   - LoRA config: $r = 16$, $\alpha = 32$, target modules: query/value projections
   - Training: 2–4 epochs, learning rate $1e - 4$, batch size 8 (gradient accumulation)
   - Infrastructure: 2x A100 GPUs, 4–12 hours per adapter
3. **Adapter Modules:**
   - Adapter 1: Question generator (input: content + Bloom level → output: question + answer)
   - Adapter 2: Rubric grader (input: question + rubric + response → output: level + rationale)
   - Adapter 3: Distractor generator (input: question + answer → output: 3 distractors)
   - Adapter 4: Pedagogical explainer (input: question + learner error → output: hint/explanation)
4. **A/B Test:**
   - Control (50%): Prompt-only pedagogy tools (M3)
   - Treatment (50%): LoRA-adapted pedagogy tools
   - Metrics: Expert rubric scores, grading consistency, $\Delta\theta$, learner feedback
   - Duration: 2 weeks (Weeks 9–10)
5. **Validation:**
   - Question quality: Sample 200 generated questions, expert review
   - Grading consistency: 100 responses graded by adapter vs. human raters, compute κ
   - Distractor quality: Pilot 50 MCQs with learners, measure distractor selection rates

6.5.3 Deliverables

- LoRA adapters: 4 trained adapters + model cards (architecture, training data, hyperparameters)
- Inference pipeline: vLLM or Lorax for efficient adapter serving
- A/B test report: Comparison of prompt-only vs. LoRA-adapted on all metrics
- Decision memo: Adopt LoRA if improvement $\geq 5\%$ on expert scores and $\Delta\theta$; otherwise, revert to prompt-only
- Migration plan: Procedure for upgrading base model while preserving adapters (transfer learning)

6.5.4 Acceptance Criteria

- Overall expert rubric score > 4.3 (5% improvement over M3)
- Grading consistency $\kappa > 0.85$ (10% improvement over M3)
- Mean $\Delta\theta$ per week > 0.10 (5% improvement over M4)
- Cost-benefit: Inference cost increase $< 20\%$ vs. prompt-only (due to adapter overhead)
- Decision: If criteria met, proceed to production with LoRA; if not, revert to prompt-only

6.6 Milestone 6 (Weeks 11–12): Production Hardening

6.6.1 Objectives

- Implement comprehensive guardrails (hallucination detection, refusal logic, safety filters)
- Deploy bias checks and compliance audits (GDPR, FERPA, accessibility)
- Set up telemetry dashboards for monitoring and alerting
- Achieve production-ready SLAs: hallucination $< 3\%$, P95 latency < 10s, cost $< \$0.50$/session

6.6.2 Tasks

1. **Hallucination Detection:**
   - Retrieval-grounded verification: All LLM outputs cite source chunks
   - NLI-based fact-checking: DeBERTa-mnli checks entailment with sources
   - Contradiction detector: Flag if output contradicts source material
   - Threshold: Reject outputs with entailment probability < 0.8
2. **Refusal Logic:**
   - Intent classifier: Detect out-of-scope queries (off-topic, harmful, non-educational)
   - Confidence threshold: Refuse if retrieval confidence < 0.5 or ambiguous query
   - Polite refusal templates: "I can help with [topics]. Your question about [X] is outside my scope."
   - Escalation: Option to connect with human tutor if available
3. **Safety Filters:**
   - Content moderation: OpenAI Moderation API or Perspective API for toxic content
   - PII redaction: Detect and anonymize names, emails, phone numbers in logs
   - Bias checks: Audit question generation for demographic bias (gender, race, age stereotypes)
4. **Compliance:**
   - GDPR: Right to erasure (delete learner data on request), data minimization, consent flows

- FERPA: Student data privacy (no sharing without consent), role-based access controls
- Accessibility: WCAG 2.1 AA compliance (screen reader support, keyboard navigation, alt text)

5. **Telemetry Dashboards:**
   - Metrics: Retrieval nDCG, minimality, question quality, $\Delta\theta$, hallucination rate, P95 latency, cost/session
   - Alerts: θ SE > 0.5, item drift > 0.2, hallucination $> 5\%$, latency > 15s
   - Tools: Grafana + Prometheus (or Datadog, New Relic)
6. **Educator Dashboard:**
   - Class-level: Average θ, $\Delta\theta$, mastery rates, struggling learners
   - Individual drill-down: Trajectory, topic proficiency, intervention recommendations
   - Content analytics: Most/least effective resources (by $\Delta\theta$)
7. **Learner Explanations:**
   - ”Why this resource?”: Explain selection rationale (relevance, length, difficulty match)
   - ”Why this question?”: Explain pedagogical intent (skill assessed, Bloom level)
   - ”What’s next?”: Suggest next topic/skill based on current mastery
8. **Load Testing:**
   - Simulate: 10k concurrent users, 50k requests/hour
   - Metrics: P95/P99 latency, error rate, resource utilization
   - Auto-scaling: Configure Kubernetes HPA or cloud auto-scaling policies
9. **Disaster Recovery:**
   - Backups: Daily snapshots of vector DB, data warehouse
   - Failover: Multi-region deployment (primary + backup)
   - Incident response: Runbooks for common failures (API outage, DB corruption, hallucination spike)

6.6.3 Deliverables

- Production deployment: Fully instrumented system with guardrails + monitoring
- Compliance report: GDPR/FERPA audit + accessibility checklist
- Runbooks: Incident response procedures for common issues
- Telemetry dashboard: Public URL for stakeholders (view-only access)
- User documentation: Educator guide + learner guide (how to use system, interpret explanations)
- Launch readiness review: Sign-off from security, legal, and product teams

6.6.4 Acceptance Criteria

- Hallucination rate < 3% (500 sample outputs audited)
- Refusal precision > 0.90, recall > 0.85 (100 test cases)
- P95 latency < 10 seconds (under 10k concurrent users)
- Cost per session < $0.50 (including all APIs, compute, storage)
- Compliance: 100% pass on GDPR/FERPA checklist
- Accessibility: WCAG 2.1 AA compliance (automated + manual audit)
- Uptime: 99.5% availability (4 hours downtime/month acceptable for pilot)

6.7 Roadmap Summary Table

Milestone	Duration	Key Tasks	Deliverables	Acceptance Criteria
M1	Weeks 1–2	RAG baseline, minimality constraints, eval harness, red-team	API, eval report, notebook, data card	nDCG@5 > 0.6, Recall@10 > 0.70, Median < 90s
M2	Weeks 3–4	Cross-encoder reranking, video/PDF segmentation, JSON outputs, MMR	Enhanced API, video/PDF pipelines, model card	nDCG@5 > 0.70, Compression < 0.3, Latency < 6s
M3	Weeks 5–6	Question generation, rubric grading, hints, answerability checks	Pedagogy APIs, prompt library, eval report	Expert score > 4.0, Pass@1 > 85%, Hallucination < 5%
M4	Weeks 7–8	Contextual bandits, IRT tracking, reward function, A/B test	Bandit service, IRT module, A/B report, telemetry	$\Delta\theta > 0.08$/week, $g > 0.35$, 10% improvement
M5	Weeks 9–10	LoRA fine-tuning (4 adapters), A/B test, validation	Adapters, inference pipeline, A/B report, migration plan	Expert score > 4.3, $\kappa > 0.85$, $\Delta\theta > 0.10$/week
M6	Weeks 11–12	Guardrails, safety, compliance, telemetry, educator/learner dashboards	Production system, compliance report, runbooks, docs	Hallucination < 3%, Latency < 10s, Cost < $0.50/session

Table 4: 12-Week Roadmap Summary

7 Reinforcement Learning Design

7.1 Problem Formulation

Objective: Learn a policy $\pi(a|x)$ that selects content (action a) given context (x) to maximize long-term learning outcomes (reward R) under minimality and accuracy constraints.

Why RL/Bandits:

- **Exploration-exploitation tradeoff:** Discover which resources work best for different learners while exploiting known good content.
- **Delayed rewards:** Learning happens over multiple sessions; need credit assignment.
- **Personalization:** Different learners have different θ, preferences, learning styles.
- **Distribution shift:** Content corpus evolves, learner population changes over time.

7.2 Multi-Objective Reward Function

Reward Definition:

$$R = w_1 \cdot \Delta\theta + w_2 \cdot B(d) - w_3 \cdot I(q, r) - w_4 \cdot L(t)$$

Component 1: Learning Gain ($\Delta\theta$)

- **Primary signal:** Change in IRT ability estimate after interaction.
- **Computation:** $\Delta\theta = \theta_{\text{post}} - \theta_{\text{pre}}$
- **Proxy (short-term):** Quiz correctness post-resource: $\frac{\text{\# correct}}{\text{total \# questions}}$
- **Ground truth (long-term):** $\Delta\theta$ over 1 week from pre/post-tests.
- **Normalization:** Scale to [0, 1], typical range $\Delta\theta \in [0, 0.5]$ per session.
- **Weight:** $w_1 = 1.0$ (highest priority).

Component 2: Minimality Bonus ($B(d)$)

- **Intent:** Reward shorter resources that still enable learning.
- **Computation:** $B(d) = \max\left(0, \frac{T_{\text{target}} - d}{T_{\text{target}}}\right)$
- Where $T_{\text{target}} = 90$ seconds (videos) or 1 page (PDFs), $d =$ actual duration/length.
- **Range:** $B \in [0, 1]$, where $B = 1$ if $d = 0$ (shortest), $B = 0$ if $d \geq T_{\text{target}}$.
- **Weight:** $w_2 = 0.3$ (encourage brevity but don't sacrifice learning).

Component 3: Irrelevance Penalty ($I(q, r)$)

- **Intent:** Penalize resources with low semantic alignment to query.
- **Computation:** $I(q, r) = 1 - \text{cosine}(\text{embed}(q), \text{embed}(r))$
- **Range:** $I \in [0, 1]$, where $I = 0$ (perfectly aligned), $I = 1$ (orthogonal).

- **Weight:** $w_3 = 0.5$ (penalize off-topic content).

Component 4: Latency Cost ($L(t)$)

- **Intent:** Penalize slow responses (UX penalty).
- **Computation:** $L(t) = \frac{t}{10 \text{ seconds}}$ where t = response time.
- **Range:** $L \in [0, \infty)$, typical range [0.3, 1.5].
- **Weight:** $w_4 = 0.1$ (minor penalty, prioritize learning over speed).

Weight Tuning:

- **Initial weights:** $w_1 = 1.0$, $w_2 = 0.3$, $w_3 = 0.5$, $w_4 = 0.1$
- **Tuning method:** Pareto frontier analysis (multi-objective optimization)
- Vary weights, plot tradeoff curves: $\Delta\theta$ vs. minimality, learning vs. relevance
- **Stakeholder input:** Educators prioritize learning (w_1 high), product team balances UX (w_2, w_4)

7.3 Contextual Bandit Approach

7.3.1 Algorithm Choice

Phase 1 (Weeks 1–4): Thompson Sampling

- **Model:** Beta-Bernoulli bandit (for binary rewards) or Gaussian bandit (for continuous rewards)
- **Prior:** Beta(1, 1) or Gaussian($\mu = 0$, $\sigma^2 = 1$) for each action
- **Update:** Posterior update after each interaction (Bayesian inference)
- **Selection:** Sample from posterior, select action with highest sampled reward
- **Exploration:** Automatic via posterior sampling (high uncertainty $\rightarrow$ more exploration)

Phase 2 (Weeks 5–8): LinUCB (Linear Upper Confidence Bound)

- **Model:** Assume reward is linear in context features: $R(x, a) = x^T \theta_a + \epsilon$
- **Features:** $x = [\theta_{\text{learner}}, \text{query\_emb}, \text{resource\_meta}]$, dimension $d \approx 50$–100
- **Update:** Ridge regression update for θ_a after each interaction
- **Selection:** $a^* = \arg\max_a \left(x^T \hat{\theta}_a + \alpha \sqrt{x^T A_a^{-1} x} \right)$
- Where A_a is the design matrix for action a, α is exploration parameter (typically 0.1–1.0)
- **Advantage:** Fast convergence, interpretable, proven regret bounds ($O(\sqrt{dT \log T})$)

Phase 3 (Weeks 9+): Neural Bandit (Optional)

- **Model:** Neural network to predict $R(x, a)$ from context-action pairs

- **Architecture:** 2–3 layer MLP, 128–256 hidden units
- **Exploration:** Ensemble disagreement or dropout-based uncertainty
- **Update:** Mini-batch gradient descent on logged data every day/week
- **Advantage:** Captures nonlinear relationships, but needs more data and compute

7.3.2 Off-Policy Evaluation

Challenge: Can't A/B test every policy change; need to evaluate new policies offline.

Method 1: Inverse Propensity Scoring (IPS)

- **Estimator:** $\hat{V}(\pi) = \frac{1}{n}\sum_{i=1}^{n} \frac{\pi(a_i|x_i)}{\pi_0(a_i|x_i)} r_i$
- Where π_0 is the logging policy (historical), π is the new policy to evaluate.
- **Advantage:** Unbiased under correct propensity estimates.
- **Disadvantage:** High variance if propensities are small (divide by near-zero).

Method 2: Doubly Robust (DR)

- **Estimator:** $\hat{V}(\pi) = \frac{1}{n}\sum_{i=1}^{n} \left[\frac{\pi(a_i|x_i)}{\pi_0(a_i|x_i)}(r_i - \hat{r}(x_i, a_i)) + \sum_a \pi(a|x_i)\hat{r}(x_i, a)\right]$
- Where $\hat{r}(x, a)$ is a learned reward model (e.g., from LinUCB or neural net).
- **Advantage:** Lower variance than IPS, still unbiased if either propensity or reward model is correct.
- **Use case:** Preferred for pre-deployment evaluation.

Method 3: Simulator

- **Approach:** Replay historical logged data under new policy.
- For each historical interaction (x_i, a_i, r_i), check if new policy would select a_i.
- If yes, count r_i toward cumulative reward; if no, skip.
- **Caveat:** Underestimates true reward (only counts on-policy data), but useful for what-if analysis.

7.3.3 Safety Constraints

Challenge: Prevent policy from over-optimizing reward at expense of safety (e.g., always selecting shortest resource even if it doesn't teach).

Constraint 1: Minimum Exploration

- **Rule:** With probability $\epsilon = 0.10$, select uniformly random action (ignore policy).
- **Rationale:** Ensures continued discovery of new content, prevents premature convergence.

Constraint 2: Blacklist

- **Rule:** If resource receives consistent negative feedback (thumbs-down $> 30\%$ or $\Delta\theta < 0$), exclude from action space for 1 week.
- **Rationale:** Protects learners from known-bad content.

Constraint 3: Per-Resource Quotas

- **Rule:** Limit each resource to $< 10\%$ of total recommendations per week.
- **Rationale:** Prevents policy from overfitting to a few popular resources, maintains diversity.

Constraint 4: Teacher Override

- **Rule:** During pilot (Weeks 7–10), educators can flag policy decisions for review.
- **Process:** Flagged cases go to weekly review meeting, update blacklist or adjust reward weights if needed.

7.4 Escalation to Full RL

When to Escalate:

- **Bandit plateau:** If LinUCB performance stops improving after 50k+ interactions.
- **Long-horizon rewards:** If learning gains manifest over multiple sessions (need credit assignment).
- **Sequential decision-making:** If we want to optimize entire learning trajectory (not just single resource).

Full RL Approach:

- **MDP Formulation:**
  - State: $s = [\theta, \text{mastery\_map}, \text{recent\_history}, \text{current\_query}]$
  - Action: $a = \text{resource\_id}$
  - Reward: Same multi-objective reward as above
  - Transition: Deterministic (state updates based on learner response)
- **Algorithm:** Offline RL (BCQ, CQL, IQL) trained on logged data.
- **Advantage:** Can optimize long-term learning trajectory (e.g., skill prerequisites, spaced repetition).
- **Requirements:** $\geq$ 100k logged interactions, simulator for policy testing, safety constraints (same as bandits).

Decision Gate (Week 12):

- **Escalate to RL if:** Bandit shows $< 5\%$ improvement after 50k interactions AND we have sufficient logged data.
- **Stay with bandits if:** Bandit continues improving OR data volume insufficient ($<$ 50k interactions).

7.5 Practical Implementation Checklist

1. **Data collection:** Log (timestamp, learner_id, context, action, reward, propensity) for every interaction.
2. **Feature engineering:** Encode context (θ, query_emb, resource_meta) as fixed-dimension vector.
3. **Cold-start:** Week 1–4 use uniform random or heuristic policy to collect initial data.
4. **Policy training:** Week 5–8 train LinUCB on logged data, deploy with $\epsilon = 0.20$ exploration.
5. **Off-policy eval:** Before each deployment, run IPS/DR to estimate new policy value.
6. **A/B test:** Deploy new policy to 50% of users, compare against control (heuristic or previous policy).
7. **Monitoring:** Track exploration rate, reward trends, regret bounds, safety violations.
8. **Retraining cadence:** Retrain policy weekly (early) $\rightarrow$ monthly (mature) with cumulative data.
9. **Safety checks:** Review blacklist, check for diversity violations, educator feedback loop.
10. **Escalation trigger:** If bandit plateaus, consider offline RL (requires 100k+ interactions + simulator).

8 Fine-Tuning Policy

8.1 General Principle: Avoid Premature Fine-Tuning

Rationale:

- Prompt engineering + RAG is fast, flexible, and works well for most tasks.
- Fine-tuning locks us into specific model versions, requires retraining for upgrades.
- Foundation models improve rapidly; fine-tuning today may be obsolete in 6 months.
- LoRA/PEFT provides middle ground: task-specific adaptation with upgrade agility.

Decision Rule: Fine-tune ONLY if:

1. Prompt engineering has plateaued (no improvement after multiple prompt iterations).
2. We have $\geq$ 500–2k high-quality labeled examples for the task.
3. Task-specific fine-tuning shows $\geq$ 5% improvement on key metrics in offline eval.
4. Inference cost/latency is acceptable (fine-tuned models can be faster/cheaper if self-hosted).

8.2 When to Fine-Tune

8.2.1 Task 1: Question Generation

Entry Criteria:

- Prompt-only expert approval $<$ 80% (quality plateau).
- We have $\geq$ 500–2k exemplars: (content, Bloom_level, question, answer, distractors).
- Sources: Historical Q&A logs + M3 expert-reviewed questions + synthetic augmentation.

Fine-Tuning Approach:

- Base model: Llama 3.1 8B or Mistral 7B (smaller models sufficient for this task).
- LoRA config: $r = 16$, $\alpha = 32$, target modules: `q_proj`, `v_proj`.
- Training: 2–4 epochs, learning rate $1e-4$, batch size 8.
- Validation: Hold out 20% of exemplars, track perplexity + expert approval on validation set.

Expected Improvement:

- Expert approval: 80% (prompt-only) $\rightarrow$ 88–92% (LoRA).
- Consistency: Reduced variance in question quality across topics.
- Inference: Slightly faster (no need for long few-shot examples in prompt).

8.2.2 Task 2: Rubric-Based Grading

Entry Criteria:

- Grading consistency (Krippendorff's α) < 0.80 with prompt-only.
- We have $\geq$ 1k–2k graded responses: (question, rubric, student_response, level, rationale).
- Sources: Historical graded assignments + M3 expert-labeled test cases.

Fine-Tuning Approach:

- Base model: Llama 3.1 13B or GPT-4o (if API supports adapters).
- LoRA config: $r = 16$, $\alpha = 32$, target modules: `q_proj`, `v_proj`.
- Training: 3–5 epochs, learning rate $5e - 5$, batch size 4.
- Validation: Inter-rater agreement (α) on holdout set.

Expected Improvement:

- Grading consistency: 0.75–0.80 (prompt-only) $\rightarrow$ 0.85–0.90 (LoRA).
- Rationale quality: More detailed, pedagogically grounded feedback.
- Confidence calibration: Better alignment between confidence scores and actual correctness.

8.2.3 Task 3: Distractor Generation (MCQ)

Entry Criteria:

- Distractor quality: $< 70\%$ rated as plausible by educators.
- We have $\geq$ 300–500 MCQs with high-quality distractors + quality ratings.
- Sources: Historical assessment database + expert-curated MCQs.

Fine-Tuning Approach:

- Base model: Llama 3.1 8B or Mistral 7B.
- LoRA config: $r = 8$, $\alpha = 16$ (smaller adapter for simpler task).
- Training: 2–3 epochs, learning rate $1e - 4$, batch size 8.
- Validation: Distractor plausibility (expert ratings), selection rates (pilot with learners).

Expected Improvement:

- Plausibility: 65–70% (prompt-only) $\rightarrow$ 80–85% (LoRA).
- Selection rates: More uniform distribution (good distractors attract learners with misconceptions).

8.2.4 Task 4: Pedagogical Explainer (Hints, Worked Examples)

Entry Criteria:

- Hint quality: $< 75\%$ rated as helpful by learners.
- We have $\geq$ 200–500 exemplars: (question, learner_error, hint, worked_example).
- Sources: Expert-written hints + scaffolding templates.

Fine-Tuning Approach:

- Base model: Llama 3.1 13B (larger model for nuanced pedagogy).
- LoRA config: $r = 16$, $\alpha = 32$, target modules: `q_proj`, `v_proj`.
- Training: 2–4 epochs, learning rate $5e-5$, batch size 4.
- Validation: Learner feedback (helpfulness ratings), $\Delta\theta$ after hint.

Expected Improvement:

- Helpfulness: 72–75% (prompt-only) → 85–88% (LoRA).
- Scaffolding quality: More tailored to learner's specific error pattern.

8.3 LoRA/PEFT Configuration

Advantages of LoRA:

- **Parameter efficiency:** Train only 0.1–1% of model parameters (adapters).
- **Upgrade agility:** When base model updates, retrain adapters (much faster than full FT).
- **Multi-task serving:** Load different adapters for different tasks (vLLM, Lorax).
- **Lower cost:** Training takes hours (not days) on 1–2 GPUs.

Typical LoRA Hyperparameters:

- **Rank (r):** 8–16 for simple tasks, 16–32 for complex tasks.
- **Alpha (α):** $2r$ (scaling factor for adapter weights).
- **Target modules:** `q_proj`, `v_proj` (query and value projections in attention layers).
- **Dropout:** 0.05–0.1 (regularization to prevent overfitting).

Training Infrastructure:

- **GPUs:** 1–2x A100 (40GB or 80GB) for 7B–13B models, 2–4x A100 for 70B models.
- **Framework:** HuggingFace PEFT, Unsloth, or Axolotl.
- **Training time:** 4–12 hours per adapter (depending on dataset size and model size).
- **Cost:** \$50–\$200 per training run on cloud (AWS, GCP, Azure).

Inference Infrastructure:

- **Serving:** vLLM (with LoRA support) or Lorax (multi-adapter serving).
- **Adapter swapping:** Load different adapters for different tasks (e.g., question gen vs. grading).
- **Latency:** Adapter adds < 50ms vs. base model inference.

8.4 Migration Plan for Model Upgrades

Challenge: Base models upgrade frequently (e.g., GPT-4o → GPT-4.5, Llama 3.1 → Llama 4). How to preserve adapter investment?

Strategy 1: Adapter Transfer Learning

1. When new base model released, evaluate prompt-only performance on validation set.
2. If new base model $> 5\%$ better, consider migration.
3. Retrain adapters on new base model (faster than training from scratch, can reuse datasets).
4. A/B test: Old base + adapters vs. New base + retrained adapters.
5. Migrate if new base outperforms old by $\geq 3\%$ on key metrics.

Strategy 2: Adapter Ensembles (Optional)

- Run both old and new base models in parallel for 1–2 weeks.
- Route queries to each model 50/50, compare outputs.
- Gradually increase traffic to new model if performance is better.
- Decommission old model once new model proves stable.

Strategy 3: Gradual Rollout

- Week 1: 10% of traffic to new model (canary deployment).
- Week 2: 25% of traffic if no regressions.
- Week 3: 50% of traffic.
- Week 4: 100% of traffic, decommission old model.

Versioning:

- Track: (base_model_version, adapter_version, training_date, performance_metrics).
- Rollback plan: If new model underperforms, revert to previous version within 1 hour.

8.5 Decision Checklist for Fine-Tuning

Criterion	Threshold	Assessment Method
Prompt-only performance plateau	$< 80\%$ on key metric	Offline eval on validation set
High-quality labeled data available	$\geq$ 500–2k exemplars	Data audit, inter-rater agreement check
Expected improvement	$\geq 5\%$ on key metric	Offline eval of fine-tuned model
Inference cost acceptable	$< 20\%$ increase vs. prompt-only	Benchmarking (tokens/sec, latency)
Upgrade agility preserved	Can retrain in < 1 week	LoRA/PEFT architecture (not full FT)
Team capacity	Can manage training pipeline	ML engineering availability

Table 5: Fine-Tuning Decision Checklist

Decision Rule:

- If ALL criteria met: Proceed with LoRA fine-tuning.
- If ANY criterion fails: Stick with prompt engineering, revisit in 3 months.

9 Risks & Mitigations

9.1 Risk 1: Cold-Start for Content Recommendations

Description: No historical labels for "best resource" for each query. Initial recommendations may be poor quality.

Impact:

- Low user satisfaction in first 2–4 weeks.
- Learners may abandon system if initial experience is bad.
- Data collection is slow (need real interactions to train bandit).

Mitigation:

1. **Heuristic baseline (M1):** Use metadata filters + semantic similarity to deliver "good enough" recommendations day-one.
2. **Weak labels (M1):** Bootstrap with Q&A logs and assessment data to create initial training set (5k–20k pairs).
3. **Teacher-in-the-loop (M1–M2):** Rapid labeling sprint (500–1k high-quality labels in 2 weeks).
4. **Active learning (M2+):** Prioritize uncertain cases for expert review, improve dataset quality iteratively.
5. **Bandit exploration (M4):** 20% exploration rate ensures system discovers good content even with poor initial policy.
6. **Fallback:** If retrieval confidence < 0.5, offer to connect learner with human tutor.

Monitoring:

- Track: nDCG@5, user satisfaction (thumbs-up rate), session abandonment rate.
- Alert: If nDCG < 0.5 or satisfaction $< 60\%$ after Week 2, escalate to manual review.

9.2 Risk 2: Over-Long Resources (Minimality Failure)

Description: Retrieval returns hour-long videos or 50-page PDFs instead of short segments.

Impact:

- Poor user experience (cognitive overload).
- Learners skip resources, don't engage with content.
- Violates core product requirement (micro-learning).

Mitigation:

1. **Hard caps (M1):** Filter out any resource > 3 min (video) or > 2 pages (PDF).
2. **Sufficiency scoring (M2):** Rank by coverage-per-unit-time/pages, prefer shorter segments.

3. **Segmentation (M2):** Break videos/PDFs into chunks before retrieval (30–180s clips, paragraph-level segments).
4. **Bandit reward (M4):** Brevity bonus in reward function directly penalizes long resources.
5. **Post-retrieval summarization:** If segment still too long, provide extractive summary + link to full content.

Monitoring:

- Track: Median resource length, overkill rate (% over threshold), compression ratio.
- Alert: If median > 90s or overkill $> 15\%$, audit retrieval pipeline for failures.

9.3 Risk 3: Hallucinations (Factual Errors)

Description: LLM generates incorrect questions, explanations, or hints not grounded in content.
Impact:

- Misleading learners (teach wrong information).
- Erosion of trust (educators flag system as unreliable).
- Potential harm (e.g., incorrect medical/legal information).

Mitigation:

1. **Retrieval-grounded generation:** All outputs cite source chunks (M1+).
2. **Answerability check (M3):** Before generating question, verify "Can this be answered from resource?"
3. **Factuality verification (M3):** NLI model checks entailment with source content, reject if contradiction.
4. **Confidence thresholding (M3):** If model confidence < 0.7, defer to human or flag for review.
5. **Refusal paths (M6):** If query is ambiguous or outside scope, politely refuse rather than hallucinate.
6. **Expert review loop (M3+):** Sample 50–100 outputs weekly, educators rate factuality, update prompts/adapters.

Monitoring:

- Track: Hallucination rate (via NLI + expert audit), refusal rate, learner flags ("this is wrong").
- Alert: If hallucination $> 5\%$, pause deployment and investigate root cause.

9.4 Risk 4: Misaligned Difficulty (Questions Too Easy/Hard)

Description: Generated questions don't match learner's ability level (θ), leading to frustration or boredom.

Impact:

- Low engagement (learners skip questions).
- Poor learning outcomes (questions don't challenge appropriately).
- Inaccurate θ estimates (if all questions are too easy/hard).

Mitigation:

1. **IRT calibration (M4):** Estimate item difficulty (b) and discrimination (a) from pilot data.
2. **Adaptive questioning (M4+):** Select questions near learner's θ to maximize information.
3. **Anchor items (M4+):** Include pre-calibrated items in each test to link scales and detect drift.
4. **Recalibration cadence (M4+):** Re-estimate item parameters every 500 responses or quarterly.
5. **Drift detection (M4+):** Monitor $|b_{\text{current}} - b_{\text{initial}}| > 0.2$, flag for review.
6. **Weak difficulty estimates (M3):** Use expert judgment (1–10 scale) as initial b estimates before pilot.

Monitoring:

- Track: Item parameters (a, b, c), θ standard error (SE), test information curves.
- Alert: If SE > 0.5 (low precision) or item $a < 0.5$ (low discrimination), flag for recalibration.

9.5 Risk 5: Privacy & PII Leakage

Description: Learner PII (names, emails, performance data) leaked in logs, prompts, or outputs.

Impact:

- GDPR/FERPA violations (legal liability, fines).
- Loss of user trust, institutional adoption barriers.
- Reputational damage for Pearson.

Mitigation:

1. **Anonymization (M1):** Use hashed IDs (`learner_id = hash(email)`) in all logs and prompts.
2. **Data minimization (M1):** Don't log unnecessary PII (names, DOB, addresses).
3. **Role-based access (M6):** Only educators/admins see identifiable data, learners see only their own.
4. **PII redaction (M6):** Scan logs for names/emails/phone numbers, redact before storage.

5. **Encryption (M6):** Encrypt data at rest (database, logs) and in transit (TLS).
6. **Right to erasure (M6):** Implement GDPR compliance (delete learner data on request within 30 days).
7. **On-prem deployment option:** For institutions with strict privacy requirements, offer self-hosted version.

Monitoring:

- Track: PII detection alerts (automated scans), access logs (who viewed what data), data retention policies.
- Alert: If PII detected in logs or unauthorized access, escalate to security team immediately.

9.6 Risk 6: Model Drift & Degradation

Description: Foundation models update (GPT-4o → GPT-4.5), prompts break, adapters become misaligned.

Impact:

- Sudden performance drop (questions become nonsensical, retrieval degrades).
- User complaints, loss of trust.
- Emergency rollback required.

Mitigation:

1. **Prompt versioning (M1+):** Store prompts in git repo, tag with model version, rollback if needed.
2. **Model pinning (M1+):** Use specific model versions (e.g., `gpt-4o-2024-08-01`) not `gpt-4o` (which auto-updates).
3. **Regression testing (M3+):** Before upgrading model, run evaluation harness on 200–500 test cases, compare to baseline.
4. **Adapter retraining (M5):** When new base model released, retrain LoRA adapters (takes 1–2 days).
5. **Gradual rollout (M6):** Canary deployment (10% traffic) for 1 week, full rollout only if metrics stable.
6. **Fallback (M6):** If new model underperforms, rollback to previous version within 1 hour.

Monitoring:

- Track: Model version in use, key metrics (nDCG, expert scores, $\Delta\theta$) per model version.
- Alert: If metrics drop $> 5\%$ after model upgrade, trigger rollback procedure.

9.7 Risk 7: Bias & Fairness

Description: System exhibits bias (demographic, topic, language) in recommendations or questions.

Impact:

- Unequal learning outcomes across demographic groups.
- Reinforcement of stereotypes (e.g., gender bias in examples).
- Legal/ethical concerns, institutional backlash.

Mitigation:

1. **Bias audit (M3, M6):** Sample 200–500 outputs, check for demographic bias (gender, race, age stereotypes).
2. **Diverse training data:** Ensure exemplars span diverse topics, perspectives, cultural contexts.
3. **Counterfactual testing:** Swap demographic attributes in prompts (e.g., "he" → "she"), check for output changes.
4. **Fairness metrics (M4+):** Compute $\Delta\theta$ by demographic group, flag if disparity > 0.1 (effect size).
5. **Content review:** Educators review questions/resources for stereotypes, update prompts/adapters if issues found.

Monitoring:

- Track: Bias metrics (demographic parity, equalized odds), content review flags.
- Alert: If bias detected (counterfactual test fails or disparity > 0.1), escalate to ethics review.

9.8 Risk Summary Table

Risk	Impact	Mitigation	Monitoring
Cold-start	High	Heuristics, weak labels, teacher-in-the-loop, exploration	nDCG, satisfaction, abandonment
Over-long resources	High	Hard caps, sufficiency scoring, segmentation, brevity reward	Median length, overkill rate
Hallucinations	High	Grounded generation, answerability checks, NLI verification	Hallucination rate, learner flags
Misaligned difficulty	Medium	IRT calibration, adaptive questioning, drift detection	SE, item parameters, information
Privacy/PII	High	Anonymization, encryption, RBAC, right to erasure	PII detection, access logs
Model drift	Medium	Versioning, pinning, regression testing, gradual rollout	Metrics per model version
Bias & fairness	Medium	Bias audit, diverse data, counterfactual testing, fairness metrics	Demographic disparity, content flags

Table 6: Risk Summary

10 Deliverables per Milestone

10.1 Milestone 1: RAG Baseline

Code & Pipelines:

- Jupyter notebook: `01_rag_baseline.ipynb` (reproduces evaluation)
- Python modules: `retrieval.py`, `embedding.py`, `fusion.py`
- API: `POST /retrieve` endpoint with OpenAPI spec
- Docker image: Containerized RAG service for deployment

Data & Models:

- Data card: Content corpus stats (videos/PDFs, total duration/pages, metadata schema)
- Embedding model: OpenAI `text-embedding-3-large` or self-hosted `bge-large-en-v1.5`
- BM25 index: Elasticsearch or custom index on corpus
- Vector DB: Pinecone or Weaviate collection with 100k–1M embeddings

Evaluation & Reports:

- Evaluation report: nDCG@5, Recall@10, median length, overkill rate (tables + charts)
- Test set: 200–300 queries with relevance labels (expert-annotated)
- Red-team cases: 50 adversarial/edge-case queries + failure analysis
- Decision memo: Go/no-go for M2 based on acceptance criteria

10.2 Milestone 2: Reranking & Segmentation

Code & Pipelines:

- Video processing: `video_segmenter.py` (ASR + scene detection + timestamp extraction)
- PDF processing: `pdf_sectionizer.py` (header detection + paragraph chunking)
- Cross-encoder reranker: `reranker.py` (HuggingFace model wrapper)
- API: `POST /retrieve_v2` with JSON schema for structured outputs

Data & Models:

- Model card: Cross-encoder (`ms-marco-MiniLM-L-12-v2`) specs, latency, performance
- ASR outputs: Whisper transcripts for video corpus (stored in DB)
- PDF segments: Section-level chunks with page ranges (stored in DB)

Evaluation & Reports:

- Evaluation report: nDCG@5 (improved), compression ratio, segment examples
- Benchmark: Latency comparison (M1 vs. M2), cost per query
- Ablation study: BM25-only vs. Dense-only vs. Hybrid + reranking

10.3 Milestone 3: Pedagogy Tools v1

Code & Pipelines:

- Question generator: `question_gen.py` with few-shot prompts
- Rubric grader: `grader.py` with chain-of-thought prompts
- Hint generator: `hint_gen.py` with progressive scaffolding
- APIs: `POST /generate_question`, `POST /grade_response`, `POST /get_hint`

Data & Models:

- Prompt library: Few-shot exemplars for each Bloom level (10–20 per level)
- Rubric templates: 4-level rubrics (Novice/Developing/Proficient/Advanced) for common tasks
- Validation dataset: 200 generated questions with expert ratings

Evaluation & Reports:

- Evaluation report: Expert rubric scores (per dimension), Pass@1, hallucination rate
- Example outputs: 50 question/grade/hint triplets with expert annotations
- Failure analysis: Common error modes (ambiguous questions, off-topic, hallucinations)

10.4 Milestone 4: Bandit Optimization

Code & Pipelines:

- Bandit service: `bandit_policy.py` (Thompson Sampling or LinUCB)
- IRT module: `irt_estimator.py` (ability estimation, item calibration)
- Logging pipeline: Kafka producer + consumer, data warehouse schema
- A/B test framework: Randomization, metric computation, statistical tests

Data & Models:

- Logged data: 10k–50k interactions (context, action, reward, propensity)
- IRT parameters: (a, b, c) for each item, θ for each learner
- Bandit policy: Serialized model (Thompson Sampling parameters or LinUCB weights)

Evaluation & Reports:

- A/B test report: Control vs. treatment comparison on $\Delta\theta$, gain, minimality, satisfaction
- Telemetry dashboard: Real-time bandit diagnostics (exploration rate, reward trends, regret)
- Decision memo: Proceed to LoRA fine-tuning if $\Delta\theta$ improves by $\geq 10\%$

10.5 Milestone 5: LoRA Fine-Tuning

Code & Pipelines:

- Training scripts: `train_lora.py` for each adapter (question gen, grader, distractor, explainer)
- Inference pipeline: vLLM or Lorax adapter serving
- A/B test: Prompt-only vs. LoRA-adapted comparison

Data & Models:

- Training datasets: 500–2k exemplars per task (deduplicated, high-quality)
- LoRA adapters: 4 trained adapters (.safetensors files) + config files
- Model cards: Per adapter (architecture, training data, hyperparameters, performance)

Evaluation & Reports:

- A/B test report: Prompt-only vs. LoRA-adapted on expert scores, consistency, $\Delta\theta$
- Validation curves: Training/validation loss, perplexity over epochs
- Decision memo: Adopt LoRA if improvement $\geq 5\%$; otherwise revert to prompt-only
- Migration plan: Procedure for upgrading base model while preserving adapters

10.6 Milestone 6: Production Hardening

Code & Pipelines:

- Guardrails: `hallucination_detector.py`, `refusal_handler.py`, `pii_redactor.py`
- Monitoring: Grafana dashboards, Prometheus metrics, alert rules
- Compliance: GDPR/FERPA audit scripts, right-to-erasure handler
- Load testing: Locust or K6 scripts for 10k concurrent users

Data & Models:

- NLI model: DeBERTa-mnli for factuality checks
- Moderation API: OpenAI Moderation or Perspective API integration
- Telemetry schemas: Metrics definitions, alert thresholds

Evaluation & Reports:

- Compliance report: GDPR/FERPA checklist + accessibility audit (WCAG 2.1 AA)
- Load test report: P95/P99 latency, error rate, resource utilization under 10k users
- Runbooks: Incident response procedures for hallucination spikes, API outages, PII leaks
- Launch readiness review: Sign-off from security, legal, product teams

Dashboards & Documentation:

- Educator dashboard: Class-level and individual learner analytics
- Learner explanations: "Why this?" UI components with rationale
- User docs: Educator guide (100 pages) + learner guide (20 pages)
- Technical docs: API reference, architecture diagrams, deployment guide

10.7 Deliverable Checklist (All Milestones)

Category	Deliverables
Code & Pipelines	Notebooks, modules, APIs, Docker images, training scripts, inference pipelines
Data & Models	Datasets, embeddings, adapters, IRT parameters, prompt libraries, schemas
Evaluation & Reports	Metric tables, charts, A/B test results, failure analyses, decision memos
Documentation	Model cards, data cards, runbooks, user guides, API docs, architecture diagrams
Dashboards & Monitoring	Grafana/Prometheus, educator/learner dashboards, alert rules
Compliance & Security	GDPR/FERPA audits, PII redaction, encryption, incident response

Table 7: Deliverable Checklist

11 First 14 Days: Executable Task List

11.1 Objective

Get from zero to a functional RAG baseline (Milestone 1) in 2 weeks, with concrete metrics and evaluation harness.

11.2 Team Composition (Small Team)

- **1 ML Engineer:** Responsible for RAG pipeline, embeddings, retrieval.
- **1 Data Scientist:** Responsible for evaluation, metrics, analysis.
- **1 Content Engineer:** Responsible for content ingestion, ASR, parsing.
- **1 Product Manager (part-time):** Coordinate with educators, define test cases.

11.3 Day-by-Day Task Breakdown

11.3.1 Week 1 (Days 1–7): Content Ingestion & Baseline Retrieval

Day 1–2: Environment Setup & Content Audit

1. Set up cloud environment (AWS/GCP/Azure), provision GPUs (optional for local embedding).
2. Set up vector DB (Pinecone free tier or Weaviate Docker) and Elasticsearch.
3. Audit content corpus: Count videos, PDFs, total duration/pages, identify metadata fields (title, topic, keywords).
4. **Output:** Environment ready, content inventory spreadsheet.

Day 3–4: ASR & PDF Parsing

1. Videos: Run Whisper ASR on 100–500 sample videos (batch job, 1–2 hours per 10 hours of video).
2. PDFs: Parse with PyMuPDF, extract text, identify section headers, chunk by paragraphs.
3. Store: ASR transcripts and PDF text in database (Postgres or MongoDB).
4. **Output:** 100–500 videos transcribed, 500–1k PDFs parsed.

Day 5–6: Embedding & Indexing

1. Chunk content: 30–180s for videos (sentence boundaries), paragraphs for PDFs (100–500 tokens).
2. Embed chunks: OpenAI `text-embedding-3-large` (or self-hosted `bge-large-en-v1.5`).
3. Index: Upload embeddings to vector DB, build BM25 index on chunk text.
4. Metadata: Store title, topic, duration/length, chunk_id, parent_resource_id.
5. **Output:** 10k–100k chunks indexed in vector DB + BM25.

Day 7: Hybrid Retrieval Implementation

1. Implement: BM25 retrieval (top-50), dense retrieval (top-50), RRF fusion (0.3 BM25 + 0.7 dense).
2. Test: 10 sample queries, inspect top-10 results manually.
3. Filter: Apply hard caps (videos $\leq$ 3 min, PDFs $\leq$ 2 pages).
4. **Output:** Functional `/retrieve` API endpoint.

11.3.2 Week 2 (Days 8–14): Evaluation & Baseline Metrics

Day 8–9: Test Set Creation

1. Collect 200–300 test queries: Sample from historical Q&A logs, generate synthetic queries (LLM), recruit educators to write queries.
2. Stratify: Simple (factual) vs. complex (conceptual), short (1 word) vs. long (full sentence).
3. **Output:** Test set CSV: (query_id, query_text, topic, complexity).

Day 10–11: Expert Labeling

1. Run retrieval on 200–300 test queries, get top-10 results per query.
2. Recruit 2–3 educators, each labels 100 queries (overlap 20% for inter-rater agreement).
3. Labeling UI: Simple web form with query, top-10 results, relevance rating (0–3: not/somewhat/relevant/highly relevant).
4. **Output:** Labeled dataset: (query_id, resource_id, relevance_score).

Day 12: Evaluation Harness

1. Implement metrics: nDCG@5, nDCG@10, Recall@10, median length, overkill rate.
2. Jupyter notebook: Load test set, run retrieval, compute metrics, generate report (tables + plots).
3. **Output:** `01_eval_baseline.ipynb` with metric results.

Day 13: Red-Team Testing

1. Adversarial queries: Ambiguous ("What is X?"), out-of-scope ("How to hack?"), edge cases (typos, jargon).
2. Test refusal logic: "I don't have information on that" for out-of-scope.
3. Document: Failure modes (retrieval returns irrelevant, over-long, or no results).
4. **Output:** Red-team report with 50 test cases + failure analysis.

Day 14: Report & Decision Memo

1. Compile: Evaluation report (metrics table, example retrievals, failure analysis).
2. Decision memo: Go/no-go for M2 based on acceptance criteria (nDCG@5 $>$ 0.6, Recall@10 $>$ 0.70).
3. Presentation: 15-minute review with stakeholders (product, engineering, educators).
4. **Output:** PDF report + slides for review meeting.

11.4 Task Assignment Table

Day	Task	Owner	Deliverable
1–2	Environment setup, content audit	ML Engineer	Inventory spreadsheet
3–4	ASR & PDF parsing	Content Engineer	Transcripts & parsed PDFs
5–6	Embedding & indexing	ML Engineer	Vector DB + BM25 index
7	Hybrid retrieval implementation	ML Engineer	`/retrieve` API
8–9	Test set creation	Data Scientist + PM	Test set CSV
10–11	Expert labeling	PM + Educators	Labeled dataset
12	Evaluation harness	Data Scientist	Eval notebook
13	Red-team testing	All team	Red-team report
14	Report & decision memo	Data Scientist + PM	PDF report + slides

Table 8: Task Assignment (Days 1–14)

11.5 Daily Standup Agenda

1. Yesterday: What did I complete?
2. Today: What am I working on?
3. Blockers: Any dependencies or issues?
4. Metrics: Current progress toward Day 14 acceptance criteria.

11.6 Success Criteria (End of Day 14)

- **Functional API:** `POST /retrieve` returns top-3 resources in < 8s (P95).
- **Metrics:** nDCG@5 > 0.6, Recall@10 > 0.70, median length < 90s (videos) / < 1.5 pages (PDFs).
- **Evaluation harness:** Reproducible notebook with automated metric computation.
- **Red-team:** 50 adversarial cases documented with failure modes.
- **Decision:** Go/no-go for M2 based on acceptance criteria.

11.7 Tools & Resources (Quick Start)

- **ASR:** OpenAI Whisper API or `faster-whisper` (self-hosted)
- **PDF parsing:** PyMuPDF (`fitz`) or Apache PDFBox
- **Embeddings:** OpenAI `text-embedding-3-large` or HuggingFace `sentence-transformers`
- **Vector DB:** Pinecone (free tier) or Weaviate (Docker)
- **BM25:** Elasticsearch (Docker) or `rank_bm25` (Python library)

- **Notebooks:** Jupyter or Google Colab
- **API:** FastAPI or Flask
- **Labeling:** Label Studio or Google Forms

Estimated Cost (Days 1–14):

- OpenAI embeddings: $50–$100 (100k chunks)
- Whisper ASR: $50–$200 (100–500 videos)
- Vector DB: Free (Pinecone tier) or $50/month (Weaviate cloud)
- Compute: $100–$200 (cloud GPUs/CPUs for 2 weeks)
- **Total:** $250–$650 for first 14 days