01 · FLAGSHIP · PRODUCTION · CTO

Nabgh نابغ

An AI teaching assistant that turns 249 official textbooks into grounded lesson plans, quizzes and worksheets in each school department's exact print format.

ROLE CTO & Lead AI Engineer — technical owner since Aug 2026

STACK Python · BGE embedding & reranker models · ChromaDB · BM25/RRF · GPT models · Gemini models · Node.js · MongoDB · AWS EC2/S3 · React Native · React

400+	249	55K	6,147	100%	99.6%
teachers onboarded	textbooks · ~28,500 pages	chunks in 2 hybrid indexes	grounded lesson plans	recall@5 on golden set	quiz items proven printable

THE PROBLEM

Teachers in Kuwait write lesson plans, short quizzes and worksheets every week. Each subject department expects a strict paper format and the content has to follow the official textbook page by page. Generic LLMs fail three ways: they invent facts that aren't in the book, ignore the format, and break right-to-left Arabic print layout.

WHAT I BUILT

- **Arabic RAG pipeline.** A deterministic lesson resolver maps a school lesson to the exact book pages, then hybrid retrieval (BGE dense embeddings + BM25, fused with RRF) scoped by grade, subject, term and lesson, then cross-encoder reranking.
- **Guardrails in code.** A pre-generation similarity gate refuses before the model is called; every citation must be a retrieved chunk; every quoted span must exist in the cited chunk; exam questions must be answerable from specific pages.
- **Verification chain.** Worksheets are drafted by a fast GPT model, blind-solved and reviewed by a stronger GPT model, then checked by deterministic rules (no ambiguous items, key ↔ blank alignment, page fill ≥ 95%).
- **The product around it.** Provider-agnostic AI gateway (keys moved off the mobile app), PDF + Word renderer with RTL validators, feature flags, kill switches, staging → production with immutable releases and one-line rollback.

PIPELINE

1. Teacher app
2. Node API · auth · flags · cache
3. Lesson → exact book pages
4. Hybrid retrieve (dense + BM25 → RRF)
5. Rerank + refusal gate
6. Generate → JSON schema
7. Citation & quote gates
8. Paper planner → PDF / Word

EVIDENCE

Golden set (56 questions)	recall@5 100% · recall@1 89.6–97.9% · MRR 0.95–0.99 · over-refusal 0–2.1%
Printability proof (production)	4,340 / 4,358 quiz items printed alone (99.6%); failures hidden from the picker
Independent auditor agent	Worksheet pass rate 11 → 28 → 31 → 44 of 44; final round 0 high/medium defects
Automated tests	1,894 test functions on the RAG service · 102 test files on the API
Cost	~$0.009 per RAG generation · ~$0.014 per verified worksheet · stored plans served at ~$0 in ~1 ms

HARD PROBLEMS SOLVED

Unreadable Arabic textbooks

Official PDFs store ligatures in reverse and add diacritic noise, so retrieval returned garbage. I wrote a repairer that rebuilds lines from glyph bounding boxes at near-OCR quality for $0, with vision OCR only as fallback. Turning Gemini's reasoning off cut OCR from ~$0.02 to $0.0007–0.0025 per page.

"These questions look like they come from the internet"

Teacher feedback showed model-written questions drifting from the book. I moved grounding from the prompt into threshold, citation and quote checks, with tracked printed-page offsets.

Correct items, wrong paper

Valid questions failed together: one printed another's answer. I added a leak detector, rule-driven paper planners per department, and printed every one of 4,358 items alone on production.

REAL OUTPUT (SAMPLE FILES, NAMES BLANK)



Grounded lesson plan (geology, grade 11)



Short quiz in a department's format (physics, grade 10)



Grade-1 worksheet with vision-verified pictures