■ Roga Coaching Engine — Implementation Spec (MVP)

Purpose: Upgrade raw LLM responses into Roga-quality coaching feedback that is consistent, educational, and brand-aligned.

1. Coaching Engine Overview

Transform raw API outputs into structured coaching feedback anchored in the QI taxonomy, evaluated with Rubric v1.1, and always structured as Why it works → Improvement → Pro Tip → Example upgrade.

2. JSON Schema (v1.0)

```
{ "schema": "roga.feedback.v1", "scenario_id": 7, "user_question": "string",
"character_reply": "string", "coach_feedback": { "score_1to5": 4,
"qi_skills": ["clarifying", "probing"], "why_it_works": "string",
"improvement": "string", "pro_tip": "string", "example_upgrade": "string" },
"meta": { "brand_check": true, "length_ok": true, "banned_content": [],
"hash": "sha256..." } }
```

3. CoachFilter Pipeline

Step	Description
0 — Guardrails	Brand style: supportive, clever, concise. ≤80 words. No phones in school. Always name ≥1 QI skill.
1 — Normalize	Ensure schema compliance; auto-fix missing fields.
2 — Classify QI Skills	Tag 1–3 skills from taxonomy.
3 — Judge	Evaluate feedback block with Rubric v1.1. Thresholds: Clarity ≥4, QI Alignment ≥4, Brand ≥5.
4 — Critique & Revise	If thresholds fail, self-critique + revise. Max 2 iterations.
5 — Style Check	≤80 words. Structure: Why it works, Improvement, Pro Tip, Example upgrade.
6 — De-duplication	Optional: vector DB to avoid repeating pro tips.
7 — Emit	Return final JSON to frontend.

4. API Endpoint

```
@router.post("/coach") def coach(payload: CoachIn): final = { "schema":
"roga.feedback.v1", "scenario_id": payload.scenario_id, "user_question":
payload.user_question, "character_reply": payload.character_reply,
"coach_feedback": { "score_1to5": 4, "qi_skills":
["clarifying","criteria-setting"], "why_it_works": "You targeted the unclear
part, reducing confusion fast.", "improvement": "Name the first deliverable
to make action obvious.", "pro_tip": "Anchor on criteria for success before
steps.", "example_upgrade": "What’s the first deliverable due and how will
it be graded?" } } return final
```

5. Prompt Library

QI Skill Classifier, Judge (Rubric v1.1), and Rewrite prompts should be versioned in codebase.

6. Deployment Notes

• Insert CoachFilter middleware. • Enforce schema + thresholds server-side. • Store feedback + metadata. • Optional vector DB for freshness.

7. Next Steps

• Build `/coach` endpoint in FastAPI. • Update Next.js frontend to call `/coach`. • Seed Pro Tip Library. • Track analytics to refine coaching quality.