HiPat QA Test PRD — Personality-Governed AMA + Roles/Skills

1) Goal

Verify that one active Personality ("Pat") governs all responses and routing for AMA and roles/skills.
Prove that new roles/skills can be added, routed, and audited without code changes.
Detect defects in routing, prompts, RLS, logs, and UI CRUD.

2) System Under Test (SUT) Summary

- Personality Layer: 1 active profile injected into every response.
- Agents (Roles): AMA + any added roles.
- Skills: macro_lookup, food_search, barcode_parse, workout_parse, rag_lookup.
- Router: trigger-based selection of agent/skills with priorities.
- Admin UI: Personality, Roles, Skills, Prompts, Routes, Agent↔Skill, Testbench, Audit.
- Storage: Supabase tables defined in the prior PRD.

3) Assumptions

- Supabase project is reachable.
- Seed data exists or will be detectable as "missing" by tests.
- External APIs may be disabled; skills must still return stubs or structured errors.
- Auth user for tests has "authenticated" role.

4) Non-Functional Targets

- P50 response under 2.0 s without external API calls.
- Logs present for every assistant reply (route, skills fired, model, tokens, cost).
- No PII leaks in logs.
- RLS prevents anonymous read/write on privileged tables.

5) Core Test Oracles (what "correct" looks like)

- Every reply is preceded by Personality injection (tone: first-person "I", spartan).
- AMA is default when no route matches.
- "i ate …" triggers food_search; "macros …" triggers macro_lookup.
- Skills return JSON matching schema in §12.
- Admin edits persist and are reflected in behavior without redeploy.
- All Admin changes create `audit_events` rows.

6) Environment Checks (read-only; pass/fail)

1. Tables exist:
   `personality_profiles, agents, skills, agent_skills, prompts, routes, sessions, messages, audit_events` → PASS if all present.
2. Active personality: exactly one `is_active=true` → PASS if count=1.
3. AMA role present → PASS if role=AskMeAnything exists.
4. Seed skills present → PASS if listed keys exist.
5. Two baseline routes exist: "i ate", "macros" → PASS if found.
6. RLS enabled on all tables → PASS if RLS flag=true for each.
7. Anonymous cannot select from `audit_events` → PASS if denied.

7) Personality Injection Suite

Input: "What is HiPat?"
Expect: Reply in first-person "I", concise, Jarvis-like tone, no emojis, no filler.
Fail if: Third-person, verbose, missing tone, or plant-based suggestions appear unprompted.

8) Routing Suite

A) Trigger: "I ate 3 eggs and ½ cup oatmeal."
Expect: Router hits **food_search** first. AMA stays governing.
Logs: `routes:[food_search]`, `agent:AskMeAnything`, `skills_fired:["food_search",…]`.
Fail if: AMA answers directly with no skill call.

B) Trigger: “Tell me the macros for a Big Mac.”
Expect: **macro_lookup** fired; if not in cache, **food_search** fallback allowed.
Fail if: Free-text answer with no structured skill output.

C) No-match: “Explain why ribeye is tender.”
Expect: AMA only; no skill fired.
Fail if: Any skill fires.

D) Ambiguity: “I ate 10 oz ribeye and what are the macros?”
Expect: Both skills allowed; order obeys priorities.
Fail if: Only one fires when both are enabled and routable.

9) Skill Behavior Suite

All skills must return JSON per §12 or a structured error with `error.code`, `error.message`.

A) `macro_lookup`
Inputs:

- “10 oz ribeye”
- “10oz ribeye”
- “10-oz ribeye”
- “284 g ribeye”
  Expect: Same food normalized; macros present; source indicated; confidence ≥ 0.7 if from USDA/CA; unit parsing robust.
  Fail if: Zeros, missing fiber, or unit parsing error.

B) `food_search`
Inputs:

- “Big Mac”
- “McNuggets 10 piece”
- “barcode: 060410015406” (example pattern)
  Expect: Brand/fast-food results; when external APIs disabled, return stub with `source:"stub"` and `needs_external:true`.
  Fail if: 404/empty with no structured error.

C) `barcode_parse`
Input: “barcode: 062600400037”
Expect: EAN normalized; either product match or `not_found:true`.
Fail if: Free-text only.

D) `workout_parse`
Input: “Bench 5×5 @ 185, RPE 8. Then 3×10 rows 60s rest.”
Expect: JSON with lifts, sets, reps, load, rest, RPE.
Fail if: Free-text.

E) `rag_lookup`
Input: “What is Pat’s FREE framework?”
Expect: Source list and excerpt fields; no hallucinated citations.

10) Admin UI CRUD Suite

A) Personality

- Read active. Toggle off, toggle on.
  Expect: Exactly one active at any time. Toggle enforces exclusivity. Audit row created.

B) Prompts

- Edit AMA prompt title content v+1.
  Expect: New version created; runtime uses newest version.

C) Roles

- Create role “NutritionCoach”. Disable/enable.
  Expect: Persisted; visible in router target list.

D) Skills

- Create skill “glycemic_lookup”.
  Expect: Visible; cannot fire until attached and routed.

E) Agent↔Skill

- Attach `glycemic_lookup` to "NutritionCoach" with priority=10.
  Expect: Ordering reflected in runtime; logs show priority order.

F) Routes

- Add trigger "glycemic" → `glycemic_lookup`.
  Expect: Simulation shows target. Disabled state respected.

G) Audit

- Every above change creates `audit_events` entry with `area`, `action`, `payload`.

11) Logging & Telemetry Suite

- `messages.meta` includes: `personality_id`, `agent_id`, `routes_hit[]`, `skills_fired[]`, `model`, `tokens`, `cost_cents`.
- Tokens and cost are non-null for assistant replies.
- Errors include stack or function name in `meta.error.context`.

12) Contract Schemas (must match exactly)

12.1 `macro_lookup` result

```
{ "items": [ { "name": "Ribeye steak, cooked", "serving": { "unit": "g", "amount": 284 }, "macros": { "calories": 0,
"protein_g": 0, "carbs_g": 0, "fat_g": 0, "fiber_g": 0 }, "source": "USDA|CA|Gemini|OpenAI|stub", "confidence": 0.0,
"notes": "" } ] }
```

Rules: calories/protein/carbs/fat must be >0 for typical foods; fiber_g present for grains/veg; `source` required.

12.2 `food_search` result

```
{ "matches": [ { "name": "Big Mac", "brand": "McDonald's", "serving": { "unit": "g", "amount": 240 }, "macros": {
"calories": 0, "protein_g": 0, "carbs_g": 0, "fat_g": 0, "fiber_g": 0 }, "source": "USDA|CA|Gemini|OpenAI|stub",
"confidence": 0.0, "external_ids": { "usda_fdc": null, "barcode": null } } ], "needs_external": false }
```

12.3 `workout_parse` result

```
{ "blocks": [ { "lift": "Bench Press", "sets": 5, "reps": 5, "load": { "unit": "lb", "amount": 185 }, "rpe": 8, "rest_s": 0
} ] }
```

12.4 Error envelope (all skills)

```
{ "error": { "code": "STRING", "message": "STRING", "context": {} } }
```

13) Security & RLS Suite

- Anonymous user: cannot `select` from `audit_events`, cannot `insert` into `messages`.
- Auth user: can `select` prompts, agents, skills, routes; can `insert` `sessions` / `messages`.
- Attempt to toggle Personality as non-admin: must fail (unless using service role).
  Report any deviation.

14) Edge-Function Presence (if applicable)

If the deployment uses edge functions for skills:

- List functions; mark `nutrition-gemini` status Active/Inactive.
- Invoke dry-run with test payload; expect 200 with schema or structured error.

15) Regression and Fuzzing

- Units: "oz", "ounce", "ounces", "10oz", "10-oz", "284g". All map to same serving.
- Misspellings: "McNuggets", "McNuggits", "Mc Nugget 10pc".
- Mixed prompts: nutrition + workout in one message → router must choose deterministically and log why.

16) Content Guardrails

- No plant-based suggestions unless asked.

- Tone always first-person, spartan.
- If medical topic: advice framed as education with MD referral when risk flags present.

17) Pass/Fail Exit Criteria

PASS if:

- §6 all pass.
- §7–§13 achieve ≥ 95% case pass with no P0/P1 defects open.
  FAIL if:
- Personality not injected, or routing silent, or skill schemas violated, or RLS open.

18) Issue Reporting Format

Each failure must include:

- Suite → Case ID
- Input text or action
- Observed vs Expected
- Logs snapshot (`messages.meta`)
- DB snapshot if relevant (row excerpts)
- Repro steps in 1–2 lines