Integration Guide: Multi-Round Scenario Gameplay

This document provides implementation instructions for integrating a multi-round character-driven scenario feature into the existing website using the ChatGPT API. The goal is to replicate an immersive back-and-forth dialogue (like the sample Dennett session), complete with feedback scoring and gamified elements.

Architecture Overview

Flow per session: 1. /api/sessions (POST) → create a session with persona, difficulty, topic. 2. /api/sessions/{id}/turns (POST) → each round: send user question, get character reply, get feedback scores, persist everything. 3. /api/sessions/{id}/complete (POST) → finalize; compute badges, best question, summary. Server responsibilities: - Prompt templating for persona - Guardrails (topic filtering, tone, length) - Two API calls per turn (character reply + evaluator scoring) - Persistence in database - Analytics hooks Frontend responsibilities: - Chat-style UI with streaming replies - Feedback cards with scores, tip, suggested upgrade - XP, badges, streak tracking

Database Schema (example with Supabase/Postgres)

- users(id, email, display_name, created_at) - sessions(id, user_id, persona, topic, difficulty, rounds_planned, rounds_completed, created_at, completed_at) - turns(id, session_id, round_no, user_question, character_reply, feedback_json, created_at) - badges(id, code, label, description) - user_badges(user_id, badge_id, session_id, awarded_at)

Prompt Design

System Persona Prompt: - Modeled after an analytic philosopher (Dennett-like) - Domain limited to philosophy, ethics, sports, business, culture (no AI topics unless explicitly asked) - Concise, respectful, mildly challenging tone - 2–6 sentence replies Evaluator Prompt: - Scores clarity, depth, insight, openness (0–25 each) - JSON output enforced - Provides tip and suggested upgrade

API Endpoints (Next.js + Node/TypeScript Example)

1) POST /api/sessions → create session 2) POST /api/sessions/{id}/turns → process turn (character reply + scoring) 3) POST /api/sessions/{id}/complete → finalize session and award badges Streaming is recommended for character replies for smoother UX.

Frontend Implementation

- Start session → API call to create - Each round → submit question → stream reply → display feedback card - Show progress bar (round/5) - Completion → summary card with Best Question, badges, XP

Guardrails & Quality

- Topic filter: block/reframe AI-related questions - JSON reliability: enforce response_format or parse with try/catch - Rate limiting: prevent abuse - Telemetry: log session length, scores, stickiness metrics

Next Steps

1. Build basic session flow with one persona 2. Implement Daily Challenge + Deep Practice as two modes 3. Pilot test with 5–10 users for stickiness 4. Extend to additional personas, gamification, leaderboards