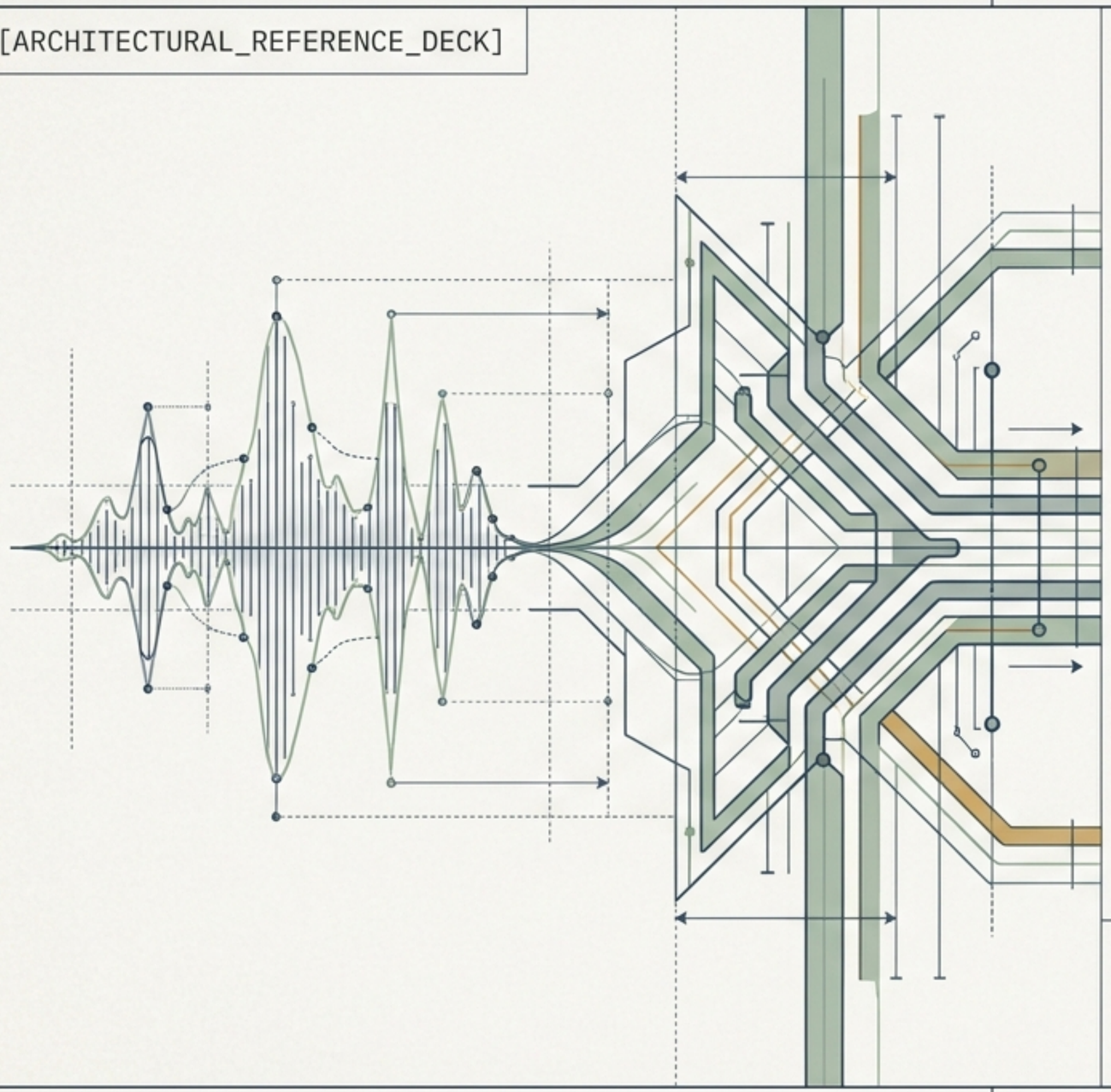
[ARCHITECTURAL_REFERENCE_DECK]



نبق: بناء منصة ذكاء اصطناعي لخدمات الصحة النفسية بلغات متعددة

تشريح النظام، هندسة التلقين، ودروس من بيئة الإنتاج الحقيقية.

التقرير الفني - الإصدار 1.0

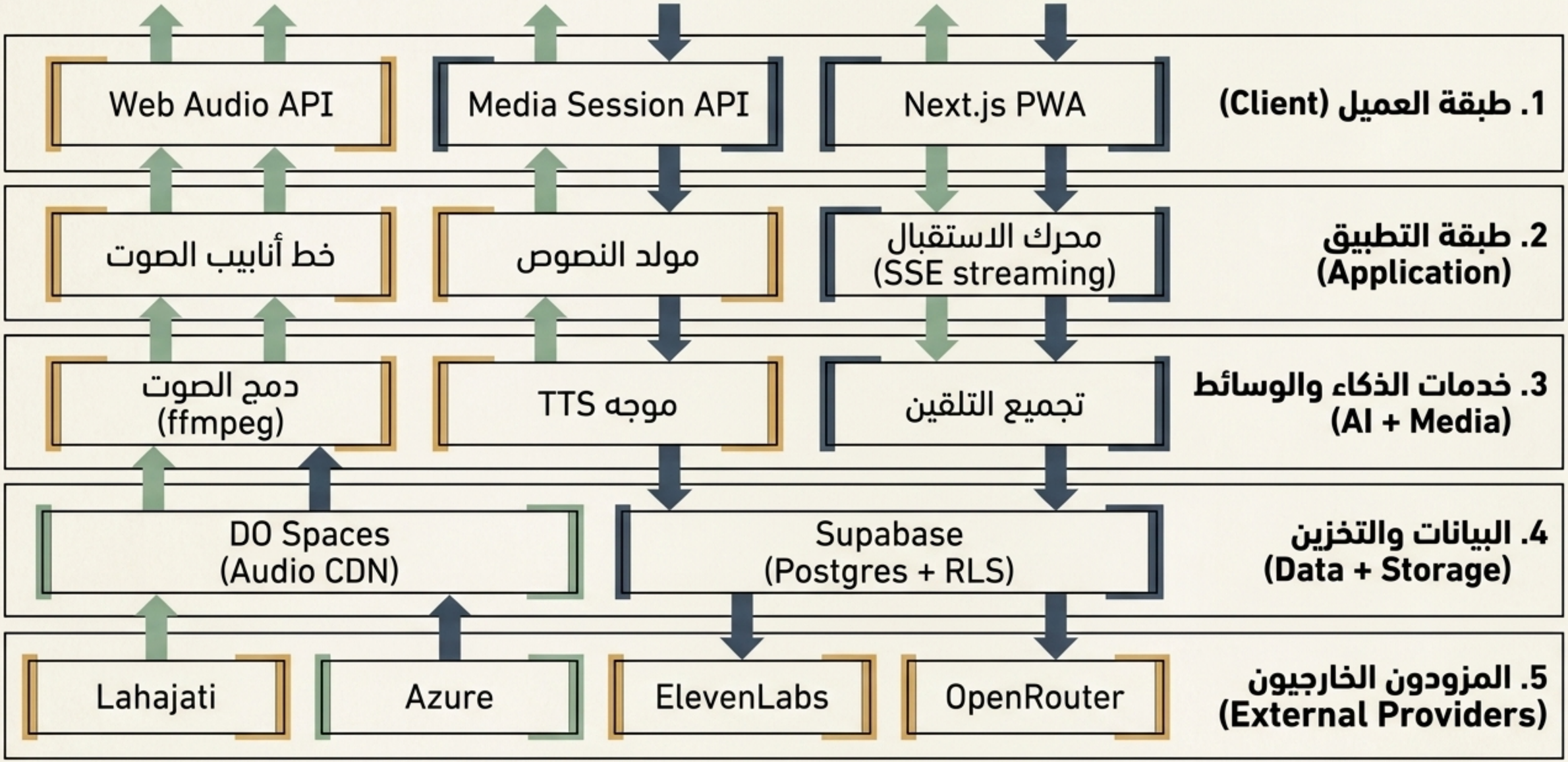
مُطوّر بواسطة: Tariq Alturkestani

الفجوة في سوق التأمل الذهني:
9 مليارات دولار تتجاهل 422 مليون ناطق بالعربية

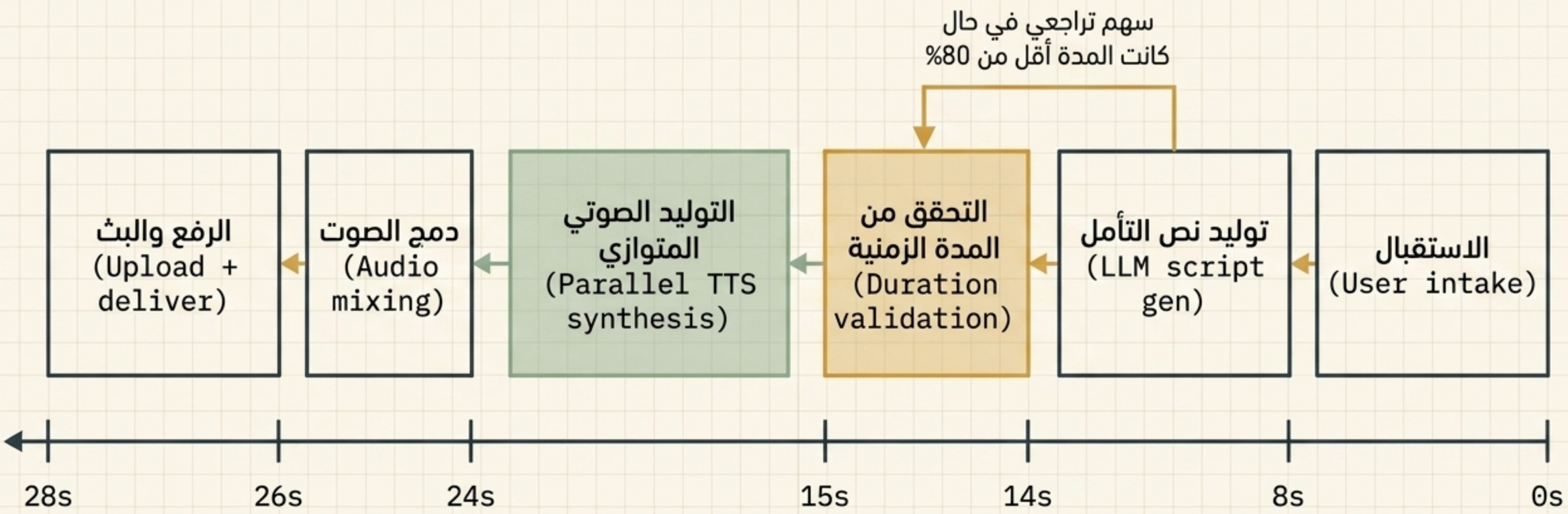
البعد	التطبيقات الحالية (Incumbents)	نظام نبق (Nabq)
نموذج المحتوى	مكتبة صوتية مُسجلة مسبقاً (Curated MP3s)	توليد نصوص ديناميكية عبر الذكاء الاصطناعي (Generative)
التركيز اللغوي	أولوية للغة الإنجليزية، لغة عربية فصحى جافة (MSA)	لهجات عربية محلية (سعودي، خليجي، مصري، تونسي)
الطابع الثقافي	علماني، عام (Secular/Generic)	أصالة ثقافية، إسلامي مُخصص (Culturally Authentic)

اللغة العربية الفصحى تخلق حاجزاً عاطفياً. التجربة الحميمية تتطلب لغة التحدث اليومية.

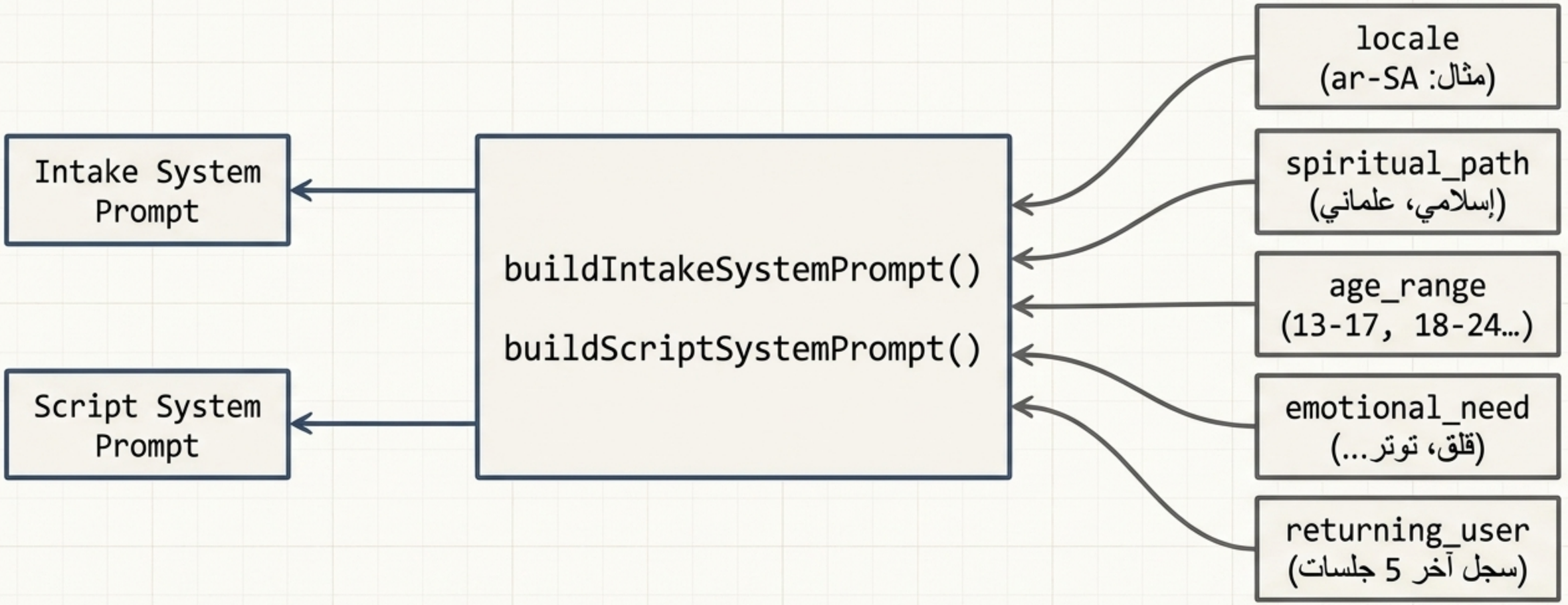
البنية الهيكلية لـ "نبق": 5 طبقات للتوليد اللحظي



رحلة الطلب: من النقر إلى الاستماع (أقل من 30 ثانية)

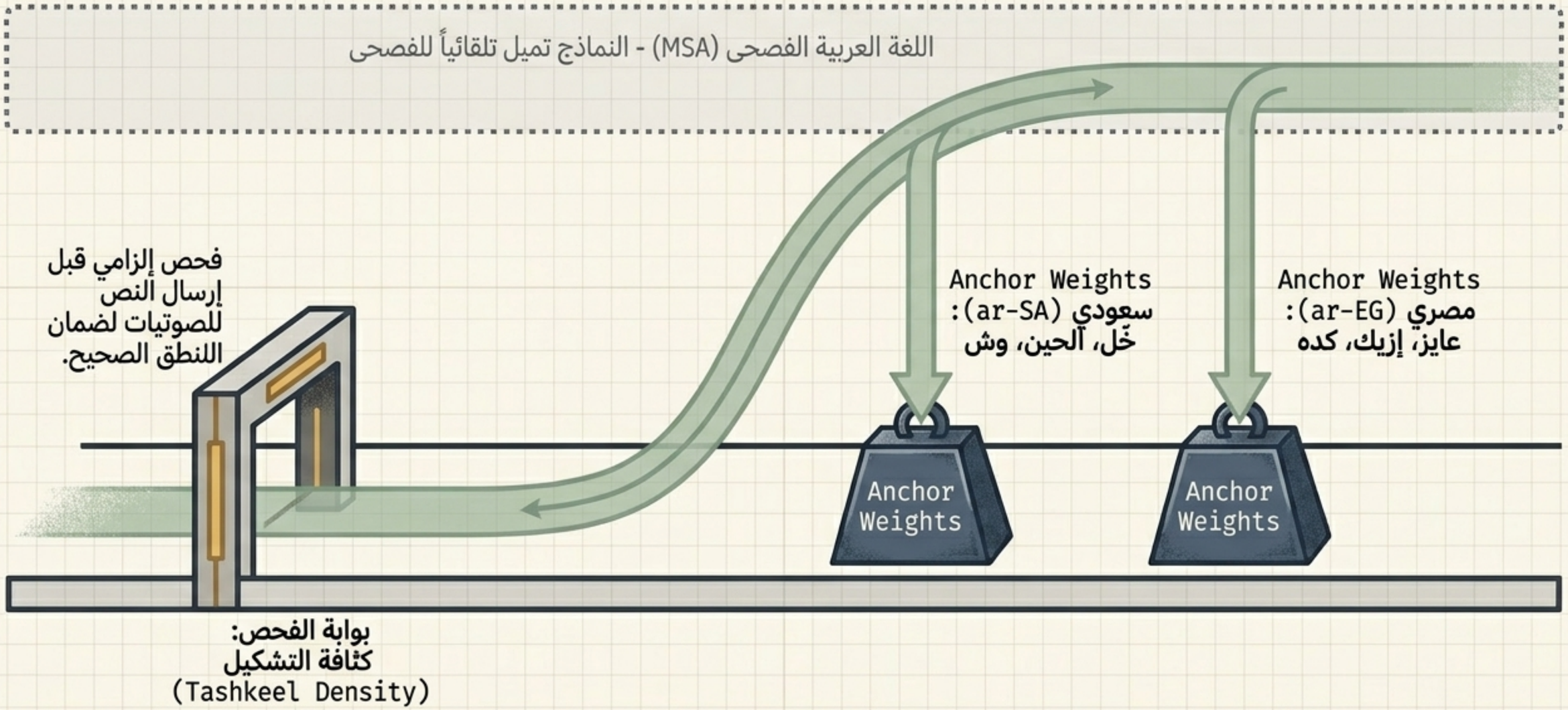


الابتكار الأول: التجميع المعياري لهندسة التلقين



المقياس الحقيقي للنجاح: إضافة لغة جديدة أو مسار روحي يتطلب كتابة قالب توجيهي واحد فقط، دون المساس بالبنية الأساسية للبرمجية.

الابتكار الثاني: فرض اللهجة العامية والتشكيل الإلزامي



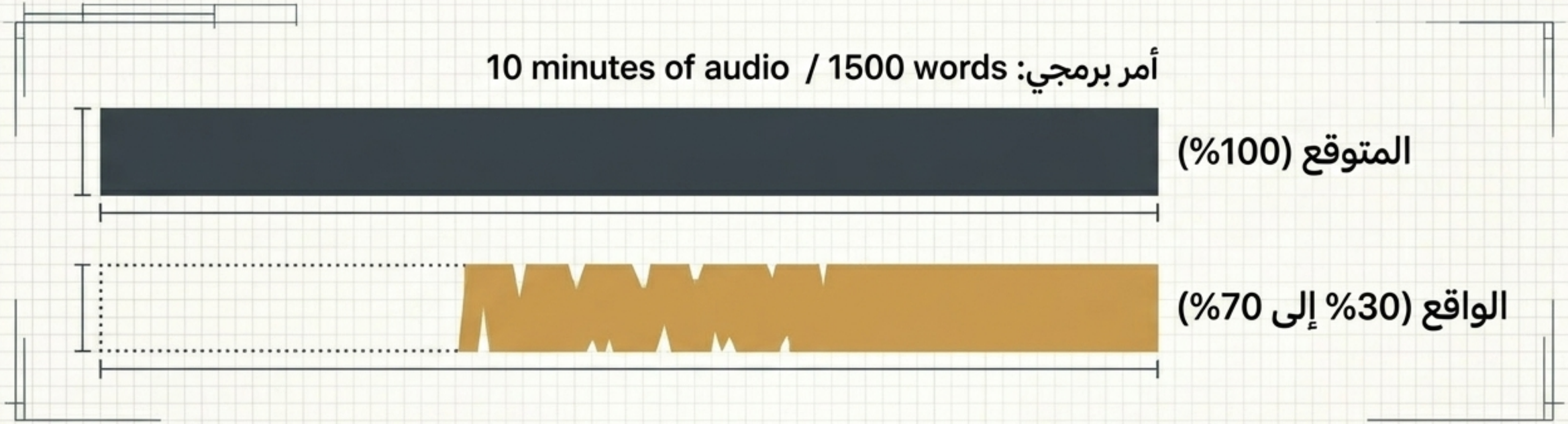
الابتكار الثالث: قبو البيانات للنصوص المقدسة

المشكلة: النماذج التوليدية قد تهلوس أو تعيد صياغة النصوص، وهذا مرفوض في القرآن والأدعية.

الحل الهندسي: التعامل مع النصوص المقدسة كـ بيانات صلبة (Data) وليس كـ مخرجات توليدية (Output).

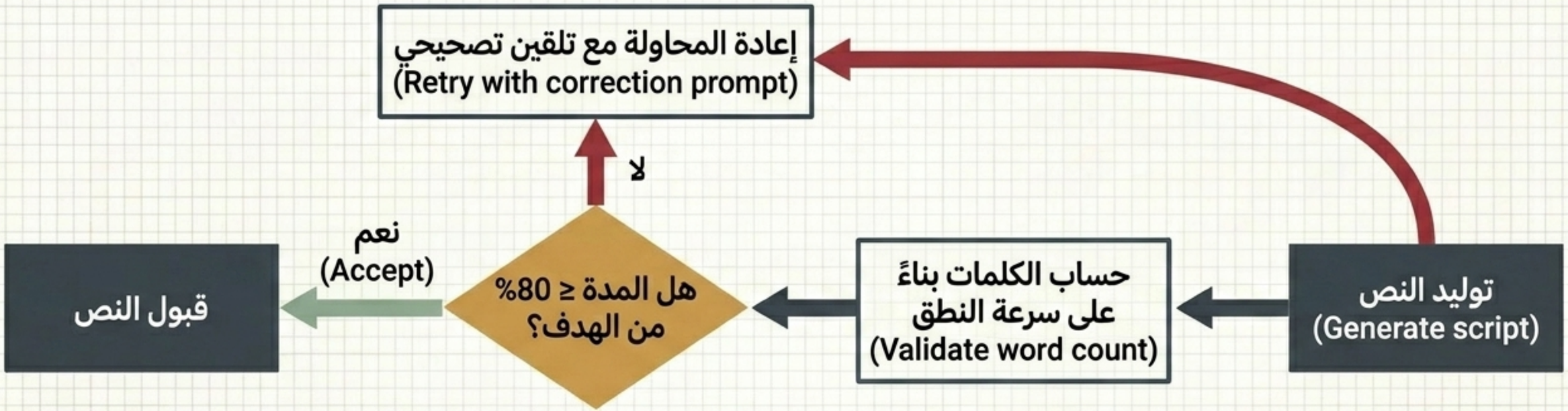
يربط النظام الحالة النفسية بالآيات المناسبة، ويُدخلها مع أمر صارم بالحفاظ على سلامتها اللغوية.

المعضلة: النماذج اللغوية لا تجيد العد



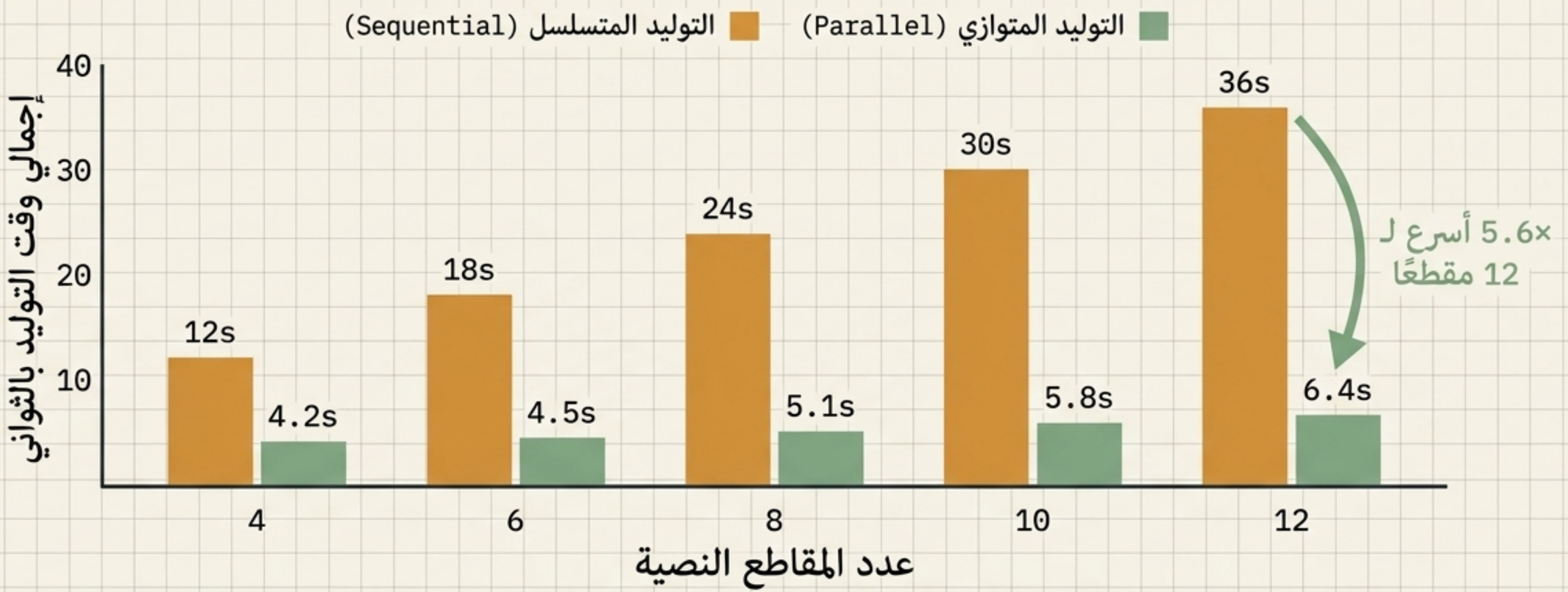
لا يمكن حل مشكلة قصر المخرجات بمجرد تعديل أوامر التلقين (Prompting). نحتاج إلى تدخل برمجي صارم على مستوى التطبيق.

الابتكار الرابع: الفرض الهيكلي للمدة الزمنية



بيانات التلقين التصحيحي:
- عدد الكلمات الحالي
- الهدف الزمني
- الأقسام القصيرة جداً
- أمر بالتوسيع دون تغيير الهيكل (الحد الأقصى: 3 محاولات)

الابتكار الخامس: تسريع التوليد الصوتي بنسبة 5.6x



نظرًا لأن مزودي TTS يحاسبون على الحرف وليس على الطلب، فإن الإرسال المتوازي للمقاطع يقلل وقت الانتظار بشكل هائل دون أي تكلفة إضافية.

الابتكار السادس: التوجيه الذكي لمزودي الصوتيات (Fallback Router)

ElevenLabs

الأفضل للإنجليزية والفرنسية الطبيعية. (تغطية ضعيفة للهجات العربية).

Azure Speech

تغطية واسعة وموثوقة (ممتاز للتركية والعربية الفصحى).

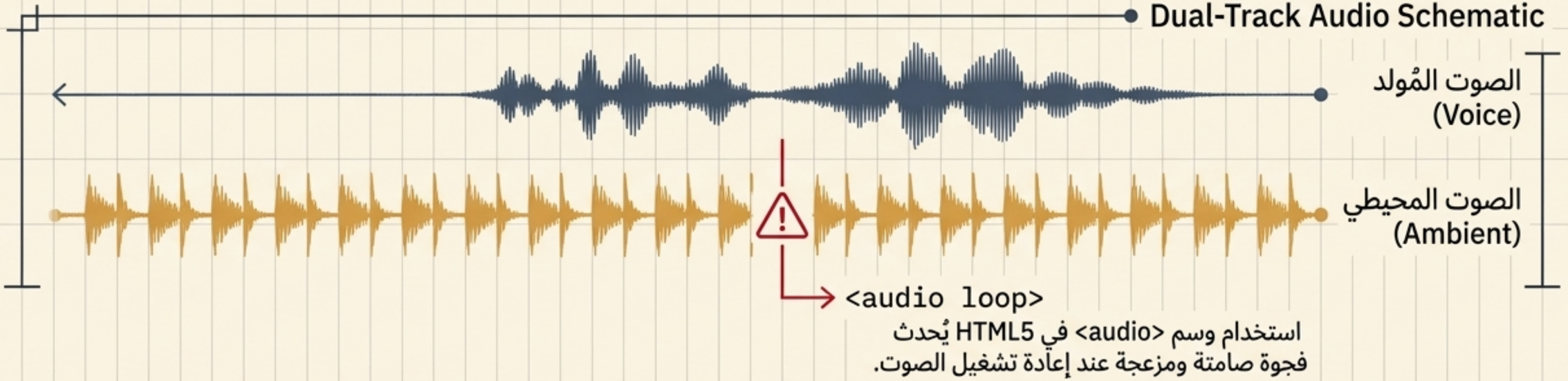
Lahajati.ai

متخصص في العمق اللهجوي العربي (متميز في اللهجات الفهجيات الخليجية).

The Router Logic

يختار النظام المزود الأفضل بناءً على اللغة (Locale)، مع نظام ارتداد تلقائي (Automatic Fallback) في حال فشل المزود الأساسي أو تأخر في الاستجابة.

الابتكار السابع: التغلب على ثغرات متصفحات الويب في دمج الصوت



الحل: مصفوفة الارتداد ثلاثية الطبقات (3-Tier Fallback)

استخدام Web Audio API مع AudioBufferSourceNode لضمان حلقة صوتية بلا فجوات (Gapless).

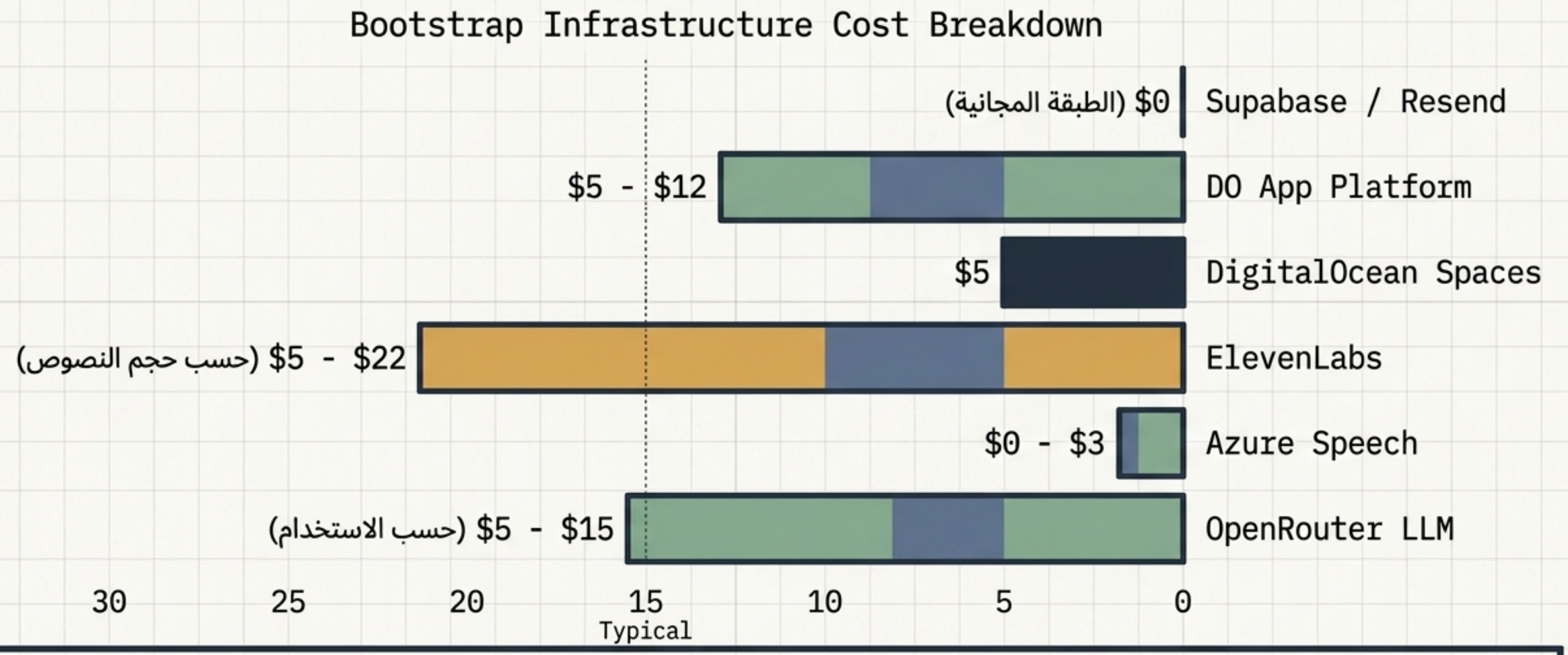
↓

الارتداد إلى HTML5 كخيار ثانوي للمتصفحات القديمة.

↓

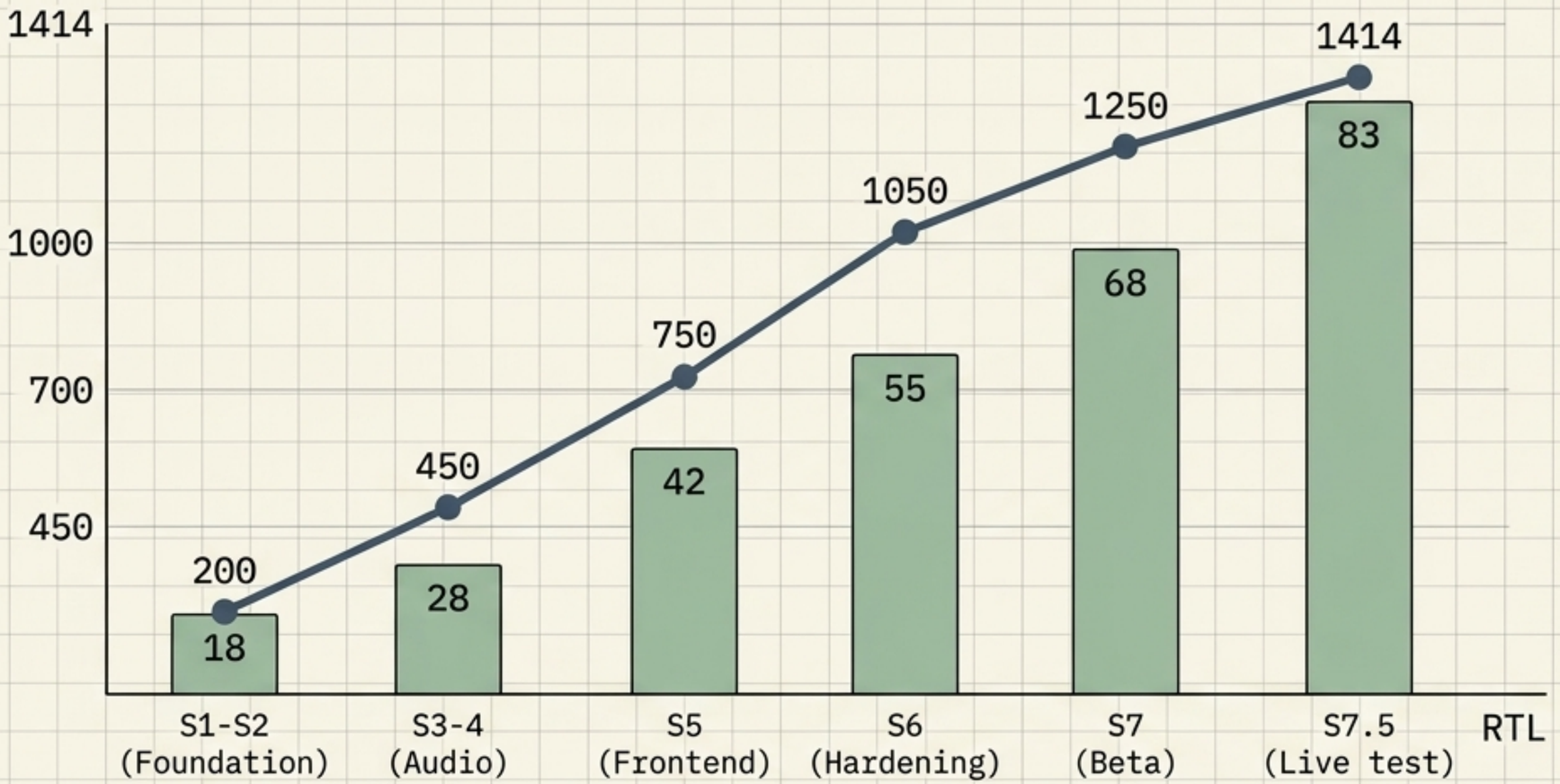
الارتداد النهائي: توليد ضوضاء بنية (Brown Noise) خوارزمياً للمتصفحات الضعيفة جداً.

اقتصاديات المُطوّر: 15$ شهرياً لتشغيل البنية التحتية



الخلاصة: تكلفة الجلسة الواحدة (توليد + صوت) تتراوح بين 0.02$ و 0.08$ فقط.
اقتصاديات تجعل من النظام نموذج عمل قابل للاستدامة السريعة.

سرعة الإنجاز بدعم الذكاء الاصطناعي: مطور واحد، 30 يوماً



7
Sprints
(دورات تطوير)

83
Merged PRs
(طلبات سحب مكتملة)

1,414
Tests
(اختبار برمجي آلي)

العمل تمت هندسته كلياً بواسطة مطور بشري واحد بالتعاون مع وكلاء ذكاء اصطناعي (Claude Code للتنفيذ، و Gemini CLI لمراجعة الأكواد).

حقائق قاسية من بيئة الإنتاج الحقيقية (الجزء الأول)

كمون المزودين (Provider Latency)

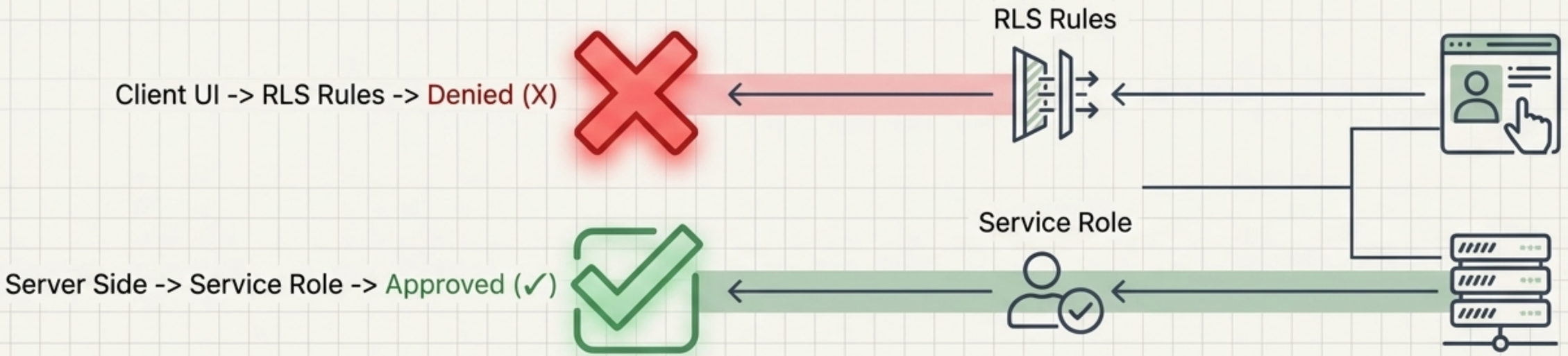
- **النظرية:** أرقام التسويق للمزودين تبدو سريعة في بيئة التطوير.
- **الواقع:** توليد 10 دقائق من الصوت بشكل متسلسل يفشل تماماً أمام صبر المستخدم.

التوليد المتوازي ليس رفاهية، بل ضرورة بقاء.

حالة النظام غير المتزامنة (Async Persistence)

- **المشكلة:** المستخدم يضغط توليد ثم يغادر الصفحة، ليعود ويجد شاشة فارغة.
- **الحل:** يجب حفظ نية التوليد (Generation Intent) في قاعدة البيانات قبل إرسال الطلب إلى النماذج اللغوية، لضمان استرجاع الحالة.

حقائق قاسية من بيئة الإنتاج الحقيقية (الجزء الثاني)



أفخاخ أمان مستوى الصف (Row-Level Security)

- **المشكلة:** سياسات RLS في قاعدة بيانات Supabase تكون خفية حتى تفشل العمليات فجأة في بيئة الإنتاج (Production).
- **الدرس:** العمليات الإدارية في الخوادم الخلفية تحتاج استخدام `createServiceClient()` مخصص لتجاوز القيود الأمنية، وإلا ستفشل الطلبات بصمت.

القاعدة الذهبية: لا تفترض أن الصلاحيات في بيئة التطوير المحلية ستعمل بنفس الطريقة على السيرفرات الحية.

الاستنتاج الجوهري: ضمانات البرمجة مقابل اقتراحات التلقين

ما يعتقده المطورون:

- يمكننا أمر النموذج بإنتاج 10 دقائق صوتية بحلقة مستمرة. (وهم)
- النماذج اللغوية (LLMs) محركات تنسيق فاشلة.

واقع الإنتاج:

- أوامر التلقين مجرد اقتراحات.
- الجودة تتطلب أكواد برمجية صارمة للتحقق، شبكات متوازية، وواجهات برمجة لضمان الاتصال.

القيمة الحقيقية في منتجات الذكاء الاصطناعي لا تكمن في النموذج نفسه، بل في هيكلة حواجز الحماية (Guardrails) وبنية الدمج حوله.

بناء المستقبل: البنية مفتوحة المصدر (MIT)

1. لغات جديدة

تتطلب قالب توجيهي واحد وملف صوتي لكل لغة.

2. مسارات روحية جديدة

تتطلب فقط إدخال مكتبة نصوص جديدة.

3. مزودون جدد

واجهة محايدة تتقبل التبديل بسهولة.

الكود البرمجي ليس هو الخندق الدفاعي (Moat)، بل المحتوى والاتصال الثقافي. المشروع متاح كمرجع تقني لبناء الجيل القادم.

المستودع: github.com/alturkestani/
رخصة الاستخدام: MIT License