BRAIN STATION 23

CUSTOMER CASE STUDY · EDTECH / GENERATIVE AI

PadhAI

AI-Powered Personalised Tutoring for Emerging Markets — built for scale across **7 emerging markets**, with a 70/20/10 AI model strategy that keeps unit economics sane while delivering a premium tutoring experience.

INDUSTRY	ENGAGEMENT	SCOPE	ISSUED
EdTech / Generative AI	Full-stack product engineering	Architecture, build & launch readiness	April 2026

EXECUTIVE SUMMARY

01 The 60-second story

Brain Station 23 partnered with the founders of **PadhAI** to architect and deliver an AI-native tutoring platform built for the realities of emerging markets — low-bandwidth networks, WhatsApp-first behaviour, regional curricula, mobile-money payments, and a hard ceiling on per-student AI cost.

12	7	9	3
PRODUCTION SERVICES	TARGET MARKETS	PAYMENT GATEWAYS	CHANNELS: WEB / WHATSAPP / TELEGRAM

What we delivered

- A **polyglot microservices platform** — 9 Node.js / TypeScript services and 2 Python / FastAPI AI services — orchestrated through an API gateway, message queues, and an event bus.
- A patentable **70/20/10 AI model router** that dynamically classifies query complexity and dispatches to lightweight, mid-tier, or premium LLMs — cutting the projected AI cost per active student by an order of magnitude versus naive single-model usage.
- An **adaptive assessment engine** with knowledge tracing, spaced repetition, and exam-pattern generators, plus a **Socratic tutor** with confidence scoring, math verification, content safety, and RAG over curriculum content.
- **Channel parity** — the same student can resume a lesson on a Next.js PWA, on WhatsApp, or on Telegram, with session state preserved server-side.
- A **localisation & compliance backbone** covering Bangla / English UIs, country-specific curricula and pricing, parental controls, content moderation, and data-privacy primitives (DPDP / GDPR-ready).

> *"We engineered PadhAI so that the same architecture can launch in Dhaka on Monday and in Lagos the next quarter — without re-platforming."*

THE CLIENT

02 About PadhAI

PadhAI is an AI-native edtech startup with a clear thesis: that 500 million+ school-age children in developing countries deserve the same quality of one-on-one tutoring that wealthier peers take for granted — delivered at a price-point and in a channel that actually fits their lives.

Mission

Personalised, curriculum-aligned tutoring for grades 1–12 across Bangladesh, India, Nigeria, Indonesia, Brazil, Pakistan and the Philippines — in local languages, on the channels students already use.

Stage & Constraints

Solo-founder, AI-native, pre-launch. Required a production-grade platform that a small team could operate, with unit economics that worked at sub-$5/month price points.

What made this engagement interesting

- **Polyglot from day one.** Python where AI tooling is strongest; Node/TypeScript where business logic, payments and integrations dominate.
- **WhatsApp- and Telegram-first.** The web app is a PWA, but the primary acquisition channel is conversational, not browser-based.
- **Cost-aware AI.** A single GPT-class model on every query would have made the business unviable. The router was a non-negotiable design constraint, not an optimisation.
- **Multi-country compliance.** Curriculum boards (e.g., NCTB in Bangladesh), local payment rails (bKash, Nagad, Razorpay, M-Pesa-class), and data-residency considerations baked into the schema.

THE CHALLENGE

03 What had to be solved

Edtech in emerging markets is a graveyard of well-funded products that ignored network reality, payment rails, or curriculum specificity. PadhAI's brief was to avoid every one of those failure modes — simultaneously.

Constraint	Implication for engineering
Per-student AI cost ceiling	Must not call a frontier model on every turn. Need provider abstraction, dynamic routing, caching, and aggressive prompt economy.
WhatsApp / Telegram as a primary UI	Stateless HTTP webhooks must reconcile with stateful tutoring sessions; identity must work without an app install.
Low-bandwidth, intermittent networks	Web client must be a PWA with offline behaviour; payloads must be small; long-poll/WebSocket patterns avoided where possible.
Multiple curricula & languages	Content schema must separate *concept* from *presentation*; UIs must be i18n-clean; AI prompts must be curriculum-aware.
Local payments	Subscription engine must abstract over 9 gateways with very different reconciliation models (bKash, Nagad, Razorpay, Stripe, PayPal, etc.).
Trust & safety for minors	Content moderation, parental controls, and audit trails are first-class — not afterthoughts.
Solo-founder operability	Architecture must be observable, container-native, and runnable end-to-end on a single laptop for development.

Bottom line: the platform had to behave like a Series-B product on day one — without the headcount of a Series-B team. That shaped every choice that follows.

OUR APPROACH

04 How Brain Station 23 structured the build

We treated PadhAI as a **product engineering** engagement, not a body-shop project. That meant owning the architecture, the AI cost model, the data design, and the path to production — with the founder in the loop on every load-bearing decision.

1. Architect for parity

One source of truth for student state — web, WhatsApp, and Telegram are channels into the same domain, not separate apps.

2. Architect for cost

AI model routing was designed before any LLM call was written. Every prompt has a tier; every tier has a budget.

3. Architect for geography

Curriculum, language, payments and compliance are configurable per country — not hard-coded for a single launch market.

Engineering practices we standardised

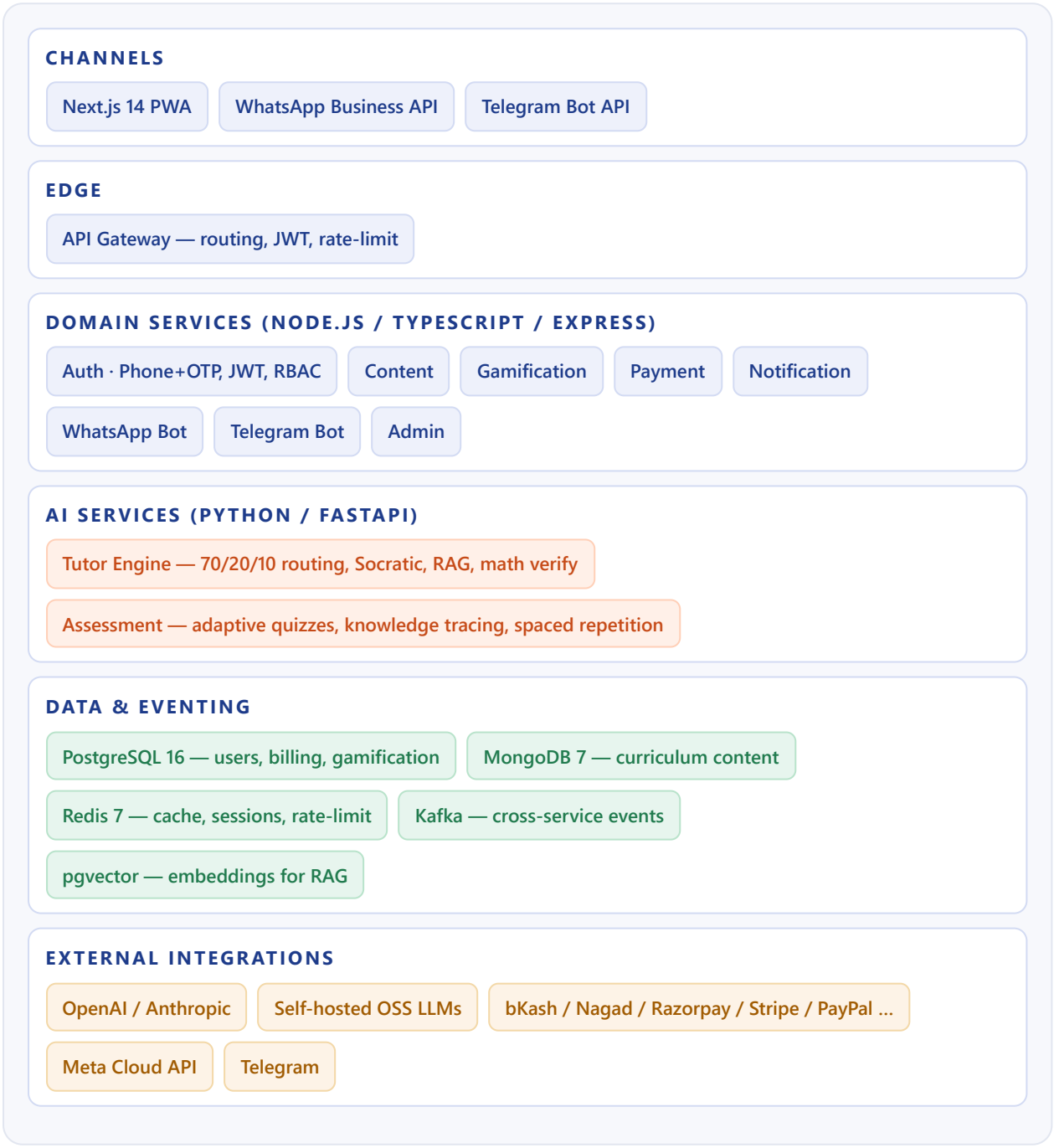
- **Monorepo with pnpm workspaces** for the Node services + web app; Poetry-style projects for Python services; shared TypeScript package for cross-service types.
- **TypeScript strict mode** across all Node code; **Python type hints + Ruff** on every Python file.
- **Uniform API envelope** — every service returns `{ success, data?, error? }` and exposes `/health`.
- **Structured JSON logging** via pino (Node) and structlog (Python), ready for any log aggregator.
- **Validated env at startup** — misconfiguration is a boot-time failure, never a 3am incident.
- **Docker-first local dev** — a single `docker compose up` brings up Postgres, MongoDB, Redis and Kafka; `pnpm dev` brings up all services in parallel.

> *The deliverable wasn't a codebase — it was an operable product: a single founder can run the whole stack, ship a feature, and read its logs without a platform team.*

SOLUTION ARCHITECTURE

05 The platform, at a glance

A polyglot microservices estate, deliberately split along the lines of *conversational surface*, *core domain*, and *AI brain*. Each service owns its data, exposes a typed API, and is independently deployable.



Why this shape

- **Two databases, on purpose.** Postgres for transactional, relational state (users, subscriptions, attempts, XP). MongoDB for content that is fundamentally document-shaped (chapters, lessons, multimedia). Redis for ephemeral state.
- **Kafka, not just for hype.** Streak updates, notifications, and analytics fan-out cleanly off domain events — without coupling services synchronously.
- **Python only where it earns its place.** The Tutor and Assessment engines are Python because that is where the AI tooling lives. Every other service is Node/TypeScript, because that is where the integration ecosystem is densest.

SIGNATURE INNOVATION

06 The 70/20/10 AI brain

PadhAI's economic moat is the AI router. Every incoming query is scored on a 1–10 complexity scale and dispatched to one of three model tiers — with automatic escalation if a tier fails.

Tier	Target traffic	Provider class	Used for
Tier 1 — Lightweight	~70%	Self-hosted / OSS	Definitions, recall, simple Q&A, hint scaffolds
Tier 2 — Mid-range	~20%	Mid-tier API	Step-by-step explanations, calculations, common reasoning
Tier 3 — Premium	~10%	Frontier API	Deep reasoning, proofs, advanced higher-math, edge cases

The scoring function

Every query receives a composite complexity score:

```
score = complexity · 0.4 + reasoning_depth · 0.3 + subject_difficulty · 0.3
```

where *complexity* is derived from keyword banks (e.g., "prove", "derive", "integrate" pull score up; "define", "list", "name" pull it down), *reasoning_depth* is inferred from the question structure, and *subject_difficulty* is a per-subject weight scaled by grade level. Higher Math at grade 12 hits Tier 3; "What is photosynthesis?" at grade 6 stays on Tier 1.

Beyond routing — what makes the tutor good

Socratic prompt builder

The model is steered to *guide*, not to answer outright — producing hints, asking clarifying questions, and revealing solutions only after the student commits to an attempt.

Math verifier

A symbolic engine independently checks numeric and algebraic answers, catching the class of LLM "confidently wrong" failures that destroy trust in tutors.

Confidence scorer

Every response carries a model-internal confidence that drives whether to escalate, ask for clarification, or hand off to a human curator.

RAG over curriculum

Curriculum-aligned passages are retrieved via pgvector embeddings and grounded into prompts — so the tutor explains *this textbook's* photosynthesis, not a generic one.

Content safety

Pre- and post-generation guardrails block self-harm, abuse, and off-curriculum drift — non-negotiable for a product used by minors.

Session memory

Sessions persist server-side with rolling summaries, so a student can pause on web and resume on WhatsApp without losing context.

ADAPTIVE LEARNING

07 Assessment that actually adapts

A separate assessment service runs alongside the tutor — because *checking what a student knows* is a different problem from *teaching them*, and deserves its own engine.

Adaptive quiz generator

Question difficulty is selected per-student, per-topic — using attempt history, mastery estimates, and exam-pattern templates that mirror the local board's actual papers.

Knowledge tracing

Per-concept mastery is updated after every answer, feeding both the recommendation engine and the parent dashboard.

Spaced repetition

Concepts the student is about to forget are surfaced — via WhatsApp nudges and in-app cards — on a forgetting-curve schedule.

Free-text answer evaluation

Beyond MCQs — short answers, fill-in-the-blanks, and ordered/matching question types are graded with rubric-driven LLM evaluation.

Question types supported: MCQ, fill-in-the-blank, true/false, short answer, matching, ordering, and free text.

CHANNELS & COMMERCE

08 Where students meet the product, and how they pay

Three channels, one student

The same identity, the same session, the same gamification — whether the student opens a Next.js PWA, sends a WhatsApp message, or DMs a Telegram bot. Channel adapters normalise inbound messages into a single domain event vocabulary.

Next.js PWA

Installable on Android home screens, offline-aware banners, instant-install prompts, and a Tailwind-driven UI in Bangla and English.

WhatsApp Bot

Meta Cloud API connector with webhook verification, media handling, and the same tutor backend — reaching students who will never download an app.

Telegram Bot

Long-poll and webhook modes, group-aware, with parity to the WhatsApp surface for markets where Telegram dominates.

Nine payment gateways under one abstraction

Subscription billing is the single hardest non-AI problem in this market. We built a payment service that exposes a uniform plan / subscription / webhook API, with adapters for the gateways that matter in each launch country.

- bKash
- Nagad
- SSLCommerz

Bangladesh

- Razorpay
- UPI

India

- Stripe
- PayPal

Global & cards

- Paystack
- Flutterwave

Nigeria & Africa

Each gateway integration includes idempotent webhook handling, reconciliation jobs, refund flows, and country-specific tax / receipt formatting.

TRUST, SAFETY & COMPLIANCE

09 Built for users who are minors

A product used by 6–18 year olds carries duties that a B2B SaaS does not. We baked these in — not bolted them on.

Parental controls

Parent accounts linked to children, screen-time limits, content visibility settings, and weekly progress digests — all enforced at the service layer, not just the UI.

Content moderation

Curator role and review pipeline (*draft* → *review* → *approved* → *published*) for human-authored content, plus automated guardrails on AI output.

Data privacy

Dedicated migrations for consent, data export, and deletion — aligned with GDPR-class and Bangladesh DPDP-class requirements. Soft-deletes and audit timestamps on every user-facing table.

Auth that fits the market

Phone-number + OTP login as the default — because email isn't universal — with JWT access tokens, refresh rotation, and role-based access control.

BY THE NUMBERS

10 Scope, in figures

A snapshot of what Brain Station 23 delivered into the PadhAI repository.

12 PRODUCTION SERVICES	**35+** POSTGRESQL TABLES	**8** SCHEMA MIGRATIONS	**2** PYTHON AI SERVICES
9 NODE.JS SERVICES	**3** DATASTORES (PG/MONGO/REDIS)	**3** USER CHANNELS	**9** PAYMENT INTEGRATIONS
7 TARGET LAUNCH COUNTRIES	**2** LANGUAGES AT LAUNCH (BN/EN)	**12** SUBJECTS MODELED (G1–12)	**3** AI MODEL TIERS

Domain coverage in the schema

The PostgreSQL design covers users & auth, curriculum boards, subjects, chapters, lessons, attempts, XP & streaks, badges, leaderboards, subscriptions, payments, notifications, parent links, parental controls, content moderation, referral system, vector embeddings for RAG, and data-privacy artefacts (consents, deletion requests, export jobs).

TECH STACK

11 What's under the hood

Layer	Technology
Frontend	Next.js 14, React 18, TypeScript, Tailwind CSS, PWA (installable, offline-aware), i18n (Bangla / English)
Domain services	Node.js 20 LTS, TypeScript 5 (strict), Express, Zod for validation, pino for logs, ioredis, axios
AI services	Python 3.12, FastAPI, uvicorn, structlog, pydantic, OpenAI & Anthropic SDKs, OSS-LLM provider abstraction
Datastores	PostgreSQL 16 (with pgvector), MongoDB 7, Redis 7
Eventing & queues	Apache Kafka (KRaft mode), BullMQ for in-process job queues
Auth	Phone + OTP, JWT (access + refresh), RBAC across student / parent / teacher / curator / admin roles
Channels	Meta WhatsApp Cloud API, Telegram Bot API, Web Push
Payments	bKash, Nagad, SSLCommerz, Razorpay, UPI, Stripe, PayPal, Paystack, Flutterwave
Local dev	Docker Compose, pnpm workspaces, Python venvs, single-command bootstrap
Quality	ESLint + Prettier, Ruff, Vitest (Node), Pytest (Python), validated env at startup, /health on every service
Observability	Structured JSON logs (pino / structlog), trace-ready, ready for Grafana / Loki / OpenTelemetry

OUTCOMES

12 What the founder walked away with

An operable, launch-ready platform — not a prototype, not a slide deck.

Time-to-market collapsed

What would normally be a multi-team, multi-quarter build was delivered as a coherent monorepo a single founder can run, extend, and ship from.

AI cost designed in, not bolted on

The 70/20/10 router protects unit economics from day one — the difference between a viable freemium business and a money-burning demo.

Multi-country by construction

Curriculum, language, currency, and payment-rail variation are configuration — not new code paths — for the next launch market.

Trust & safety, audited

Parental controls, content moderation, data-privacy primitives, and audit trails are first-class — making investor and regulator conversations dramatically easier.

Polyglot, but coherent

Python where it pays, Node where it pays, one envelope across all APIs, one logging convention, one health protocol — so a small team can hold the whole system in their head.

A defensible moat

The router, the curriculum-grounded RAG, the math verifier and the channel parity together form an architecture that is hard to replicate from a generic LLM wrapper.

> *The product the founder set out to build is the product Brain Station 23 delivered — on a foundation that won't have to be re-platformed at the Series A.*

WHY BRAIN STATION 23

13 What this engagement says about us

PadhAI is the kind of build that shows what Brain Station 23 actually does well: hard, multi-stack, AI-native product engineering — with founder-grade ownership of architecture, cost, and quality.

AI-native, not AI-curious

We design AI products around routing, evaluation, grounding and safety — not around a single API call.

Polyglot by default

Node + Python + Postgres + Mongo + Redis + Kafka, in production, in one repo — without it becoming a maintenance tax.

Emerging-markets fluent

Mobile-money, multi-curriculum, multi-language, low-bandwidth UX are first-instinct concerns, not retrofits.

Founder partner, not vendor

We sit on the architecture and unit-economics decisions — not just the ticket queue.

Operable on day one

Every service has /health, structured logs, validated env, and a one-command local bring-up. That's what shipping looks like.

Compliance-ready

Privacy, moderation, parental control and audit are baked into the schema — ready for GDPR-class regimes.

Capabilities demonstrated by this engagement

AI Product Architecture · LLM Routing & Cost Engineering · RAG over Domain Content · FastAPI Microservices · Node/TypeScript Microservices · Next.js PWAs · WhatsApp / Telegram Bots · Multi-Gateway Payments · PostgreSQL + pgvector · MongoDB Content Modeling · Kafka Event Streaming · RBAC & OTP Auth · Parental Controls & Moderation · Data Privacy Engineering · Docker & Monorepo DX · EdTech Domain Modeling · Emerging-Markets UX · i18n / l10n (Bangla, English, …)

LET'S BUILD SOMETHING HARD

14 Talk to Brain Station 23

If you're building something AI-native, multi-channel, multi-country — or you have a prototype that needs to grow up into a product — this is the kind of engagement we live for.

What we typically ship

- AI product architecture & LLM cost design
- Polyglot microservices estates (Node + Python)
- Conversational surfaces (WhatsApp, Telegram, voice)
- Payments, subscriptions and reconciliation
- Compliance-ready data design (GDPR, DPDP)
- Mobile-first PWAs and React Native apps

How to start a conversation

Send a paragraph — what you're building, who it's for, what hurts. We'll come back with a one-page architecture sketch and a concrete first-90-days plan.

Email info@brainstation-23.com
Web www.brainstation-23.com