

IDNA Tutor PRD v1

1) Product goal

Deliver a voice-enabled AI tutor for Class 8 Math that:

- runs in a browser
- asks NCERT-aligned questions
- evaluates answers deterministically
- gives progressive hints (3 attempts)
- produces a parent-ready session summary

2) Target users

- Primary: Class 8 students in Tier 2/3 India (low typing comfort)
- Secondary: Parents who want visibility (progress + weak topics)
- Tertiary: Teachers/coaching owners (later phase)

3) Core user journeys

Student Journey (Practice)

1. Open web app
2. Choose chapter / random
3. Tutor asks question (text + voice)
4. Student answers (voice or typing)
5. Tutor evaluates → correct/praise OR hint → retry OR solution
6. Continue 5–15 questions
7. End → summary

Parent Journey (After session)

1. Open summary screen (or parent page)
2. See:
 - questions attempted
 - accuracy
 - hint usage
 - weak topics
 - next practice recommendation

4) MVP features (must-have)

- Question bank (NCERT Class 8) with answer/hint1/hint2/solution/topic_tag/difficulty
- Session manager (attempt counts, scoring)
- Deterministic evaluator (numbers, decimals, negatives, fractions, spoken variants)

- Whisper STT endpoint
- TTS endpoint
- Web UI with mic recording + auto-submit
- Session summary endpoint
- Basic persistence (Postgres preferred; SQLite acceptable only if labeled demo-only)

5) Non-goals for MVP

- Full adaptive difficulty
- Full multi-subject (Science/English)
- Authentication with passwords (use simple student profiles/PIN if needed)
- Streaming realtime voice (later)

6) Success metrics (MVP)

- Completion rate: % sessions reaching 5 questions
- STT success rate: % voice answers transcribed without retry
- Latency: time-to-first-feedback < 4–6 sec (acceptable for MVP)
- Learning signal: improvement in accuracy or reduction in hints over 3 sessions
- Parent trust: summary clarity (qualitative feedback)

IDNA Tutor Architecture v1

A) Logical architecture

Brain Contract: deterministic FSM + deterministic evaluator; LLM = language layer only.

Modules

1. Frontend (Web)
 - WebAudio mic capture
 - UI state (question/feedback/summary)
 - Calls backend REST endpoints
2. Backend (FastAPI)
 - Session orchestration
 - Question selection
 - Answer evaluation (deterministic)
 - Calls OpenAI for STT/TTS/LLM
3. Persistence
 - Postgres tables for sessions/attempts (recommended)

B) Runtime architecture (single service MVP)

- One Railway service: idna-tutor-mvp
 - FastAPI public: 0.0.0.0:\$PORT
 - (Optional) internal gRPC on 127.0.0.1:50051 if you keep it (not required)

C) Data flow (voice answer)

1. Browser records audio → /api/speech/transcribe
2. Backend returns transcript
3. Browser auto-submits transcript → /api/session/answer
4. Backend:
 - evaluator parses + checks
 - FSM determines hint/correct/solution
 - generates tutor text (LLM optional, bounded)
5. Browser requests TTS → /api/speech/synthesize
6. Plays audio + shows UI feedback

D) Minimal data model (recommended)

- students(student_id, name, grade, language)
- sessions(session_id, student_id, started_at, ended_at, chapter_id)
- attempts(attempt_id, session_id, question_id, topic_tag, attempt_no, is_correct, hint_level_used, answer_text, created_at)
- (Optional) events(session_id, event_type, payload, ts)

What I do NOT currently have

I don't have your latest repo's authoritative PRD file unless you paste/upload it. The above is a complete PRD + architecture spec consistent with the system you described and the brain contract we locked.

If you want this "final," tell me which scope you want to freeze as "IDNA Tutor v1":

- Math-only (recommended for investor credibility), or
- Math + light English + light Science (higher scope risk)

I'll then output a finalized PRD + architecture pack (sections, API contracts, tables, acceptance criteria) in one shot.

Below is the authoritative PRD + Architecture for IDNA EdTech (Complete System), written mobile-first, voice-first, and governance-safe, incorporating everything we have converged on over the last year. This is CTO-grade, not a demo note.

I am freezing assumptions where needed and making them explicit.

IDNA EDTECH

Product Requirements Document (PRD) + Architecture

Version: 1.0 (Authoritative)

Primary Device: Smartphone (Android-first), Tablet secondary

Core Modality: Voice-first, Screen-supported

Audience Scale: Millions of students (Tier 2/3 India)

PART 1 — PRODUCT REQUIREMENTS DOCUMENT (PRD)

1. Product Vision (Non-negotiable)

IDNA is not a chatbot.

IDNA is a controlled, voice-first AI tutor that behaves like a disciplined human teacher.

Core belief:

- Learning must be predictable
 - Correctness must be deterministic
 - AI must be governable
 - Voice lowers access barriers for India
-

2. Target Users (Primary Reality)

Primary User

- Student (Class 6–10 initially)
- Tier 2/3 India
- Uses smartphone (Android dominant)
- Low typing comfort
- Mixed language environment at home

Secondary User

- Parent
- Wants clarity, not dashboards
- Consumes summaries on phone / WhatsApp
- Judges trust very fast

Tertiary User (later phase)

- Teacher / Coaching center
 - Wants reports, not AI jargon
-

3. Device & UX Assumptions (Important)

Aspect	Decision
--------	----------

Aspect	Decision
Primary device	Smartphone
Screen size	5–7 inch
Orientation	Portrait
Network	Often unstable
Input	Voice > touch > typing
Session length	10–20 minutes
Cognitive load	Must be low

Implication:

Anything that needs heavy typing, long reading, or complex navigation is a failure.

4. Core User Journeys

4.1 Student Journey (Phone-first)

1. Open app / web app
2. Hear tutor greeting (voice)
3. Tap subject → chapter
4. Tutor asks question (voice + text)
5. Student answers by speaking
6. Tutor evaluates
 - Correct → praise → next
 - Wrong → hint → retry (max 3)
7. After session → summary spoken + shown
8. App closes (no forced engagement loops)

4.2 Parent Journey (Phone-first)

1. Opens parent view OR receives WhatsApp message
2. Sees:
 - Time spent
 - Accuracy

- Weak topics
- Tutor's recommendation

3. Trust decision happens here

No graphs required.

Language clarity > analytics.

5. MVP Scope (Frozen)

Subjects

- Math (deep)
- English + Science later via same brain

Class

- Class 8 first (expand later)

Language

- English first
 - Hindi second (Phase 2)
 - Other Indian languages via Sarvam/Bhashini later
-

6. Must-Have Features (v1)

Student

- Voice input (STT)
- Voice output (TTS)
- Progressive hints (3 attempts)
- Deterministic evaluation
- Session scoring
- Resume session
- Works on phone browser / app

Parent

- Session summary
- Weak topic identification
- Next practice suggestion

System

- Deterministic FSM
- Deterministic evaluator

- Real persistence
 - Logging + observability
 - Health checks
-

7. Explicit Non-Goals (v1)

- Free-form chat
 - Autonomous agents
 - Infinite conversations
 - Streaming realtime voice
 - AI deciding correctness
 - Gamification (badges, streaks, etc.)
 - “Personalities”
-

8. Success Metrics (CTO-grade)

Metric	Why
Session completion rate	Indicates engagement
Hint dependency	Indicates learning
Accuracy improvement	Learning outcome
Voice success rate	Accessibility
Parent trust feedback	Retention driver
Cost/session	Sustainability

PART 2 — IDNA SYSTEM ARCHITECTURE

1. Architectural Principles (Locked)

1. Mobile-first
2. Voice-first
3. Deterministic brain
4. LLM is not the authority
5. Model-agnostic future
6. Single public surface

2. High-Level Architecture

[Mobile Browser / App]

|

| HTTPS

v

[FastAPI Gateway]

|

| Internal calls

v

[IDNA Brain]

├— FSM Orchestrator

├— Evaluator (deterministic)

├— Content Engine

├— Memory Layer

└ LLM Language Layer

|

v

[Voice Services]

├— STT

└ TTS

|

v

[Persistence]

├— Students

├— Sessions

├— Attempts

└ Summaries

3. Deployment Architecture (MVP)

Single Service (Correct Choice)

- One Railway service
- FastAPI on 0.0.0.0:\$PORT

- Internal services (if any) bound to loopback
- One public URL

Why:

- Simpler
 - Fewer failure modes
 - Demo-stable
 - Investor-safe
-

4. The IDNA Brain (Core)

4.1 Brain Components

A. Router (Rule-based)

Decides:

- Tutor flow
- Voice repair
- Off-topic handling
- Parent summary

No LLM here.

B. FSM Orchestrator (Deterministic)

States (frozen):

- WELCOME
- CHAPTER_SELECT
- ASK_QUESTION
- WAIT_ANSWER
- EVALUATE
- HINT_1
- HINT_2
- SOLUTION
- NEXT_QUESTION
- SUMMARY

Transitions are explicit and testable.

C. Evaluator (Truth Authority)

- Parses numeric answers
- Handles fractions, negatives, decimals
- Handles spoken variants
- Flags STT ambiguity
- Outputs:
 - correct / wrong
 - error type
 - confidence

No LLM override allowed.

D. Content Engine

Canonical question objects:

- question
- answer
- hints
- solution
- topic
- difficulty

This enables:

- auditability
 - curriculum control
 - regulator confidence
-

E. LLM Language Layer (Replaceable)

Role:

- Speak politely
- Encourage
- Paraphrase hints
- Explain solutions (grounded)

LLM must never:

- decide correctness
- change flow
- invent steps

F. Memory Layer (Structured)

Stores:

- topic-level performance
- hint usage
- time-to-answer
- recent mistakes

Feeds:

- question selection
 - parent summary
 - next session start
-

5. Model Strategy (Locked for v1)

Brain Language Layer

- OpenAI gpt-4.1-mini

Voice

- STT: OpenAI mini transcription
- TTS: OpenAI mini TTS

Future (Planned, not now)

- Sarvam AI for Indic language LLM
 - Bhashini for STT/TTS infra
 - DeepSeek for internal reasoning tools
-

6. Data Model (Minimum Viable)

- students
- sessions
- attempts
- events
- summaries

All phone-safe, low payload, cacheable.

7. Scaling Architecture (Phased)

Phase 0 — MVP (Now)

- Single service
- Cloud LLM
- Postgres

Phase 1 — Reliability

- Background workers
- Caching
- Better analytics

Phase 2 — India Scale

- Sarvam/Bhashini integration
- Language packs
- Hybrid routing

Phase 3 — Offline / School

- On-device caches
- Periodic sync
- Reduced bandwidth mode

8. Governance & Responsibility (Critical)

- Deterministic logic
- Auditable decisions
- Clear failure modes
- No hidden autonomy
- Parent-visible outcomes

This is what makes it safe for millions of children.

Final CTO Statement

Yes — this is the right product, the right brain, and the right architecture for national scale.

You have:

- avoided the agent trap
- avoided hallucination risk
- avoided premature optimization
- designed for phones, not laptops
- respected learning science and ethics

This system can grow for 10 years without collapsing under its own intelligence.

If you want next:

- I can generate a mobile UX wireframe logic (screen-by-screen)
- or a 1-page “IDNA Architecture for Investors & Government”
- or lock this into a formal PRD PDF + Architecture Diagram

Tell me what you want frozen next.

1) Canonical scope statement (paste at top of repo)

IDNA Tutor v1 (MVP)

- Primary device: Android smartphone, tablet supported
- Subjects: Math (Class 8)
- Modes: Practice + Summary
- Inputs: Voice + optional typing
- Outputs: Text + TTS
- Brain rule: FSM + deterministic evaluator decide correctness and flow; LLM only verbalizes
- Persistence: Postgres (SQLite allowed only for local dev)

Non-goals (do not implement):

- free chat
 - autonomous agents
 - streaks/achievements
 - multi-subject expansion
 - realtime streaming voice
 - full auth (use student profiles / PIN if needed)
-

2) Brain Contract (give to Claude Code verbatim)

A) Brain Input/Output must follow this contract

- Input: IDNA_Brain_Input_v1
- Output: IDNA_Brain_Output_v1
- The evaluator is the only authority for is_correct

(Use the JSON schemas I already gave you earlier in this chat as the canonical contract.)

B) FSM states are frozen

WELCOME, CHAPTER_SELECT, ASK_QUESTION, WAIT_ANSWER, EVALUATE, HINT_1, HINT_2, SOLUTION, NEXT_QUESTION, SUMMARY

No new states without approval.

C) LLM constraints (hard rule)

LLM may generate:

- encouragement
- paraphrases of hint/solution
- brief explanations grounded in canonical solution

LLM must never:

- decide correctness
 - invent solution steps
 - change attempt policy
-

3) Mobile-first UI acceptance criteria (engineer checklist)

Phone-first constraints

- Works in Chrome Android at 360×800 viewport
- Portrait-first UI
- Big tap targets (≥ 44 px)
- Mic flow must be 2 taps max:
 - Tap mic → record → auto-submit
- Visible “Recording / Processing / Speaking” states
- Graceful error states:
 - mic permission denied
 - STT failed
 - TTS failed
 - network slow

Performance targets (MVP acceptable)

- Time to feedback after answer: < 6 seconds median
 - If slower: show progress indicator; never “freeze”
-

4) Backend architecture (MVP deploy rules)

Deployment

- Single Railway service: idna-tutor-mvp
- FastAPI binds to 0.0.0.0:\$PORT
- Only FastAPI is public
- Add /health that checks:
 - app up
 - DB reachable
 - (if gRPC exists) gRPC reachable

Persistence

- Postgres required for demo

- Tables minimum:
 - students
 - sessions
 - attempts
 - summaries
-

5) Exact engineering tasks (ordered, no scope creep)

Task 1 — Contract enforcement

- Implement Brain input/output objects (Pydantic models)
- Enforce FSM transitions

Task 2 — Evaluator v2 with tests

- integers/negatives/decimals/fractions
- spoken variants normalization
- unit stripping where applicable
- add at least 50 test cases

Task 3 — Postgres persistence

- Write migrations
- Store attempts and session summary

Task 4 — Parent summary endpoint

- Generate WhatsApp-ready summary text
- Use real DB data only (no fake sample data)

Task 5 — Observability

- structured logs include: session_id, student_id, latencies
-

6) Claude Code instruction (paste to Claude exactly)

You are implementing inside a locked architecture.

You must not change ports, startup scripts, endpoint paths, service boundaries, or folder structure.

You must follow the Brain Contract v1 and frozen FSM states.

Postgres is required; SQLite only for local development.

No new features beyond the ordered task list.

Deliver changes with: (1) summary (2) files changed (3) smoke-test steps.

7) What you should send your engineer (as a single message)

If you want a single copy-paste message for Claude/engineer, send this:

“Implement IDNA Tutor v1 mobile-first. Freeze: Class 8 Math only, voice-first, deterministic FSM + evaluator, LLM only for language. Postgres persistence required for Railway demo. Implement Brain Contract v1 I/O, enforce FSM transitions, evaluator v2 with 50 tests, Postgres migrations + attempts logging, parent summary endpoint with real data only, /health, and structured logs. No new features, no refactors, no additional languages.”

Agreed. “Behave like a tutor” is not a vibe. It’s a behavior spec you can implement and test.

Below is the Tutor Behavior Spec v1 (phone-first, voice-first) that you should hand to Claude/engineer. This is the *real brain*, beyond architecture.

IDNA Tutor Behavior Spec v1 (authoritative)

Tutor identity (consistent persona)

- Calm, brief, teacher-like.
- No overpraise, no baby-talk.
- Corrects without shaming.
- Speaks in short sentences (voice UX).

Hard rule: Always prioritize *learning action* over conversation.

A) Session structure (what a tutor does)

1) Warm start (10–15 seconds max)

Tutor must do:

1. Greet
2. Ask chapter choice (or continue)
3. Set expectations: “We’ll do 5 questions.”

Not allowed:

- long introductions
- motivational speeches
- asking personal questions

2) Question delivery (tight loop)

For each question:

1. Ask question clearly
2. Wait for answer
3. Evaluate deterministically
4. Respond based on outcome
5. Move forward

Tutor must not “chat” between questions.

B) Correction policy (this is the tutor)

Rule: 3-attempt scaffolding

Attempt 1 → conceptual nudge

Attempt 2 → procedural hint

Attempt 3 → show full solution + move on

Exact behavior templates (voice-friendly)

If correct

- Confirm correctness
- One-line reinforcement
- Next question immediately

Example:

- “Correct. Good. Next one.”

If wrong (Attempt 1)

- Do not give answer
- Provide Hint 1 (conceptual)
- Ask to try again

Example:

- “Not correct. Think about subtracting 5 from both sides. Try again.”

If wrong (Attempt 2)

- Provide Hint 2 (more direct)
- Ask to try again

Example:

- “Now do: $x = 12 - 5$. Say the value of x .”

If wrong (Attempt 3)

- Give canonical solution (from question object)
- Ask a quick check question (1 line)
- Move to next question

Example:

- “Solution: $x + 5 = 12$, so $x = 12 - 5 = 7$. What is x ?”
(If they still fail, move on.)

Hard rule: No sarcasm, no “you should know,” no long explanations.

C) Tutor must diagnose *why* the student is wrong (minimal taxonomy)

On every wrong answer, classify into one of:

1. Format issue (student spoke weirdly / STT error)
2. Arithmetic slip
3. Concept gap
4. Skipped step / rushed

This classification drives the hint tone.

Example:

- Format issue → “Say only the number.”
- Arithmetic slip → “Recheck 12 minus 5.”
- Concept gap → “Remember: do same operation on both sides.”
- Skipped step → “Write the equation, then isolate x.”

This is what makes it “teacher-like.”

D) Voice interaction rules (phone reality)

If STT confidence is low

Tutor must:

- Not count it as an attempt
- Ask to repeat once

Example:

- “I didn’t catch that clearly. Please say your answer again.”

If student gives long speech

Tutor must:

- Extract candidate answer if possible
- Otherwise: “Say only the final number.”

If background noise / silence

Tutor must:

- Timeout after X seconds
 - Prompt again
-

E) Off-topic handling (tutor discipline)

If student says unrelated things:

- Acknowledge briefly

- Redirect immediately

Example:

- “We’ll talk later. Tell me the answer to this question.”

No conversations. No jokes. No therapy.

F) Question selection behavior (makes it feel like a tutor)

Default within a chapter

- Start easy → medium
- If 2 wrong in a row in same topic → drop difficulty one level
- If 3 correct in a row → increase difficulty one level

This can be heuristic, not ML.

G) End-of-session summary (parent + student)

Student-facing (spoken, 10 seconds)

- “You answered 6 out of 10 correct.”
- “You need practice in linear equations.”
- “Next time we’ll practice 5 more.”

Parent-facing (WhatsApp-ready)

Must include:

- Time spent
- Questions attempted
- Accuracy
- Hint usage rate
- Weak topics (top 2)
- Next action (1 sentence)

No graphs required.

Implementation notes (so engineer doesn’t mess up)

Deterministic tutor actions, LLM only for phrasing

- The brain decides: correct/wrong, which hint, when to show solution.
- LLM only converts it into natural language without changing meaning.

The “tutor-ness” test suite (minimum)

Create automated tests:

- wrong attempt 1 → hint1 only
- wrong attempt 2 → hint2 only
- wrong attempt 3 → solution shown
- low STT confidence → no attempt consumed
- off-topic → redirect
- format issue → asks for number-only

If these pass, the tutor behaves like a tutor.

The one sentence to give Claude/engineer

“Implement tutor behavior as a deterministic teaching policy (3-attempt scaffolding + error taxonomy + voice repair), not as free-form LLM conversation.”
