Kibundo – Learning State Target Model (Developer Specification)

First of all: what already works well

Thanks again for the detailed answers. The current backend setup is a strong foundation. In particular, the separation of session data and profile data, updating learning signals after task completion instead of per message, and the concept of injecting a summarized learning context into the AI requests are all very solid decisions. This allows us to extend the system without refactoring core components.

To make implementation easier and avoid ambiguity, we want to explicitly define the expected learning-state model below. Please treat the following as the target behavior and semantics. Internal storage and implementation details can vary, but the outputs and logic should match exactly.

```
1) Skill level (required, minimal but explicit)
Each learning unit must map to a stable skill_id.

Examples:
- math.addition.carry_10
- math.subtraction.borrow_10
- de.spelling.capitalization_nouns
- de.reading.main_idea_simple
- sci.seasons.year_cycle

Skill IDs should be versioned if changed (e.g. skill_version = v1).

------------------------------------------------

2) Mastery score (0–100) + status mapping
Each skill has:
- mastery_score: integer (0–100)
- evidence_count: integer

Status is DERIVED from mastery_score:
- 0–39   → weak
- 40–69  → improving
- 70–100 → secure

Status must NOT change unless evidence_count >= 3.

------------------------------------------------

3) Mastery score update rules (simple & deterministic)

Increase score when:
- task solved correctly with 0–1 hints  → +10
- task solved correctly with 2–3 hints → +5

No change when:
- task solved with >3 hints
- task partially solved

Decrease score when:
- same error_type repeated in same skill → –5
- task abandoned or strong frustration detected → –5 (max once per session)

Optional:
- mild decay (–2) if skill not practiced for a long time (e.g. 30 days)

------------------------------------------------

4) Error pattern catalog (actionable)

Math examples:
- carry_missing
- place_value_confusion
- counting_strategy_overuse
- operation_confusion_add_sub

German examples:
- capitalization_missing
- phonetic_spelling_error
- word_boundary_error
- sentence_structure_confusion

Each error pattern links to a skill_id and has an occurrence counter.

------------------------------------------------

5) OCR / task → skill mapping

Each task classifier must output:
- skill_id
- confidence (0–1)

Rules:
- confidence >= 0.7 → auto-assign skill
- confidence < 0.7 → fallback required:
  - ask the child/parent to confirm the topic, OR
  - mark skill as “unknown” and let the tutor ask a clarifying question

Confidence should be passable to the tutor model.

------------------------------------------------

6) AI context injection (mandatory)

For EVERY AI request, the backend must pass either:
- a summarized learning_context object, OR
- learning_context = null (explicit, e.g. new child)

Example context:

{
  "learning_context": {
    "grade": 2,
    "current_subject": "math",
    "current_skill_id": "math.addition.carry_10",
    "weak_skills": ["math.addition.carry_10"],
    "common_errors": ["carry_missing"],
    "preferred_explanations": ["visual", "story"],
    "frustration_level": "medium"
  }
}

This must be traceable and logged so we can verify that the context is always injected.

------------------------------------------------

7) Privacy

Please align wording and handling with GDPR (not POPIA).
Learning profiles should work with child_id only; name should be optional.
Please confirm retention and deletion handling.

------------------------------------------------

Final check required:
Please explicitly confirm that all of the above information is reliably passed to the AI API on ever
```

Once the above points are confirmed, we are fully aligned and can proceed without refactoring the existing system.