NCTB Class 3 Bengali AI Tutor System

System Analysis and Design Report

Technical Specifications & Architectural Documentation

Submitted By

Group 11

Department of Computer Science & Engineering
Bangladesh University of Engineering and Technology
(BUET)

AI Learning Platform — Kaggle Prototype

February 28, 2026

0 Contents

1 Technical Specifications

The technical foundation of the *NCTB Class 3 Bengali AI Tutor* is built upon a hybrid architecture combining offline data processing and real-time inference pipelines. This section details the specifications of the core components.

1.1 Data Processing Knowledge Engineering

The system relies on a pre-processed knowledge base derived from physical textbooks. This offline pipeline transforms unstructured PDF data into a semantic vector index.

Specification: Offline Knowledge Pipeline

This component acts as the "Source of Truth," converting the Class 3 Bengali textbook into searchable vector chunks to ground the AI's responses.

lightblue Component	Specification Details
Input Data	Scanned PDF document of the NCTB textbook.
OCR Engine	**Easy OCR**: Selected for its superior capability in extracting Bengali Unicode text from scanned images compared to standard OCR tools.
Text Segmentation	**Recursive Character Splitter**: Segmentation with overlap to preserve semantic context between sentences.
Embedding Model	**paraphrase-multilingual-MiniLM-L12-v2**: Generates high-dimensional dense vectors optimized for Bengali semantic similarity.
Vector Store	**FAISS (Facebook AI Similarity Search)**: Utilized for high-performance similarity search and retrieval.

Table 1: Offline Data Pipeline Specifications

1.2 Real-Time RAG Inference Pipeline

The core interaction layer processes voice inputs and generates curriculum-aligned audio responses in real-time.

Specification: Audio-Visual AI Pipeline

Orchestrates Speech-to-Text, Neural Retrieval, LLM Inference, and Text-to-Speech to provide a seamless conversational experience.

Sub-System	Technical Implementation
Audio Input	WAV format recording via Flutter Native Audio plugin.
Speech-to-Text (STT)	**OpenAI Whisper Medium**: Deployed for robust Bengali speech recognition and auto-language detection.
Context Retrieval	**Top-k Retrieval**: Fetches the most relevant textbook chunks based on Cosine Similarity.
LLM Engine	**Phi-3.5 Mini**: • Quantization: Optimized 4-bit quantization. • Context Window: Sufficient token limit for handling retrieved context. • Hardware: Efficient inference on standard consumer GPUs or CPUs.
Text-to-Speech (TTS)	**Microsoft Edge-TTS**: Uses the `bn-BD-NabanitaNeural` voice for natural, child-friendly audio synthesis.

Table 2: Real-time Inference Specifications

2 Architectural Decisions

The system architecture was designed to address the primary constraint: the lack of a **Unified Platform combining NCTB Curriculum with Personalized AI**. The following strategic decisions justify the chosen technology stack.

2.1 Retrieval-Augmented Generation (RAG) over Fine-Tuning

Decision: RAG Architecture

We utilized a RAG pipeline instead of fine-tuning a model on the textbook data to strictly control the accuracy of generated answers.

Problem Standard Large Language Models (LLMs) suffer from "hallucination," often generating plausible but incorrect facts not aligned with the NCTB curriculum.

Justification RAG forces the model to generate answers *only* using the retrieved textbook context. This ensures that the AI adheres strictly to the provided educational material without introducing external or grade-inappropriate information.

2.2 Small Language Model (SLM) on Local Infrastructure

Decision: Phi-3 Mini & Local Hosting

Adoption of the Phi-3.5 Mini model hosted via Ollama instead of relying on external cloud APIs like GPT-4.

- **Data Privacy:** Since the application targets children, privacy is paramount. By running the model locally, voice data and queries never leave our secure server infrastructure.
- **Sustainability:** This approach ensures **Zero API Dependency**. While cloud APIs charge per token, our self-hosted architecture has a fixed low operational cost, making it scalable for public education.
- **Performance:** The quantized model runs efficiently on standard hardware while maintaining high reasoning capabilities for educational content.

2.3 Unified Persistence Layer

Decision: PostgreSQL + FAISS/pgvector

Implementation of a unified database strategy to manage both structured user data and unstructured vector embeddings.

- **Simplified Architecture:** Instead of maintaining a separate vector database and a relational database, we use an integrated vector search solution within our backend infrastructure.

- **Atomic Operations:** This allows for complex queries, such as filtering vector searches based on a student's grade level or previous quiz performance in a single database transaction.

2.4 Mobile-First Frontend Strategy

Decision: Flutter Framework

Selection of Flutter for the client-side application to ensure cross-platform compatibility and high interactivity.

3 System Requirements

This section documents the complete set of functional and non-functional requirements for the NCTB Class 3 Bengali AI Tutor system. These requirements were derived from a structured analysis of the target user group (Bengali-medium primary school students aged 9–14), the educational context of the NCTB curriculum, and the technical constraints of deployment in Bangladeshi households.

3.1 Functional Requirements

3.1.1 Authentication and User Management

The system shall permit parents to register an account using their name, email address, phone number, and a hashed password. Authentication for both parent and student accounts shall be implemented via JWT-based token authentication with defined expiry periods. Parents shall be able to create and manage child profiles, specifying the student's name, grade level, and preferred language. Separate login flows shall be maintained for parent and student roles, and all active session tokens shall be invalidated upon logout.

3.1.2 Course and Content Management

The system shall provide a browsable course library filterable by grade level and subject area. Enrollments shall be initiated by parents on behalf of their children. Course content shall be delivered across multiple modalities including video lessons, textbook references, and audiobook resources. Each course shall be structured into sequential content modules with defined difficulty levels and estimated completion durations. The system shall passively track content engagement metrics including time-on-screen, scrolling behaviour, and module completion status.

3.1.3 Adaptive Quiz and Assessment

The system shall generate quizzes dynamically based on the student's current mastery level and the associated lesson content. Students shall receive instant scoring and AI-generated feedback upon submission. All quiz attempts shall be persisted, recording the score, time taken, selected answers, and per-question grading breakdown. Quiz difficulty shall adjust automatically based on prior performance history. Upon quiz completion, the student's mastery level and progress record shall be updated accordingly.

3.1.4 AI Tutor Session

The system shall provide a bilingual (Bangla and English) AI-powered voice tutoring interface available after lesson completion. The AI tutor shall operate via a Retrieval-Augmented Generation (RAG) pipeline grounded exclusively in NCTB curriculum content represented as vector embeddings. Student voice input shall be transcribed using the Whisper medium speech-to-text model with auto-language detection. Context-aware responses shall be generated using the locally hosted Phi-3.5 Mini language model, and synthesised into natural-sounding Bengali speech using Microsoft Edge TTS. The system shall infer student frustration levels from interaction logs and adapt its communication style accordingly. All tutoring session data shall be logged for subsequent progress analysis and parental review.

3.1.5 Interactive Learning Activities

The system shall include a suite of curriculum-aligned interactive activities:

- Drag-and-drop grammar exercises for auxiliary verb placement in Bengali sentences.
- Fill-in-the-blank multiplication grid puzzles with an on-screen number pad.
- A Pronoun Detective Quest requiring students to identify pronouns within story passages.
- A Visual Math Tree activity to build conceptual understanding of arithmetic operations.

All activities shall provide immediate visual feedback on correct and incorrect responses, with contextual hints offered for incorrect attempts.

3.1.6 Progress Tracking and Personalised Learning Path

The system shall maintain a personalised learning path for each student, dynamically sequencing content modules based on accumulated learning analytics. Completion percentage, mastery level, and cumulative time-on-task shall be tracked per content module. Daily learning goals shall be generated automatically, calibrated to each student's current capability. The recommended content sequence shall be re-evaluated and updated following each quiz attempt and AI tutoring session.

3.1.7 Gamification

The system shall award experience points (XP) upon completion of lessons, quizzes, and AI tutor sessions to incentivise engagement. A daily login streak counter shall encourage consistent platform usage. An ELO-style student rating system within courses shall support healthy peer comparison. The system shall additionally support student-created challenges and weekly competitive contests.

3.1.8 Parent Dashboard and Notifications

The system shall provide parents with a real-time dashboard presenting each child's progress metrics, XP totals, and login streak. Push notifications shall be dispatched to parents upon achievement milestones and upon detection of persistent student difficulty.

Parental controls shall include configurable screen time limits and sound preferences. Visual analytics shall highlight the child's strong and weak subject areas to support informed parental engagement.

3.2 Non-Functional Requirements

3.2.1 Performance

Standard REST API endpoints shall respond within 500 milliseconds under normal load conditions. End-to-end AI tutor voice responses—from receipt of transcribed student input to audio playback initiation—shall be delivered within 3 seconds to preserve natural conversational flow. The platform shall sustain a minimum of 1,500 concurrent daily active users without measurable response time degradation. All video and audio lesson content shall be served via a CDN to ensure low-latency media delivery across varied network conditions in Bangladesh.

3.2.2 Scalability

The Django REST API layer shall support horizontal scaling through Docker containerisation behind an Nginx reverse proxy. The PostgreSQL database layer shall be architected to accommodate growth to 50,000 or more registered students without requiring structural redesign. The marginal cost of onboarding each additional student shall approach zero at scale, enabled by AI-driven content generation in place of manual content authoring.

3.2.3 Security

All API communication shall be secured over HTTPS/TLS. User passwords shall be stored as salted cryptographic hashes; plaintext credentials shall never be persisted at any layer. JWT tokens shall carry defined expiry periods and shall be invalidated immediately upon logout. Access to all student data shall be restricted to authenticated and authorised users only. The system shall comply with child data protection principles consistent with COPPA guidelines. Critically, all AI inference—including STT, LLM, and TTS—shall execute locally on the server; no student voice or query data shall be transmitted to third-party AI API providers.

3.2.4 Reliability and Availability

The system shall target 99.5% uptime, achieved through deployment on a Linux VPS with Gunicorn and Nginx providing process-level fault tolerance. Automated database backups shall run at regular scheduled intervals to prevent data loss. Docker-based containerisation shall guarantee consistent runtime behaviour across development, staging, and production environments.

3.2.5 Usability

The Flutter-based frontend shall present a child-friendly, high-contrast, and visually engaging interface appropriate for students aged 9–14. The application shall fully support both Bangla and English throughout the UI and all AI interactions. The system shall operate correctly on the low-to-mid range Android and iOS devices prevalent in Bangladeshi

households. All interactive activities shall be operable by children with minimal adult supervision and shall require no prior technical knowledge.

3.2.6 Maintainability

The backend codebase shall follow Django best practices using a modular, app-based architecture to allow independent development and testing of each component. The AI pipeline—comprising the RAG retrieval engine, Whisper STT, Phi-3.5 Mini, and Edge-TTS—shall be decoupled from core business logic, enabling model upgrades without system-wide code changes. Environment-based configuration management shall cleanly separate development, staging, and production settings.

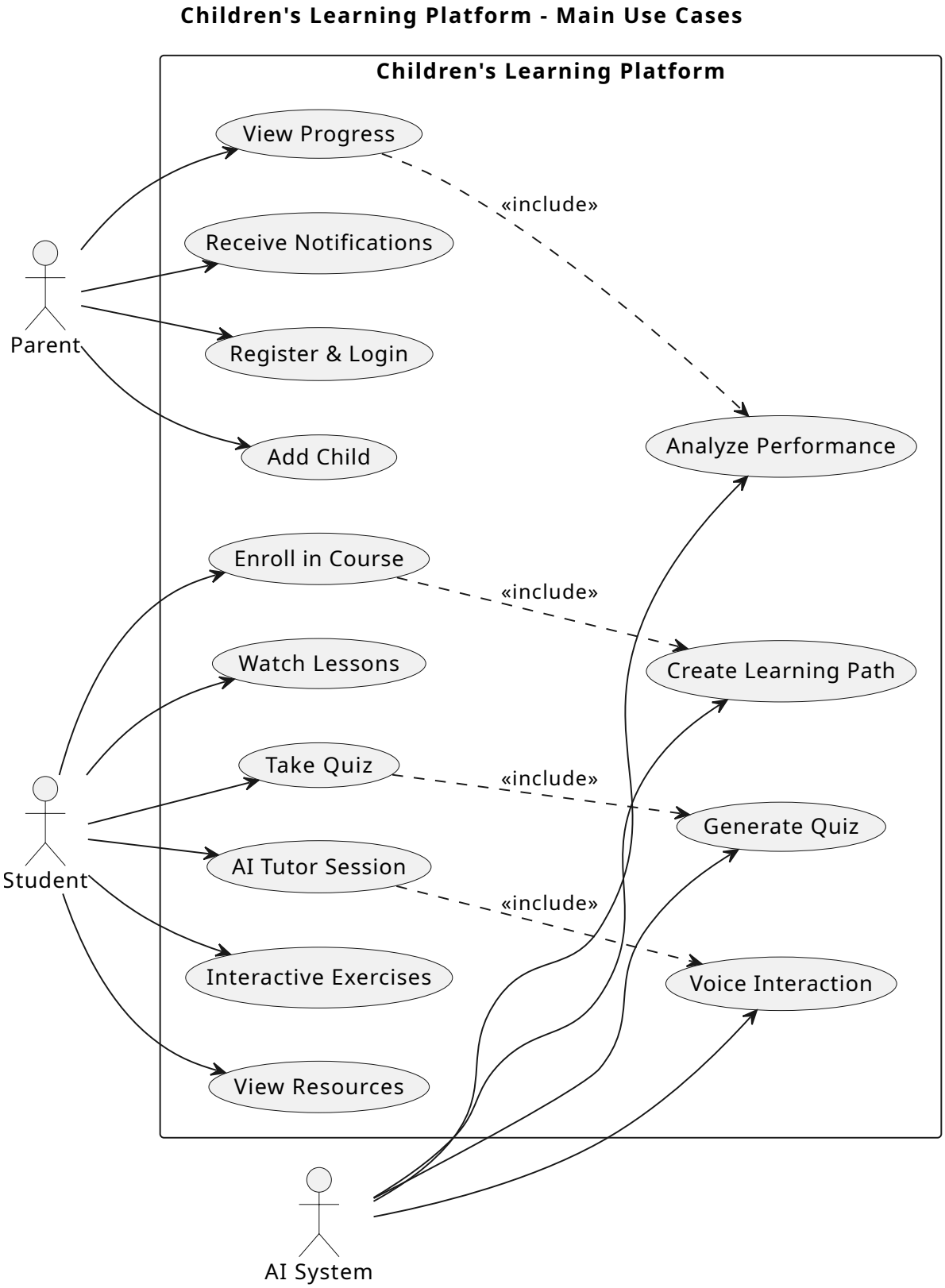
3.2.7 Portability

The complete backend stack shall be containerised using Docker Compose, enabling deployment on any Linux-based VPS or on-premises school server. The Flutter frontend shall produce a single codebase deployable to both Android and iOS platforms. The system shall be designed to tolerate edge deployment scenarios in schools with limited or intermittent internet connectivity.

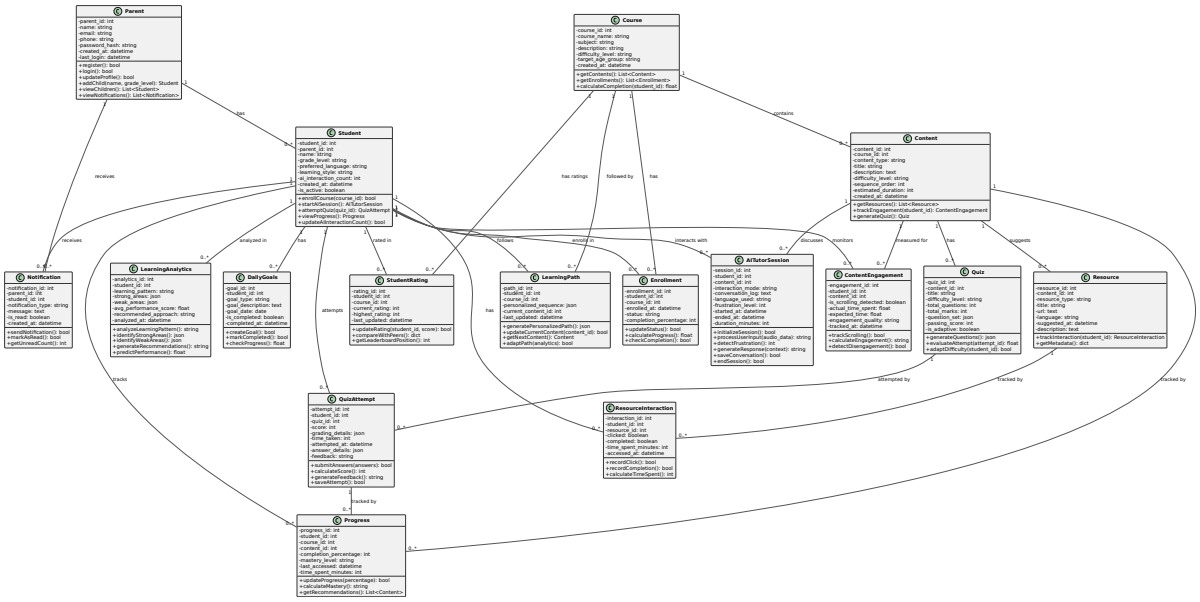
Target Audience	Children primarily interact with low-cost tablets and smartphones.
Justification	Flutter's rendering engine provides smooth animations crucial for maintaining child engagement. Additionally, a single codebase allows deployment to both Android and iOS without separate development teams.

4 Formal use-case and interaction diagrams

4.1 Use-Case Diagram

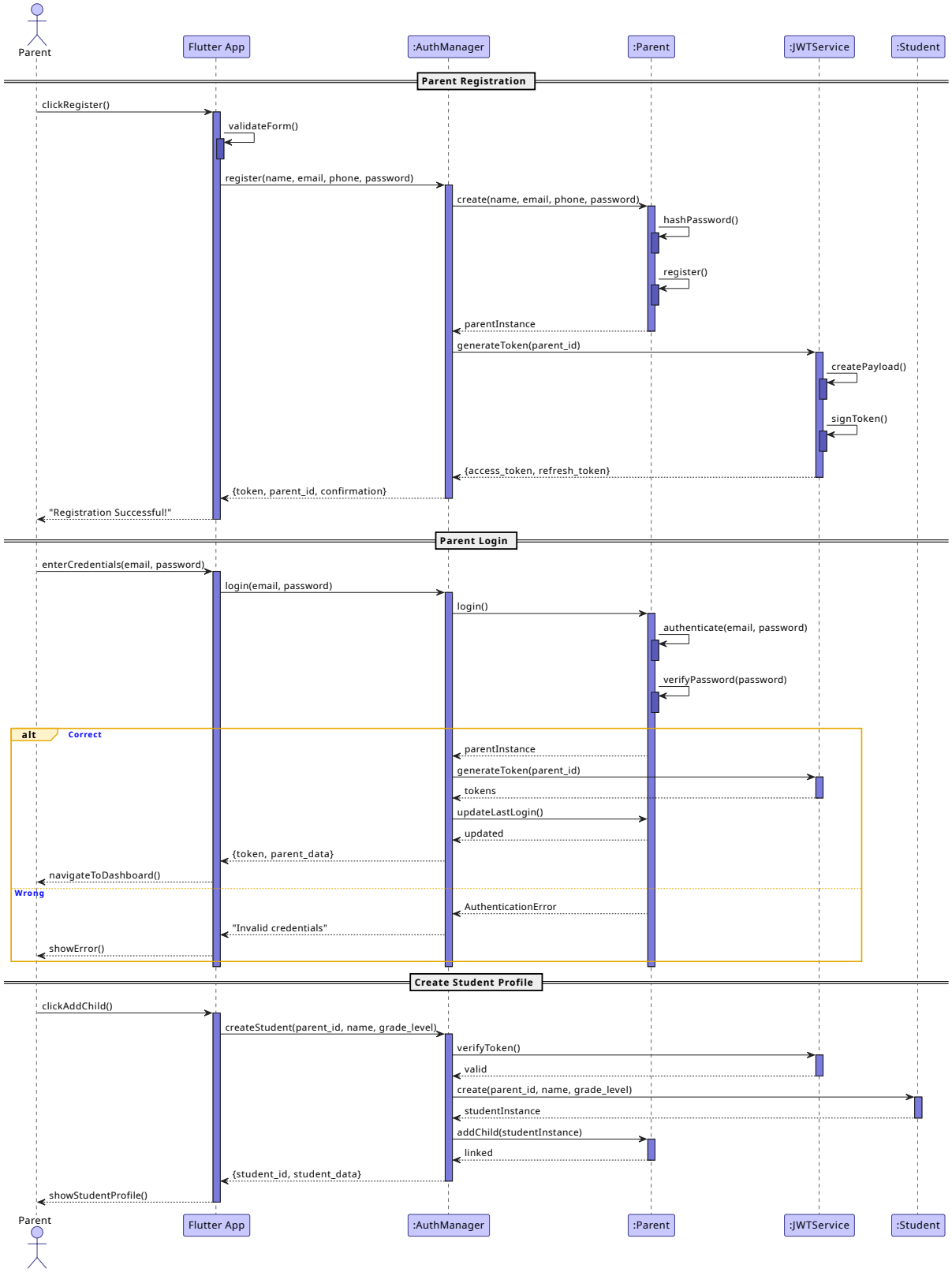


4.2 Class Diagram

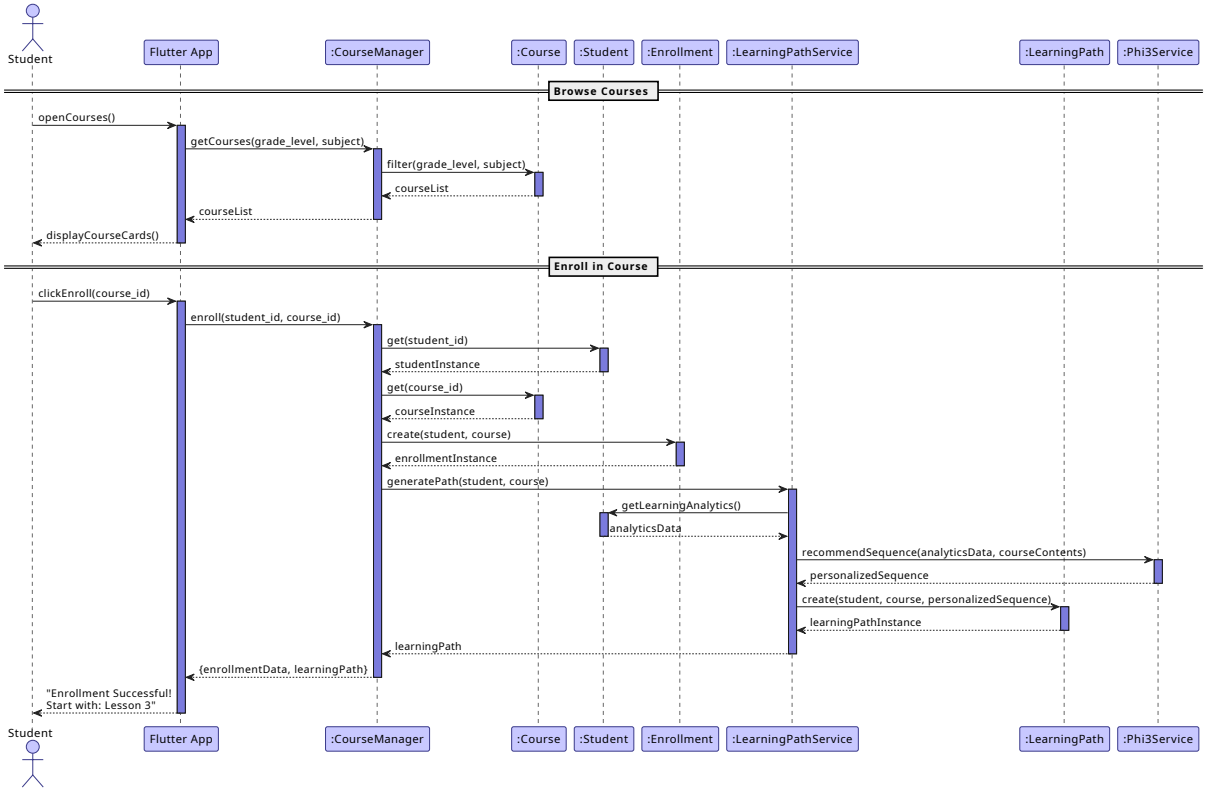


4.3 Sequence Diagrams

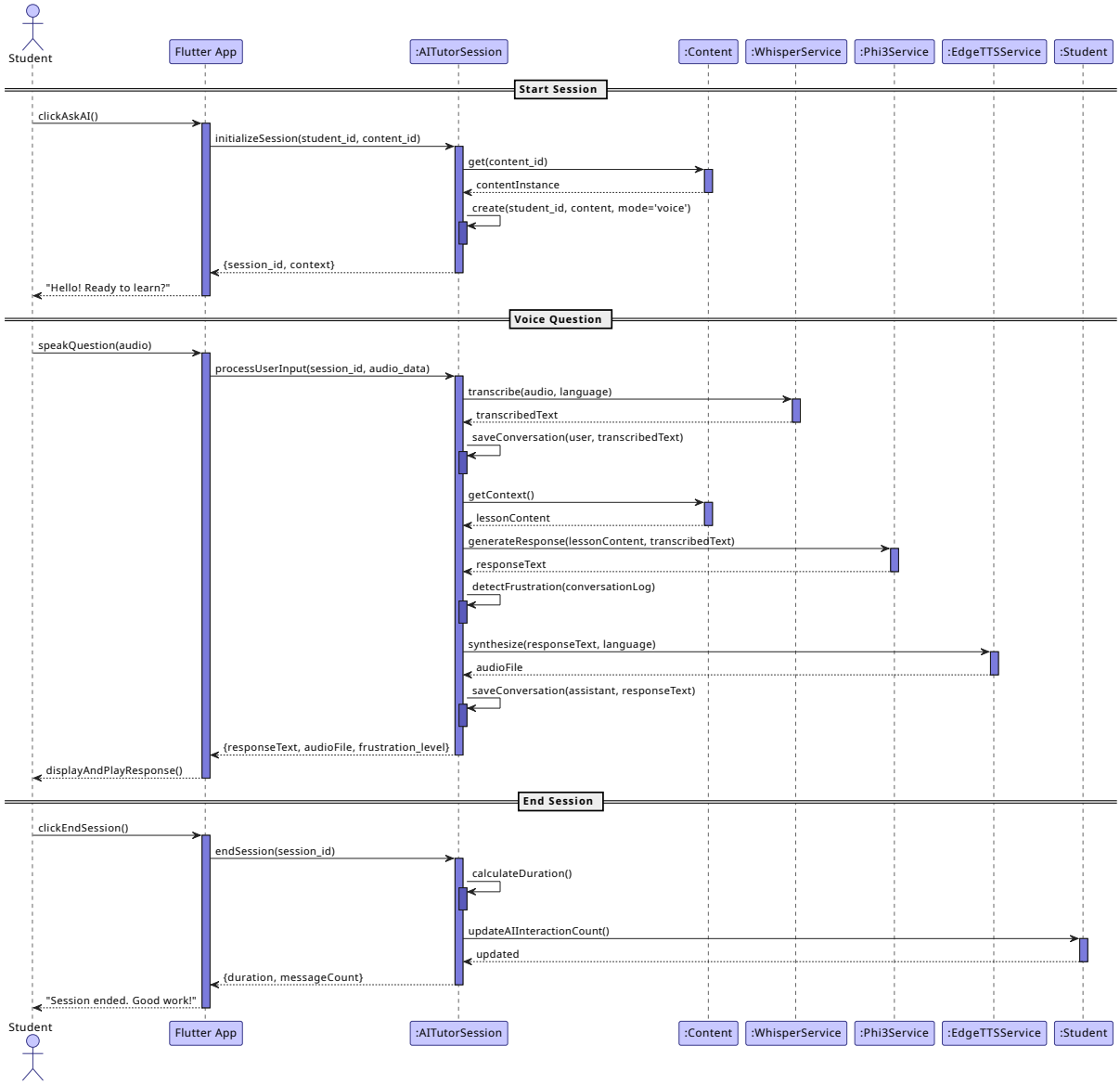
4.3.1 Authentication Sequence Diagram



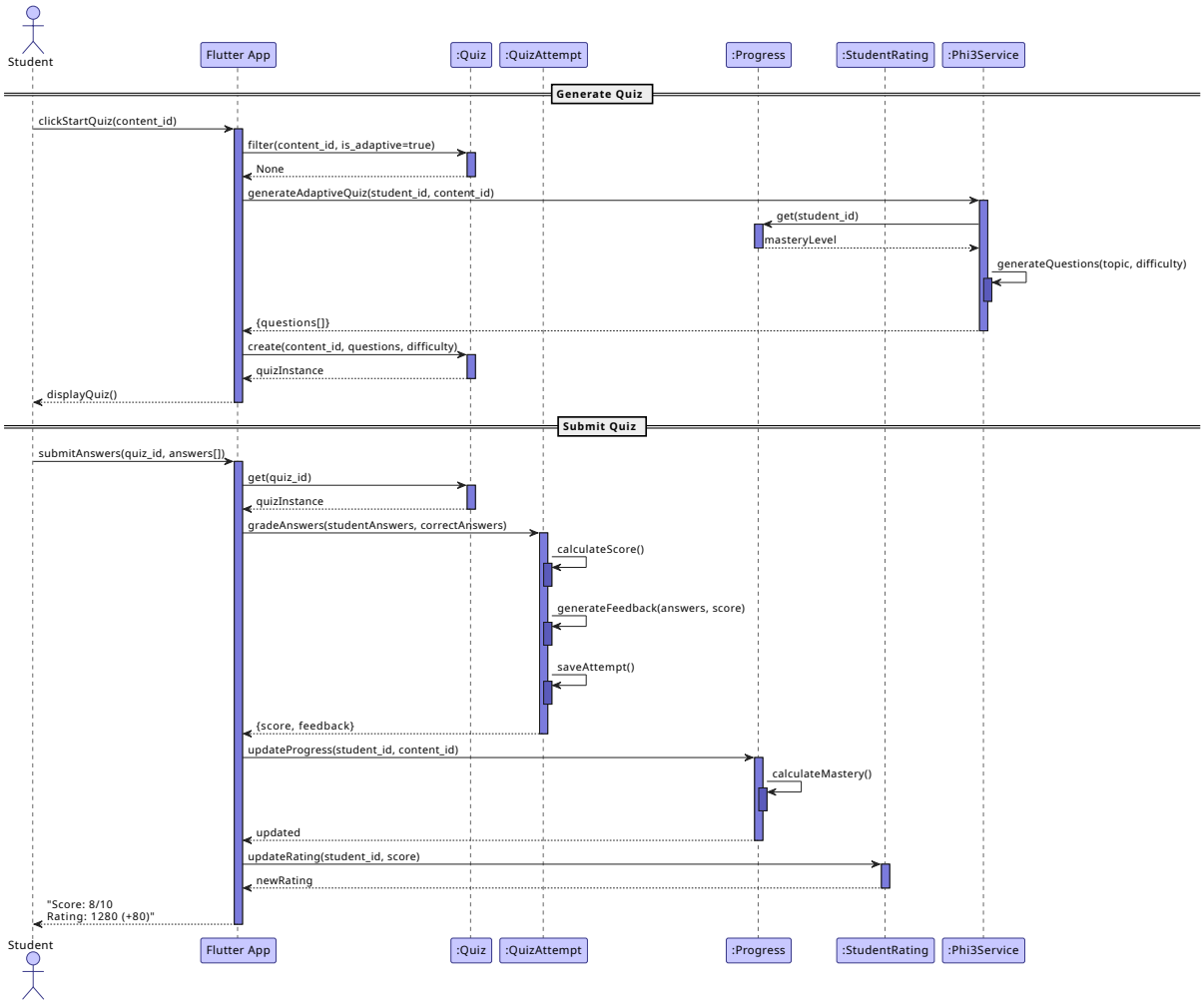
4.3.2 Enrollment Sequence Diagram



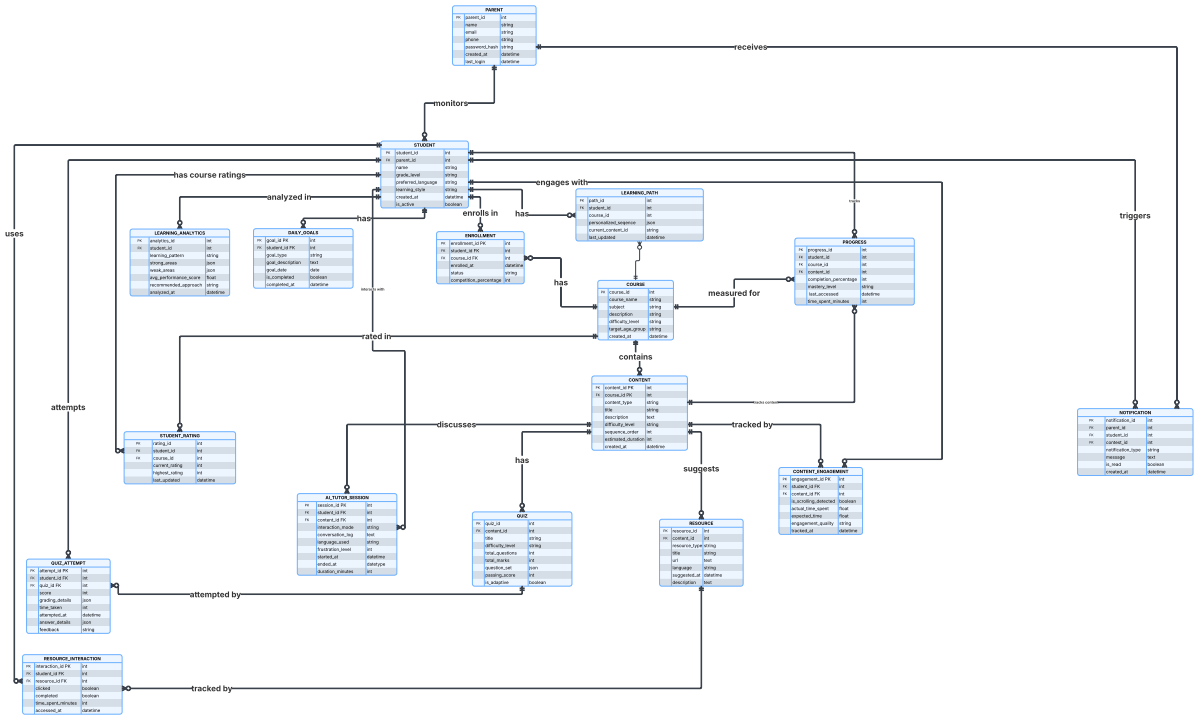
4.3.3 AI Tutor Sequence Diagram



4.3.4 Quiz Sequence Diagram



4.4 ER Diagram



5 Description of Specific Software Modules with Their I/O

The system is composed of six distinct software modules, each handling a specific stage of the pipeline. Each module is independently testable and communicates through well-defined input/output contracts.

5.1 Module 1: OCR and Knowledge Base Builder

Module 1 — OCR & Knowledge Base Builder (Notebook 1)

This offline module is executed once to process the NCTB Class 3 Bengali textbook and build the searchable vector knowledge base. It forms the foundation for all downstream retrieval.

Table 3: Module 1: OCR & Knowledge Base Builder — I/O Specification

Attribute	Details
Input	Raw PDF file: `class3_bangla.pdf` (107 pages, scanned image-based)
Sub-processes	1. **PDF-to-Image Conversion:** `pdftoppm` converts each PDF page to PNG images at DPI=250 for optimal OCR quality. 2. **Easy OCR:** Each PNG image is sent to the Easy OCR. The model extracts Bengali Unicode text. Crash-resume support skips already-processed pages. 3. **Text Chunking:** All extracted page texts are concatenated and segmented into overlapping chunks of 400 characters each. 4. **Embedding Generation:** The `paraphrase-multilingual-MiniLM-L12-v2` model encodes each chunk into a 384-dimensional dense vector. 5. **FAISS Indexing:** All vectors are stored in a FAISS flat index for fast similarity search. 6. **Quality Validation:** Bengali character ratio check and test retrieval queries validate extraction quality.
Output Files	• `class3_bangla_v2.txt` — Clean full-book Bengali Unicode text • `faiss_index/bangla_class3.faiss` — FAISS vector search index • `faiss_index/chunks.pkl` — Serialized text chunk list (500+ entries) • `texts_v2/page_XXX.txt` — Per-page individual text backup files
Execution Mode	Offline / One-time batch processing

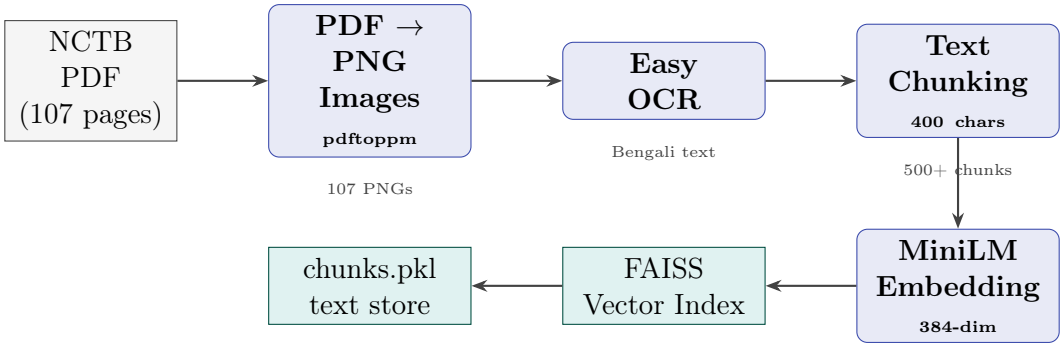


Figure 1: Module 1 Internal Processing Flow

5.2 Module 2: RAG Query and Answer Generation

Module 2 — RAG Q&A Engine (Notebook 2)

This is the core intelligence module. It retrieves relevant textbook passages using FAISS and generates grounded Bengali answers using the Phi-3.5 Mini language model. Operates in real-time during student interactions.

Table 4: Module 2: RAG Q&A Engine — I/O Specification

Attribute	Details
Inputs	• Student question text (Bengali or English string) • `bangla_class3.faiss` (pre-built vector index) • `chunks.pkl` (text chunk store) • Mode flag: `chat` | `quiz` | `explain`
Sub-processes	1. **Query Embedding:** The student's question is encoded using `paraphrase-multilingual-MiniLM` into a 384-dim vector. 2. **FAISS Similarity Search:** The query vector is compared against all stored chunk vectors; top-3 most similar chunks are retrieved (cosine similarity). 3. **Prompt Construction:** Retrieved chunks are combined with the student question into a structured prompt in Bengali instructional format. 4. **Phi-3.5 Mini Inference:** The prompt is passed to `microsoft/Phi-3.5-mini-instruct` loaded with 4-bit NF4 quantization. The model generates a Bengali answer grounded in retrieved context. 5. **MCQ Generation (Quiz Mode):** Given a topic, the model generates 3 MCQ questions in JSON format (4 options + correct answer index).
Outputs	Bengali answer string (plain text)

5.3 Module 3: Speech-to-Text (STT) Engine

Table 5: Module 3: Whisper STT — I/O Specification

Attribute	Details
Input	Audio file (`.wav` format), sampled at 16kHz from Flutter microphone recording
Processing	OpenAI Whisper Medium model processes the audio. Auto-language detection distinguishes Bengali vs English input. Outputs word-level timestamps for verification.
Output	Transcribed text string (Bengali Unicode or English ASCII) + detected language code (`bn` or `en`)
Model	`openai/whisper-medium` (769M parameters)

5.4 Module 4: Text-to-Speech (TTS) Engine

Table 6: Module 4: Edge-TTS — I/O Specification

Attribute	Details
Input	Bengali answer text string (UTF-8 encoded) + voice selection parameter
Processing	Microsoft Edge-TTS API converts text to natural-sounding Bengali speech. Two neural voice options are supported: `bn-BD-NabanitaNeural` (female) and `bn-BD-PradeepNeural` (male).
Output	Audio file (`.mp3`) of spoken Bengali answer; also returned as `base64`-encoded bytes in the API response

5.5 Module 5: Django REST API Backend

Table 7: Module 5: Django Backend — I/O Specification

Attribute	Details
Endpoints	• `POST /api/voice/` — Full voice pipeline (audio in, JSON out) • `POST /api/quiz/` — Quiz generation (topic text in, JSON MCQ out) • `GET /api/history/` — Retrieve session Q&A history
Input (Voice)	Multipart form-data: `audio_file` (.wav binary) + `session_id`
Processing	Orchestrates the complete pipeline: calls STT → RAG → TTS modules in sequence; measures per-stage timing; logs session to PostgreSQL.
Output (JSON)	`{"success": true, "question": "<Whisper transcription>", "answer_text": "<Phi-3.5 Bengali answer>", "answer_audio": "<base64-encoded mp3>", "language": "bengali", "timings": {"stt": 2.3, "retrieval": 0.1, "llm": 7.8, "tts": 1.5}}`
Framework	Django 4.x + Django REST Framework

5.6 Module 6: Flutter Mobile Application

Table 8: Module 6: Flutter Frontend — I/O Specification

Attribute	Details
Input	Microphone audio stream from device; Touch events for quiz interaction
Processing	Records audio at 16kHz, sends to Django REST API, receives JSON response, decodes base64 audio and plays it, renders text answer and quiz UI.
Output	Voice playback of Bengali answer; On-screen text display; Interactive MCQ quiz interface with instant feedback
Platform	Android / iOS (Flutter cross-platform)

6 Interaction Between Different Technology Layers

6.1 Layered Architecture Overview

The system is organized into four horizontal technology layers. Each layer has a distinct responsibility and communicates with adjacent layers through defined interfaces.

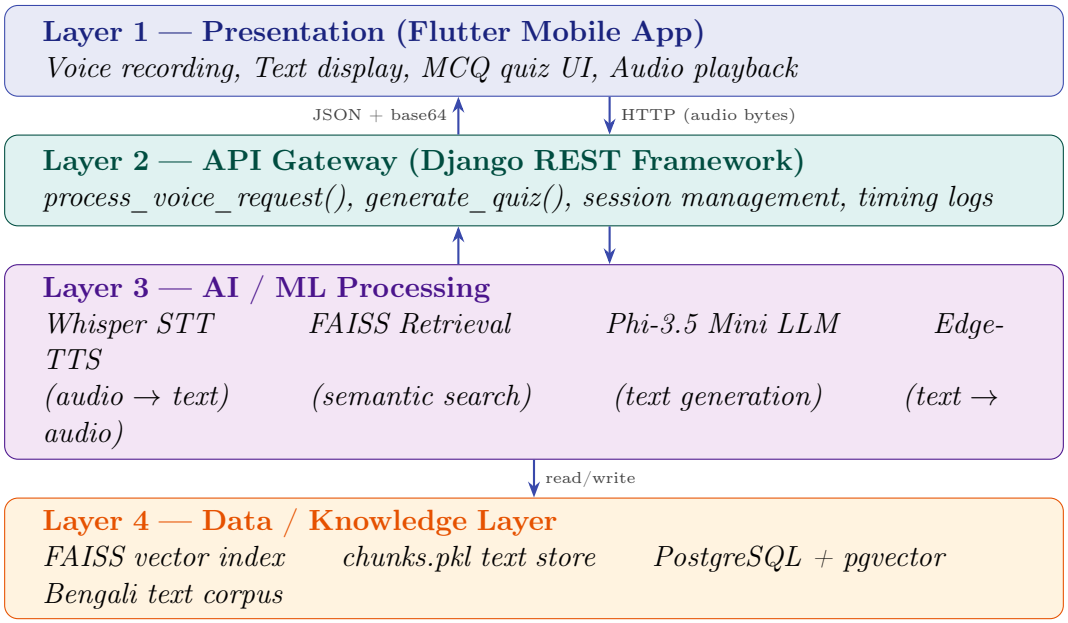


Figure 2: Four-Layer Technology Architecture of the AI Tutor System

6.2 Layer-by-Layer Communication Protocols

6.2.1 Layer 1 ↔ Layer 2: Flutter ↔ Django

Communication between the mobile frontend and the backend API occurs over HTTP/HTTPS using REST conventions.

- **Flutter → Django (Request):** Multipart POST request with `Content-Type: multipart/form-data`. Payload includes the raw `.wav` audio blob recorded by Flutter's microphone plugin at 16kHz mono.
- **Django → Flutter (Response):** JSON object with five fields: `success`, `question` (transcribed text), `answer_text` (Bengali string), `answer_audio` (base64-encoded MP3), and `timings` (stage-wise latency map).
- **Error Handling:** On any pipeline stage failure, Django returns `{"success": false, "error": "<stage>:<message>"}`.

6.2.2 Layer 2 ↔ Layer 3: Django ↔ AI Models

Django orchestrates all AI/ML modules through in-process Python function calls (not separate microservices in the prototype):

1. **STT call:** `whisper_model.transcribe(audio_path)` → returns dict with `text` and `language`.

2. **Retrieval call:** `faiss_index.search(query_embedding, k=3)` → returns indices and distances.
3. **LLM call:** Tokenizer + model pipeline with constructed prompt → returns generated token sequence decoded to Bengali string.
4. **TTS call:** `edge_tts.Communicate(text, voice).save(path)` → writes MP3 file.

6.2.3 Layer 3 ↔ Layer 4: AI Models ↔ Data Layer

- **FAISS index read:** At server startup, `faiss.read_index('bangla_class3.faiss')` loads the pre-built vector index into GPU/RAM. Each query performs an L_2 distance search across all 384-dimensional vectors.
- **Chunk retrieval:** Retrieved FAISS indices are used to look up corresponding text in the `chunks.pkl` list (a Python list of strings loaded via `pickle.load()`).

7 Detailed Data Flow Diagrams and Database Schema

7.1 Context Diagram (Level 0 DFD)

The Level 0 DFD shows the system as a single process interacting with all external entities.

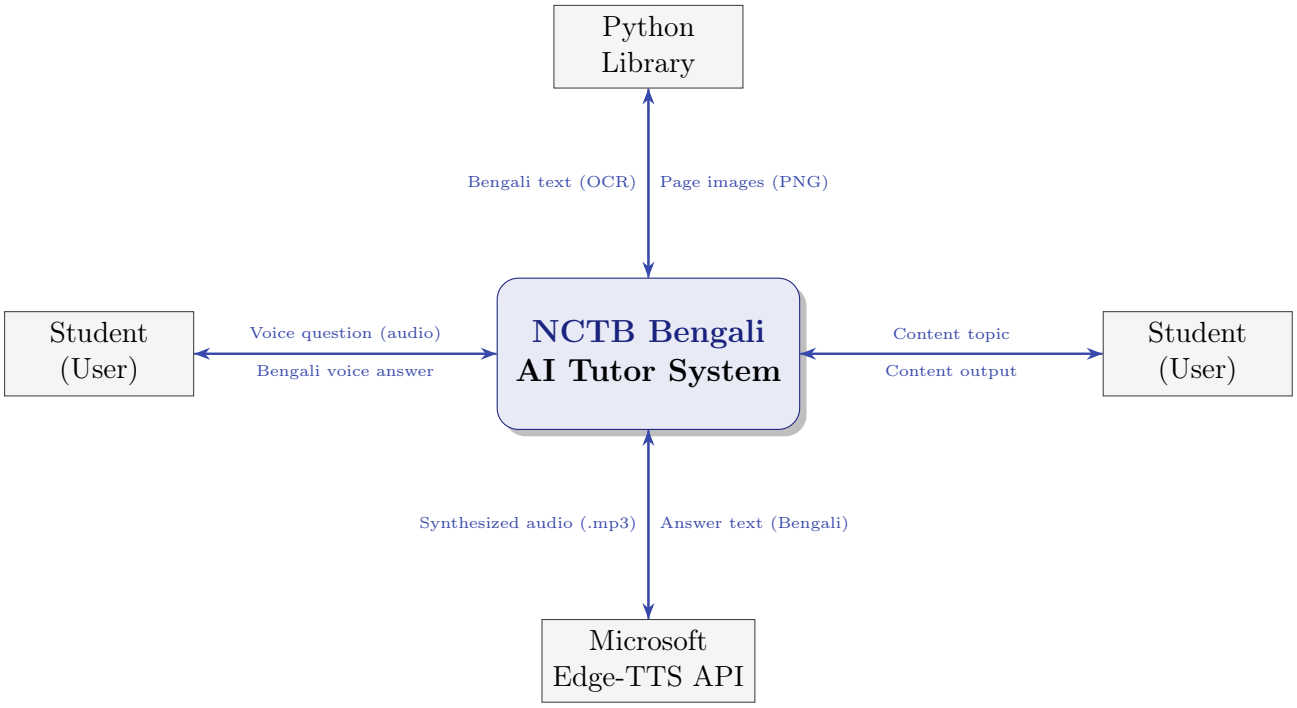


Figure 3: Level 0 DFD — System Context Diagram

7.2 Level 1 DFD — Real-Time Voice Pipeline

This diagram expands the core system into its primary functional processes during live student interaction.

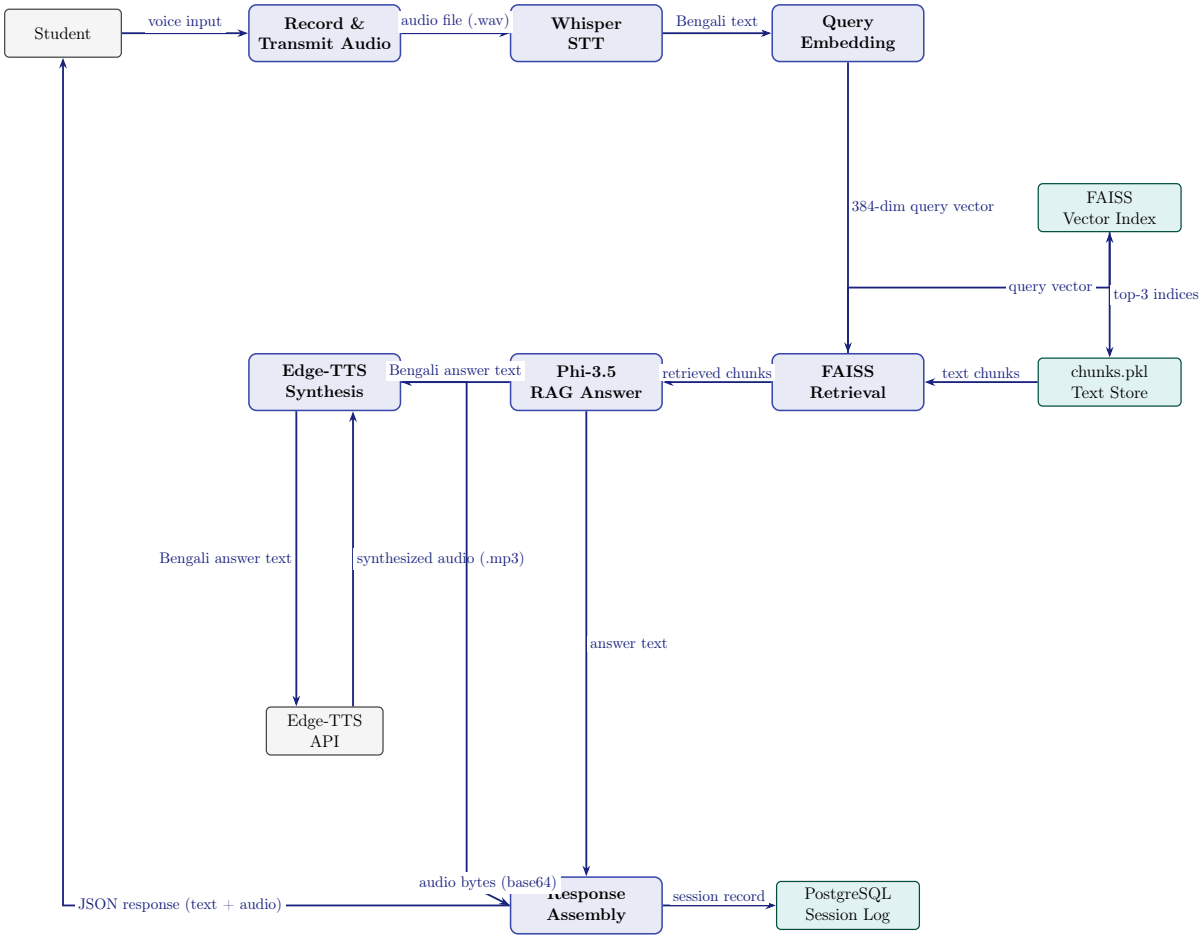


Figure 4: Level 1 DFD — Real-Time Voice Interaction Pipeline

7.3 Level 1 DFD — Offline Knowledge Base Construction

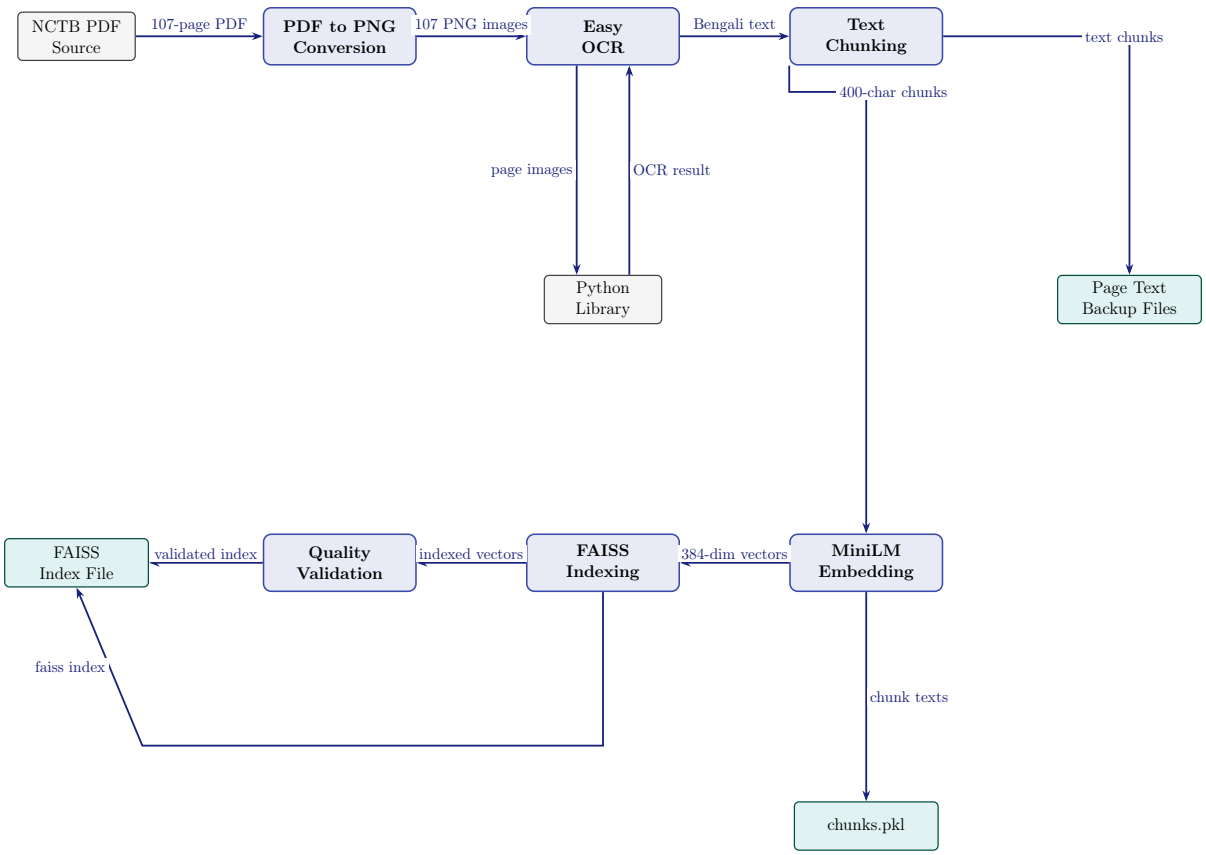


Figure 5: Level 1 DFD — Offline Knowledge Base Construction Pipeline

7.4 Level 2 DFD — RAG Answer Generation (Expanded P5)

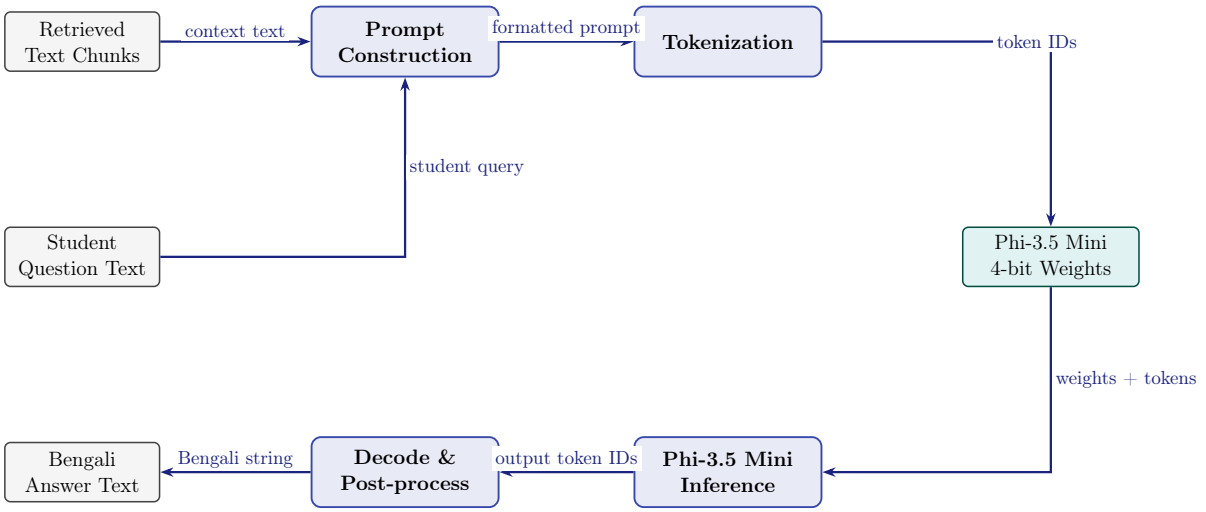


Figure 6: Level 2 DFD — Phi-3.5 RAG Answer Generation (Expanded)

7.5 Database Schema

7.5.1 Overview

The production database uses PostgreSQL with the `pgvector` extension. The schema comprises **17 tables** organized into four functional groups: *User Management* (STUDENT, PARENT), *Academic Content* (COURSE, CONTENT, RESOURCE, QUIZ), *Learning Activity* (AI_TUTOR_SESSION, QUIZ_ATTEMPT, RESOURCE_INTERACTION, CONTENT_ENGAGEMENT, ENROLLMENT, PROGRESS), and *Intelligence & Personalization* (LEARNING_ANALYTICS, LEARNING_PATH, DAILY_GOALS, STUDENT_RATING, NOTIFICATION). Together they support end-to-end tracking of student learning journeys, AI interactions, and adaptive recommendations.

7.5.2 Group 1: User Management Tables

Table 9: Table: `STUDENT`

Column	Type	Key	Description
`student_id`	`INT`	PK	Unique student identifier
`parent_id`	`INT`	FK → PARENT	Linked parent/guardian
`name`	`STRING`	NOT NULL	Full name of the student
`grade_level`	`STRING`		Current class/grade (e.g., "Class 3")
`preferred_language`	`STRING`		Bengali or English preference
`learning_style`	`STRING`		e.g., visual, auditory, reading
`created_at`	`DATETIME`	NOT NULL	Account registration timestamp
`is_active`	`BOOLEAN`	DEFAULT TRUE	Whether account is currently active

Table 10: Table: `PARENT`

Column	Type	Key	Description
`parent_id`	`INT`	PK	Unique parent identifier
`name`	`STRING`	NOT NULL	Parent full name
`email`	`STRING`	UNIQUE	Login email address
`phone`	`STRING`		Contact phone number
`password_hash`	`STRING`	NOT NULL	Bcrypt-hashed password
`created_at`	`DATETIME`	NOT NULL	Account creation time
`last_login`	`DATETIME`		Most recent login timestamp

7.5.3 Group 2: Academic Content Tables

Table 11: Table: `COURSE`

Column	Type	Key	Description
course_id	INT	PK	Unique course identifier
course_name	STRING	NOT NULL	e.g., "NCTB Bangla Class 3"
subject	STRING		Subject area (Bangla, Math, etc.)
description	STRING		Short course description
difficulty_level	STRING		Beginner / Intermediate / Advanced
target_age_group	STRING		e.g., "7–9 years"
created_at	DATETIME	NOT NULL	Course creation timestamp

Table 12: Table: `CONTENT`

Column	Type	Key	Description
content_id	INT	PK	Unique content item ID
course_id	INT	FK → COURSE	Owning course
content_type	STRING		lesson / video / exercise / story
title	STRING	NOT NULL	Content title
description	TEXT		Full description
difficulty_level	STRING		Relative difficulty within course
sequence_order	INT		Order within course curriculum
estimated_duration	INT		Expected time in minutes
created_at	DATETIME	NOT NULL	Creation timestamp

Table 13: Table: `RESOURCE`

Column	Type	Key	Description
`resource_id`	`INT`	PK	Unique resource identifier
`content_id`	`INT`	FK → CONTENT	Associated content item
`resource_type`	`STRING`		pdf / video / audio / image / link
`title`	`STRING`	NOT NULL	Resource display title
`url`	`TEXT`	NOT NULL	Accessible URL or file path
`language`	`STRING`		Bengali or English
`suggested_at`	`DATETIME`		When AI suggested this resource
`description`	`TEXT`		Resource summary

Table 14: Table: `QUIZ`

Column	Type	Key	Description
`quiz_id`	`INT`	PK	Unique quiz identifier
`content_id`	`INT`	FK → CONTENT	Content this quiz covers
`title`	`STRING`	NOT NULL	Quiz title
`difficulty_level`	`STRING`		easy / medium / hard
`total_questions`	`INT`	NOT NULL	Number of MCQ items
`total_marks`	`INT`	NOT NULL	Maximum achievable score
`question_set`	`JSON`	NOT NULL	Array of `{question, options[4], answer_index}` objects
`passing_score`	`INT`		Minimum score to pass
`is_adaptive`	`BOOLEAN`	DEFAULT FALSE	Whether quiz adapts to student level

7.5.4 Group 3: Learning Activity Tables

Table 15: Table: `AI_TUTOR_SESSION`

Column	Type	Key	Description
`session_id`	`INT`	PK	Unique session identifier
`student_id`	`INT`	FK → STUDENT	Student who initiated session
`content_id`	`INT`	FK → CONTENT	Content being discussed
`interaction_mode`	`STRING`		voice / text / quiz / explain
`conversation_log`	`TEXT`		Full Q&A turn history (JSON string)
`language_used`	`STRING`		`bn` (Bengali) or `en` (English)
`frustration_level`	`INT`		0–10 scale inferred by AI
`started_at`	`DATETIME`	NOT NULL	Session start timestamp
`ended_at`	`DATETIME`		Session end timestamp
`duration_minutes`	`INT`		Computed session length

Table 16: Table: `QUIZ_ATTEMPT`

Column	Type	Key	Description
`attempt_id`	`INT`	PK	Unique attempt identifier
`student_id`	`INT`	FK → STUDENT	Attempting student
`quiz_id`	`INT`	FK → QUIZ	Quiz being attempted
`score`	`INT`		Raw score achieved
`grading_details`	`JSON`		Per-question correct/wrong breakdown
`time_taken`	`INT`		Total time in seconds
`attempted_at`	`DATETIME`	NOT NULL	Attempt start timestamp
`answer_details`	`JSON`		Student's selected option per question
`feedback`	`STRING`		AI-generated post-quiz feedback text

Table 17: Table: ENROLLMENT

Column	Type	Key	Description
enrollment_id	INT	PK	Unique enrollment record
student_id	INT	FK → STUDENT	Enrolled student
course_id	INT	FK → COURSE	Enrolled course
enrolled_at	DATETIME	NOT NULL	Date of enrollment
status	STRING		active / completed / paused
completion_percentage	INT	DEFAULT 0	Overall course completion (0–100)

Table 18: Table: PROGRESS

Column	Type	Key	Description
progress_id	INT	PK	Unique progress record
student_id	INT	FK → STUDENT	Student being tracked
course_id	INT	FK → COURSE	Relevant course
content_id	INT	FK → CONTENT	Specific content item
completion_percentage	INT		Content completion 0–100
mastery_level	STRING		beginner / developing / proficient / mastered
last_accessed	DATETIME		Most recent access timestamp
time_spent_minutes	INT		Cumulative time on this content

Table 19: Table: RESOURCE_INTERACTION

Column	Type	Key	Description
interaction_id	INT	PK	Unique interaction record
student_id	INT	FK → STUDENT	Interacting student
resource_id	INT	FK → RESOURCE	Accessed resource
clicked	BOOLEAN		Whether resource link was clicked
completed	BOOLEAN		Whether resource was fully consumed
time_spent_minutes	INT		Time spent on this resource
accessed_at	DATETIME	NOT NULL	Timestamp of interaction

Table 20: Table: CONTENT_ENGAGEMENT

Column	Type	Key	Description
engagement_id	INT	PK	Unique engagement event
student_id	INT	FK → STUDENT	Engaged student
content_id	INT	FK → CONTENT	Engaged content item
is_scrolling_detected	BOOLEAN		Whether scrolling behavior detected
actual_time_spent	FLOAT		Measured active time (minutes)
expected_time	FLOAT		Expected time for this content
engagement_quality	STRING		low / medium / high
tracked_at	DATETIME	NOT NULL	Tracking event timestamp

7.5.5 Group 4: Intelligence & Personalization Tables

Table 21: Table: `LEARNING_ANALYTICS`

Column	Type	Key	Description
analytics_id	INT	PK	Unique analytics record
student_id	INT	FK → STUDENT	Student being analyzed
learning_pattern	STRING		Detected pattern (e.g., "visual learner")
strong_areas	JSON		Array of topics where student excels
weak_areas	JSON		Array of topics needing improvement
avg_performance_score	FLOAT		Rolling average quiz score
recommended_approach	STRING		AI-generated learning strategy
analyzed_at	DATETIME	NOT NULL	Analysis generation timestamp

Table 22: Table: `LEARNING_PATH`

Column	Type	Key	Description
path_id	INT	PK	Unique path record
student_id	INT	FK → STUDENT	Owning student
course_id	INT	FK → COURSE	Associated course
personalized_sequence	JSON		Ordered list of `content_id`s for this student
current_content_id	STRING		ID of content currently in progress
last_updated	DATETIME		When path was last re-generated

Table 23: Table: `DAILY_GOALS`

Column	Type	Key	Description
goal_id	INT	PK	Unique daily goal record
student_id	INT	FK → STUDENT	Owning student
goal_type	STRING		reading / quiz / session / vocabulary
goal_description	TEXT		Human-readable goal description
goal_date	DATE	NOT NULL	Date this goal applies to
is_completed	BOOLEAN	DEFAULT FALSE	Whether goal was achieved
completed_at	DATETIME		Completion timestamp if achieved

Table 24: Table: `STUDENT_RATING`

Column	Type	Key	Description
`rating_id`	`INT`	PK	Unique rating record
`student_id`	`INT`	FK → STUDENT	Rated student
`course_id`	`INT`	FK → COURSE	Course being rated in
`current_rating`	`INT`		Current ELO-style performance rating
`highest_rating`	`INT`		All-time peak rating
`last_updated`	`DATETIME`		Rating update timestamp

Table 25: Table: `NOTIFICATION`

Column	Type	Key	Description
`notification_id`	`INT`	PK	Unique notification record
`parent_id`	`INT`	FK → PARENT	Recipient parent
`student_id`	`INT`	FK → STUDENT	Related student
`contest_id`	`INT`	(optional)	Related contest if applicable
`notification_type`	`STRING`		progress / alert / quiz_result / goal
`message`	`TEXT`	NOT NULL	Notification message body
`is_read`	`BOOLEAN`	DEFAULT FALSE	Whether parent has read this
`created_at`	`DATETIME`	NOT NULL	Notification creation timestamp

7.5.6 FAISS Vector Store Schema (Non-Relational)

The FAISS index operates as a file-based vector store alongside the relational database. Its logical structure is:

Table 26: FAISS Vector Store — Logical Schema

Component	Format	Description
`bangla_class3.faiss`	Binary index file	FAISS `IndexFlatL2` — 500+ vectors of dimension 384
`chunks.pkl`	Python pickle list	Parallel Bengali text strings; `chunks[i]` ↔ vector `i`
Vector dimension	384 float32	Output of `paraphrase-multilingual-MiniLM-L12-v2`
Search algorithm	Exact L_2 (flat)	Brute-force search; sufficient for <1000 vectors
Chunk size	≈400 characters	Contiguous passage from NCTB textbook