AI Coach - NEET Preparation System Documentation

Overview

The AI Coach system generates personalized study plans for students preparing for NEET exams. It:

1. **Analyzes a student's recent performance and mastery of subjects/chapters.**
2. **Uses LLM (Gemini) to generate a custom 7-day plan and recommendations.**
3. **Determines whether the student is ready to take a new test.**

---

1. Project Structure

/routes

└── coach.routes.js # API route definitions

/controllers

└── coach.controller.js # Core logic for handling API requests

/services

├── coachInput.js # Prepares data (inputs) for the coach

├── coachPlan.service.js # Generates and saves AI-driven study plans

└── gating.js # Logic to decide if a student can start a test

/models

└── studentAnalytics.model.js# Sequelize model (DB table for student analytics)

---

2. API Routes (`routes/coach.routes.js`)

HTTP Method	Endpoint	Controller Function	Description
`GET`	`/api/ai/coach/plan?studentId=154`	`getCoachPlan`	Fetches the student's current AI plan. Generates a new plan if none exists.
`POST`	`/api/ai/coach/refresh`	`refreshCoachPlan`	Forces regeneration of a new AI plan.

GET	/api/tests/can-start?studentId=154&type=full	canStartTest	Checks if the student is allowed to start a test, based on performance gates.

---

3. Controller Layer (controllers/coach.controller.js)

This is the business logic layer connecting routes and services.

Helper: resolveStudentId(req)

- **Extracts studentId from:**
  - **req.user.id (preferred, if authentication middleware is used)**
  - **req.query.studentId or req.body.studentId (fallback for Postman testing)**

Functions

getCoachPlan(req, res)

- **Purpose: Fetch the student's AI plan or generate a new one if it doesn't exist.**
- **Flow:**
  1. **Get studentId via resolveStudentId.**
  2. **Look up the student in StudentAnalytics.**
  3. **If no ai_plan exists → call generateAndSaveCoachPlan().**

4. **Return the plan with a flag:**
    - `generated: true` → **new plan created.**
    - `generated: false` → **existing plan returned.**

`refreshCoachPlan(req, res)`

- **Purpose: Force regeneration of a new AI plan.**
- **Flow:**
    1. **Get `studentId`.**
    2. **Call `generateAndSaveCoachPlan()` directly.**
    3. **Always return a fresh plan.**

`canStartTest(req, res)`

- **Purpose: Determine if a student is ready to start a test.**
- **Flow:**
    1. **Get `studentId` and test `type` (default `"full"`).**
    2. **Build analytics input via `buildCoachInput()`.**
    3. **Evaluate eligibility using `evaluateGate()`.**

4. If not allowed, return recommended remedial tasks from `ai_plan`.
5. If allowed, return `allowed: true` and empty task list.

---

4. Services

4.1 coachInput.js

Purpose: Build a structured input for the AI coach and test gate evaluation.

Key Steps

1. Fetch the student's analytics data.
2. Parse DB JSON fields safely (`subject_mastery`, `chapter_mastery`, `recent_windows`).
3. Identify weak chapters:
   - Chapters with at least 2 attempts but low scores.
   - Sort by score (lowest first).
   - Select top 2 weak chapters per subject.
4. Return a comprehensive input object.
5.

Example Output:


```
{
  "studentId": 154,
  "recent_windows": { "overall_last5": 42, "Physics_last5": 40, ... },
  "subject_mastery": { "Physics": { "score": 45 }, ... },
  "chapter_mastery": { ... },
  "focus_suggestion": [
    {
      "subject": "Physics",
      "chapters": [
        { "name": "Optics", "why": "low score & sufficient attempts", "score": 30 }
      ]
    }
  ],
  "targets": { "overall": 45, "subject": 50 }
}
```

4.2 coachPlan.service.js

Purpose: Generate and store a personalized 7-day AI plan.

Main Function: `generateAndSaveCoachPlan(studentId)`

1. **Build input using** `buildCoachInput()`.
2. **Call Gemini LLM (**`getGemini("gemini-1.5-flash")`**) with:**
    - **System Prompt: Explains required JSON structure.**
    - **User Prompt: Student analytics data.**
3. **Parse LLM response into JSON (**`parseJsonLoose`**).**
4. **If LLM fails → fallbackPlan() is used.**
5. **Save the plan to the** `StudentAnalytics` **table.**

Generated Plan Structure:

- `summary`**: High-level performance overview.**
- `focus`**: Weak subjects and chapters to target.**
- `plan`**: 7-day schedule + rules for test readiness.**
- `tasks`**: Specific remedial quizzes or tasks.**
- `tone_coach_notes`**: Motivational tips.**
-

Phone : +91 959 443 0295
Email: nexcorealliance@gmail.co

Fallback Plan Example:


```
{
  "summary": { "overall_last5": 42, ... },
  "focus": [ { "subject": "Physics", "chapters": [ ... ] } ],
  "plan": {
    "next_7_days": [
      { "day": 1, "blocks": [ { "type": "learn", "subject": "Physics", "chapter": "Optics" }
    ] },
      { "day": 2, "blocks": [ { "type": "quiz", "questions": 30 } ] }
    ],
    "test_readiness_rules": [
      { "metric": "overall_last5", "op": ">=", "value": 45 }
    ]
  },
  "tasks": [ { "id": "remedial-physics-01", "type": "remedial_quiz" } ],
  "tone_coach_notes": [
    "Short, daily reps beat long gaps.",
    "Focus accuracy first; speed comes next."
  ]
}
```

4.3 gating.js

Purpose: Determine if a student meets performance criteria to start a test.

Function: `evaluateGate(analytics, rules)`

- **Inputs:**
  - `analytics.recent_windows` → **contains last 5 scores.**
  - `rules` → **thresholds (`overall: 45, subject: 50`).**
- **Logic:**
  - **Student must meet:**
    - **Overall score ≥ 45%.**
    - **Each subject score ≥ 50%.**
  - **If any condition fails → return reasons for rejection.**

Example Output:

```
{
  "allowed": false,
  "reasons": [
    "Raise Overall last-5 to ≥ 45% (current: 42%)",
    "Physics ≥ 50% (now 40%)"
  ],
  "targets": { "overall": 45, "subject": 50 }
}
```

---

5. Database: `StudentAnalytics`

- **Stores analytics and generated AI plan for each student.**

Field	Description
`student_id`	**Unique ID for each student**
`subject_mastery`	**Performance by subject (JSON)**
`chapter_mastery`	**Performance by chapter (JSON)**
`recent_windows`	**Recent performance windows (JSON)**
`ai_plan`	**Latest generated AI plan (JSON)**

---

6. System Flow

Fetch or Generate Plan

[Client] → GET /api/ai/coach/plan

↓

coach.controller.getCoachPlan()

↓

StudentAnalytics DB

↓ **(if no plan exists)**

coachPlan.service.generateAndSaveCoachPlan()

↓

Gemini LLM → Custom AI Plan

↓

DB Save → Return JSON plan to client

Check Test Eligibility

[Client] → GET /api/tests/can-start

↓

coach.controller.canStartTest()

↓

coachInput.buildCoachInput()

↓

gating.evaluateGate()

↓

Return { allowed: true/false, reasons, recommended_tasks }

---

7. Key Dependencies

- **Express.js → Routing and HTTP handling.**
- **Sequelize → ORM for database.**
- **Gemini API → AI plan generation.**
- **Node.js environment.**

---

8. Summary

- **Routes Layer handles HTTP endpoints.**
- **Controllers Layer orchestrates requests and responses.**
- **Services Layer contains core logic:**
    - `coachInput.js` **→ Data preparation.**
    - `coachPlan.service.js` **→ AI plan generation.**
    - `gating.js` **→ Eligibility rules.**
- **Database persists analytics and generated plans.**

This document should be sufficient for future developers to understand, maintain, and extend the system.