Programming Task: AI-Driven Learning Platform (Mini MVP)

Role: Full-stack Developer
Submission format: Public GitHub repository with clear documentation and usage instructions.

Objective

Create a mini learning platform that allows users to select what they want to learn (by **category** and **sub_category**), send **prompts** to an AI to receive generated lessons, and view their learning history. The platform will include a REST API backend, database, AI integration, and a basic web dashboard UI.

This task is designed to assess your skills in software architecture, API integration, modular code organization, and delivery quality.

Technical Requirements

Backend

- Language: Node.js, Python (FastAPI), Go, or any backend language/framework you prefer — as long as it's production-grade.
- Database: PostgreSQL, MySQL, or MongoDB (use ORM if applicable)
- A REST API server with structured routes.
  AI Integration:
- Use OpenAI GPT API (or mock a local response with a clear integration interface)
- Send the user's selected topic and prompt, and receive a lesson-like response.
- Models / Tables:
- users: id, name, phone
- categories: id, name
- sub_categories: id, name, category_id
- prompts: id, user_id, category_id, sub_category_id, prompt, response, created_at
- Users can have multiple prompts (learning history)
- Maintain proper database relationships and constraints Frontend
- Language: React, Vue, or any modern frontend framework
- Create a simple web dashboard that supports:
- Registering new users
- Selecting category & sub-category
- Submitting a prompt
- Viewing the AI-generated response
- Viewing the user's learning history
- Admin dashboard to list all users and their prompt history

Mandatory Requirements

- Organized project structure with clearly separated layers (routes/controllers/models/services/etc.)
- Clear README file with:
- Setup instructions
- Technologies used
- Assumptions made
- How to run locally (frontend & backend)
- Sample .env example file
- Use Docker or Docker Compose to spin up the database if needed
- GitHub repository must be public with a clear commit history
- Code should be clean, well-documented, and follow best practices
- Handle basic input validation and API error handling
- Use dotenv or configuration management

Bonus (Optional but Impressive)

- Use of TypeScript in the frontend or backend
- JWT-based user authentication
- Pagination and filtering in the admin dashboard
- Unit/integration tests (using Jest, Pytest, etc.)
- Swagger/OpenAPI documentation.
- Deploy a working demo (e.g., via Vercel, Netlify, or Heroku)

Example Use Case

Israel signs up and selects to learn about Science → Space. He enters a prompt: "Teach me about black holes." The system stores his input, sends it to an AI model, and returns a lesson. He can revisit the dashboard later to view all the lessons he received.

Time Estimate

You can spend up to 2-3 days on this task, depending on your availability. Quality and clarity are more important than completeness. Please focus on architecture, maintainability, and clarity.

Submission

- Upload to a public GitHub repository
- Share the link along with a brief note (1-2 paragraphs) explaining your Approach
- When you are done, please send your task as follows:
  Email subject: first name last name
  Body: GitHub repository https link (make sure it's public)
  Resume: attached Resume in PDF format
  Note Letter: Write a brief about yourself

We're looking for clean and modular code, creative problem-solving, and a sharp architectural mindset. Treat this as a mini production-grade system — the kind of work you would do for our clients.

Good luck!