SDE Intern Project Assignment: Online Learning Platform

1. Project Description

Welcome to the second round of our hiring process. This take-home project is designed to evaluate your practical skills in software development, including your ability to build a full-stack application, design a scalable system, and write clean, maintainable code.

The task is to build a simplified online learning platform. This platform will have two types of users: **Instructors** and **Students**. An instructor can create courses and lectures, which can be either a simple reading or a quiz, while a student can browse available courses, complete lectures, and track their progress.

This project should be a complete, end-to-end application. You are free to use any framework, language, or database technology you are most comfortable with, as long as the project is a full-stack solution (i.e., it must include a frontend, a backend, and a database).

---

2. Functional Requirements

The application must implement the following core functionalities:

User Management & Authentication

- **Sign-Up/Registration:** Users should be able to create an account. The system must support two user roles: "Instructor" and "Student."
- **Login:** Users should be able to log in to the application.
- **Authentication & Authorization:** The application must protect its API endpoints. Access to certain functionalities should be restricted based on the user's role (e.g., only instructors can create courses).

Instructor Functionality

- **Course Creation:** An authenticated instructor must be able to create a new course with a title and a description.
- **Lecture Management:** For a course they own, an instructor must be able to add new lectures. A lecture can be of two types:
    - **Reading:** A lecture with a title and simple text or a link as content.
    - **Quiz:** A quiz with a title and one or more multiple-choice questions. Each question must have question text, a list of potential answers, and an indication of which answer is correct.

Student Functionality

- **Course Browsing:** Students should be able to view a list of all available courses on the platform. The list should display the course title and description.
- **Lecture Viewing:** A student must be able to view a lecture and its content. The frontend should display lectures sequentially, allowing the student to navigate to the next lecture only after completing the current one.
- **Progress Tracking:** The system must track a student's progress and display it (e.g., "5/10 lectures completed"). A lecture is considered complete under these conditions:
    - **Reading Lecture:** When the student views it.
    - **Quiz Lecture:** When the student successfully submits it with a passing grade (Optional) (e.g., at least 70% correct answers).
- **Quiz Attempt & Grading:** A student can attempt a quiz. When the student submits the quiz, the backend must **grade it in real-time** and determine if the student passed. The student should see their score on the frontend immediately.

---

3. Technical Requirements & Criteria

Your project will be evaluated based on the following criteria. Please keep these in mind as you design and implement your solution.

Backend

- **RESTful API Design:** The API should be well-structured, easy to understand, and follow REST principles. Use meaningful endpoints and HTTP methods.
- **Data Modeling:** The database schema should be well-designed, capturing the relationships between courses, lectures, users, and student progress. The schema must also support the two types of lectures (reading and quiz) and the associated questions/answers.
- **Authentication & Authorization:** The implementation of user authentication and role-based authorization should be secure and correctly implemented.
- **Logic & Algorithms:** The backend must contain the core logic for quiz grading and for managing student progress. This logic should be efficient and correctly calculate scores and completion status.
- **Error Handling:** The API should return appropriate HTTP status codes and clear error messages for invalid requests or unauthorized access.

Frontend

- **User Interface:** The UI should be functional, intuitive, and easy to navigate for both instructors and students.
- **State Management:** Demonstrate a clear and efficient way to manage application state (e.g., user authentication status, course data, student progress).
- **API Integration:** The frontend should communicate with the backend API to fetch and submit data, handling different data structures for reading and quiz lectures.

General

- **Code Quality:** The code should be clean, well-organized, and easy to read. Use clear variable names, follow consistent coding styles, and write comments where necessary.
- **Readme File:** You **must** include a detailed `README.md` file in your repository. This file should contain:
    - A brief overview of the project.
    - Instructions on how to set up the project locally (including any dependencies).
    - Instructions on how to run the application.
    - A brief description of your project's architecture, including your technology choices (frontend framework, backend language, database) and the reasons for those choices.
- **Version Control:** The project must be submitted as a Git repository (e.g., on GitHub or GitLab). We will review your commit history to understand your development process.
- **Submission:** Please submit a link to your public Git repository.

---

4. Bonus Points (Optional)

These are not required, but implementing them will give us a more complete picture of your skills.

- **Content Upload:** Allow instructors to upload a file (e.g., an image or PDF) for a lecture instead of just a URL.
- **Course Search:** Implement a search functionality for students to find courses by title or keywords.
- **Responsive Design:** Ensure the application is usable on different screen sizes (mobile, tablet, desktop).

Submission Guidelines

- create your own public repository.
- Commit your code as you work.
- Upon completion, share the repository link.