MISUN Academy

Technical Assessment: Web Developer

Project: Full-Featured EdTech Platform ("CourseMaster")

Duration: **4 Days**

1. Project Overview

The purpose of this technical assessment is to evaluate your ability to design and develop a full-featured, production-ready E-learning platform using the MERN stack. This project, titled **CourseMaster**, simulates a real-world environment where modern EdTech platforms must support thousands of students, instructors, and administrators while ensuring scalability, performance, and reliability.

You are required to build the core functional modules of an E-learning system where:

- Students can browse, purchase, and consume courses.
- Administrators can manage courses, track enrollments, and review assignments.
- The platform supports secure authentication, efficient APIs, and a structured database design.

The objective is not just to "make it work," but to demonstrate that you can build a clean, maintainable, and scalable system with production-ready architecture.

2. Technology Stack (Mandatory)

- **Frontend:** Next.js (Pages or App Router) or React.js.
- **Backend:** Node.js with Express.js.
- **Database:** MongoDB with Mongoose ODM.
- **State Management:** Redux (Toolkit preferred) or Context API.

3. Key Functional Requirements

A. Authentication & Authorization

- **Student:** Registration, Login, and Logout (JWT Authentication).

- **Admin:** distinct login (can be a hardcoded seeder or a specific registration key).
- **Security:** Passwords must be hashed (bcrypt). Protect private routes (e.g., Dashboard).

B. Public Pages (Unprotected)

1. **Home/Course Listing:**
   - Display available courses.
   - **Advanced Features:** Implement Server-Side Pagination, Searching (by title/instructor), Sorting (Price: Low/High), and Filtering (Category/Tags).
2. **Course Details:**
   - Detailed view of a course (Title, Description, Instructor, Syllabus, Price).
   - "Enroll Now" button (If logged in -> Payment/Enroll; If guest -> Redirect to Login).

C. Student Features (Protected)

1. **Student Dashboard:**
   - View list of enrolled courses.
   - Progress tracking (e.g., "40% Completed").
2. **Course Consumption:**
   - Watch video lectures (can use embedded YouTube/Vimeo links for simplicity).
   - **Mark as Completed:** A button to mark a specific lesson as done, updating the progress bar.
3. **Assignments & Quizzes:**
   - **Assignment:** An input field to submit a Google Drive link or text answer for a specific module.
   - **Quiz:** A multiple-choice quiz interface for a specific module. The score is displayed immediately upon submission.

D. Admin Features (Protected)

1. **Course Management:**
   - Create, Read, Update, Delete (CRUD) courses.
   - Define "Batches" for courses (e.g., Batch 1 starts Jan 1st).
2. **Enrollment Management:**
   - View all students enrolled in a specific course or batch.
3. **Assignment Review:**
   - View submitted assignments from students.

4. Technical & Non-Functional Requirements

Database & Performance

1. **Schema Design:** Use Mongoose relationships/references properly (e.g., linking `Courses` to `Users` for enrollment).
2. **Optimization:**
   - Implement **database indexing** for search fields.
   - Prevent the N+1 problem when fetching course data.
   - Use caching where applicable (optional but recommended).

Code Quality

1. **Error Handling:** Proper try/catch blocks and a global error handling middleware in Express.
2. **Validation:** Use Joi or Zod for input validation (API side).
3. **Clean Code:** Modular file structure (Controllers, Routes, Services, Models).

5. Bonus Challenge (To Stand Out)

*Implement **one** of the following to earn extra credit:*

1. **Redis Integration:** Cache the "All Courses" API response to reduce database load.
2. **Analytics Dashboard:** An admin chart showing enrollments over time (use Recharts or Chart.js).
3. **Email Notification:** Send a welcome email upon registration (using Nodemailer).

6. Submission Guidelines

1. **Source Code:** Push your code to a public **GitHub repository**.
   - Include a `README.md` with:
     - Project Name & Description.
     - Installation/Run instructions.
     - List of `.env` variables required.
     - API Documentation.
2. **Live Deployment:**
   - **Frontend:** Vercel or Netlify.
   - **Backend:** Vercel, Render, or Heroku.
   - *Ensure the live link is functional.*

Evaluation Criteria

You will be judged on:

1. **Completeness:** Are all requirements met?
2. **Code Structure:** Is the code reusable, clean, and organized?
3. **UI/UX:** Is the application responsive and easy to use?

4. **Bug Free:** Does the search/filter logic break? Does the auth persist on reload?