# **LMS Backend API – Approach & Implementation Document**

## **Project Objective**

To develop a scalable and secure **Learning Management System (LMS)** backend API using **Node.js**, **Express**, and **MongoDB**. The API manages user registration, course content, lesson tracking, quizzes, and progress analytics.

## **Tech Stack**

| **Layer** | **Technology** |
| --- | --- |
| Runtime | Node.js (v20+) |
| Framework | Express.js |
| Database | MongoDB (Mongoose ODM) |
| Auth | JWT |
| Validation | Joi |
| Middleware | Express middlewares |
| Rate Limiting | express-rate-limit |

## **Folder Structure**

project/

├── src/

│ ├── config/ # DB and JWT config

│ ├── controllers/ # Business logic

│ ├── models/ # Mongoose schemas

│ ├── routes/ # API route definitions

│ ├── middlewares/ # Auth, validation, rate limiting

│ └── app.js # Core Express app setup

├── server.js # Entry point

├── package.json # Project config

├── .env # Environment variables (local only)

└── .gitignore # Excludes node\_modules, .env, etc.

## **Implemented Features**

### **1. User Authentication**

* **Signup/Login**
  + JWT-based
  + Password hashing
* **Middleware**
  + JWT verification for protected routes
  + Role-based access (admin vs user)

### **2. Course Management**

* **Admin-only course creation**
* **List all courses with pagination**
  + GET /api/courses?page=1&limit=10
* **Get course details with nested lessons/quizzes (paginated)**
  + GET /api/courses/:id?lessonPage=1&quizPage=1

### **3. User Enrollment**

* Users can enroll into courses:  
  + POST /api/courses/:id/enroll
* Prevents duplicate enrollments

### **4. Lesson Management**

* **Admin** can add lessons to a course
* **User** can mark lessons as completed:  
  + POST /api/lessons/:id/complete

### **5. Quiz System**

* Admin adds quizzes to courses
* Users can:  
  + Attempt quizzes
  + Submit multiple attempts
  + View scores per quiz

### **6. Progress Tracking**

* Tracks:  
  + Completed lessons
  + Quiz scores
* Returns:  
  + % completion
  + Number of lessons completed
  + Quiz attempt history

## **Enhancements**

| **Enhancement** | **Implementation** |
| --- | --- |
| **Rate Limiting** | express-rate-limit to prevent brute-force attacks on login & abuse of APIs |
| **Input Validation** | Joi used for validating API payloads before processing |
| **Pagination** | Applied to courses, lessons, and quizzes |
| **Error Handling** | All controllers return consistent error formats with status codes |

## **API Overview (Sample)**

| **Method** | **Endpoint** | **Access** | **Description** |
| --- | --- | --- | --- |
| POST | /api/auth/signup | Public | Register new user |
| POST | /api/auth/login | Public | User login (returns JWT) |
| GET | /api/courses | Public | Paginated course list |
| POST | /api/courses | Admin | Create a new course |
| GET | /api/courses/:id | Public | Get course with lessons/quizzes |
| POST | /api/courses/:id/enroll | User | Enroll in a course |
| POST | /api/courses/:id/lessons | Admin | Add lesson to course |
| POST | /api/lessons/:id/complete | User | Mark lesson as completed |
| POST | /api/courses/:id/quizzes | Admin | Add quiz to course |
| POST | /api/quizzes/:id/attempt | User | Attempt a quiz |
| GET | /api/quizzes/:id/scores | User | View quiz scores |
| GET | /api/progress/:courseId | User | View overall course progress |

## **Testing**

* **Tool:** Thunder Client
* Sample data prepared for:  
  + Courses with multiple lessons/quizzes
  + Valid and invalid JWT scenarios
  + Input validation failures
  + Pagination tests