MERN Stack Project Assignment

Learning Management System (LMS)

🎯 Project Objective

Build a full-stack **Learning Management System (LMS)** where:

- Admin can manage users and courses
- Instructors can create courses and upload lessons
- Students can enroll and access courses
- Authentication & Authorization implemented
- REST APIs built with Node.js & Express
- MongoDB used as database
- ReactJS used as frontend

📐 Tech Stack

- Frontend: ReactJS
- Backend: Node.js + Express.js
- Database: MongoDB + Mongoose
- Authentication: JWT
- Password Encryption: bcrypt

📌 Functional Requirements

👤 1. Authentication System

- Register (Student / Instructor)
- Login
- JWT Token generation
- Protected Routes
- Role-based Access (Admin / Instructor / Student)

🗄️ Database Models (MongoDB + Mongoose)

1️⃣ User Model

{

```
 name: String,
 email: String,
 password: String,
 role: {
  type: String,
  enum: ["admin", "instructor", "student"],
  default: "student"
 },
 enrolledCourses: [
  {
   type: mongoose.Schema.Types.ObjectId,
   ref: "Course"
  }
 ],
 createdAt: Date
}
```

---

2 Course Model

```
{
 title: String,
 description: String,
 price: Number,
 instructor: {
  type: mongoose.Schema.Types.ObjectId,
  ref: "User"
 },
 studentsEnrolled: [
  {
   type: mongoose.Schema.Types.ObjectId,
   ref: "User"
  }
```

```
 ],
 lessons: [
  {
   type: mongoose.Schema.Types.ObjectId,
   ref: "Lesson"
  }
 ],
 createdAt: Date
}
```

3️⃣ Lesson Model

```
{
 title: String,
 content: String,
 videoUrl: String,
 course: {
  type: mongoose.Schema.Types.ObjectId,
  ref: "Course"
 },
 createdAt: Date
}
```

4️⃣ Enrollment Model (Optional Advanced Level)

```
{
 student: {
  type: mongoose.Schema.Types.ObjectId,
  ref: "User"
 },
 course: {
  type: mongoose.Schema.Types.ObjectId,
  ref: "Course"
```

},

enrolledAt: Date,

progress: Number

}

🚀 **API Routes Structure**

Base URL: /api

🔐 **Auth Routes**

Method	Route	Description
POST	/auth/register	Register user
POST	/auth/login	Login user
GET	/auth/me	Get logged in user

👤 **User Routes (Admin Only)**

Method	Route	Description
GET	/users	Get all users
GET	/users/:id	Get single user
DELETE	/users/:id	Delete user

📚 **Course Routes**

Method	Route	Role	Description
POST	/courses	Instructor	Create course
GET	/courses	Public	Get all courses
GET	/courses/:id	Public	Get course details
PUT	/courses/:id	Instructor	Update course
DELETE	/courses/:id	Instructor	Delete course

🎓 **Enrollment Routes**

Method	Route	Role	Description
POST	/enroll/:courseId	Student	Enroll in course
GET	/my-courses	Student	View enrolled courses

📖 Lesson Routes

Method	Route	Role
POST	/lessons/:courseId	Instructor
GET	/lessons/:courseId	Student
PUT	/lessons/:id	Instructor
DELETE	/lessons/:id	Instructor

🔒 Middleware Requirements

Students must implement:

- authMiddleware (Verify JWT)
- roleMiddleware (Check role)

🖥️ ReactJS Frontend Structure

📁 Folder Structure

```
src/
 ├── pages/
 ├── components/
 ├── context/
 └── App.js
```

📄 React Pages

1️⃣ Public Pages

- Home Page

- Course Listing Page
- Course Details Page
- Login Page
- Register Page

2️⃣ Student Dashboard

- My Courses Page
- Course Learning Page
- Profile Page

3️⃣ Instructor Dashboard

- Create Course Page
- My Created Courses Page
- Add Lesson Page
- Edit Course Page

4️⃣ Admin Dashboard

- Manage Users Page
- Manage Courses Page

🧠 React Concepts Students Must Use

- useState
- useEffect
- useContext (AuthContext)
- Protected Routes
- Axios for API calls
- React Router
- Role-based UI rendering
- Form validation

📦 Deliverables

Students must submit:

1. GitHub Repository
2. Backend Folder
3. Frontend Folder
4. README with:
    - Setup Instructions
    - API Documentation
    - Database Schema
    - Screenshots