Full Project Setup & Configuration Guide

1. Project Overview

This project is a full-stack web application for managing an online course platform. It includes a React frontend, a Node.js + Express backend, and a MongoDB database.

Users are divided into three roles:

- Admin: Manages users and course listings.
- Teacher: Creates and manages courses.
- Student: Views, enrolls, and takes courses.

Project Stack:

- Frontend: React.js, Vite, Axios, Tailwind CSS
- Backend: Node.js, Express.js, Mongoose
- Database: MongoDB (NoSQL)

2. Frontend Setup (React)

Step-by-Step Setup:

1. Create React App:
   - Run: npm create vite@latest frontend -- --template react
   - cd frontend

2. Install dependencies:
   - npm install
   - npm install axios react-router-dom

3. Folder Structure:
   - /src/components/common: Shared UI components
   - /src/components/admin: Admin dashboard pages
   - /src/components/user/student: Student dashboard
   - /src/components/user/teacher: Teacher dashboard

4. Main Entry:
   - App.jsx: Routes and Layout
   - main.jsx: Application bootstrap

5. Environment Setup (optional):
   - Create `.env` file with any frontend-specific config (e.g. API base URL)

6. Start App:
   - npm run dev

3. Backend Setup (Node.js + Express)

Step-by-Step Setup:

1. Initialize Project:
   - mkdir backend && cd backend
   - npm init -y

2. Install Dependencies:
   - npm install express mongoose dotenv cors jsonwebtoken bcryptjs multer

3. Folder Structure:
   - /config: DB configuration
   - /controllers: Logic for each endpoint
   - /routers: API route definitions
   - /schemas: Mongoose models (Users, Courses, Enrollments)
   - /middlewares: JWT authentication
   - /uploads: Store uploaded files

4. Entry Point (index.js):
   - Load environment variables
   - Connect to MongoDB
   - Configure middleware and routes
   - Start server on defined port

5. Run Server:
   - node index.js OR npm start

6. Environment Setup:
   Create `.env` file with:

- PORT=5000
- DB_URI=mongodb://localhost:27017/yourdb
- JWT_SECRET=your_secret

4. Database Schema Design (MongoDB)

Collections & Schemas:

1. Users (userModel.js)
   - Fields: name, email, password (hashed), role

2. Courses (courseModel.js)
   - Fields: title, description, teacherId, videoLinks, etc.

3. EnrolledCourses (enrolledCourseModel.js)
   - Fields: courseId, userId, progress, enrolledOn

4. Payments (coursePaymentModel.js)
   - Fields: courseId, userId, paymentStatus, date

5. Relationships:
   - One teacher can create many courses
   - One student can enroll in many courses
   - A course can have many students

Mongoose ensures schema validation and provides easy querying.

5. API & Routing Overview

API Routing:

1. User Routes (userRoutes.js)
   - POST /register Create user
   - POST /login Authenticate user
   - GET /user/profile View profile
   - GET /courses View available courses

2. Admin Routes (adminRoutes.js)
   - GET /admin/users List all users
   - DELETE /admin/user/:id Remove user
   - POST /admin/course Approve or reject course

3. Auth Middleware (authMiddleware.js)
   - Validates JWT and protects private routes

All routes are organized and protected by middleware for role-based access.