Project Overview: Concept Master LMS

This document provides a comprehensive technical analysis of the Concept Master Learning Management System (LMS). It covers the technology stack, project structure, backend and frontend architectures, and instructions on how to run the project.

1. Technology Stack

Backend

- **Runtime**: Node.js
- **Framework**: Express.js
- **Database**: MongoDB with Mongoose ODM
- **Authentication**: JSON Web Tokens (JWT)
- **Security**: Helmet, CORS, Rate Limiting
- **Logging**: Morgan, Winston
- **File Uploads**: Multer

Frontend

- **Framework**: React.js (Vite)
- **Styling**: Tailwind CSS, Emotion, Material UI Icons
- **Routing**: React Router DOM
- **HTTP Client**: Axios
- **State Management**: React Context API (AuthContext)
- **Visuals**: Framer Motion, Recharts, React Three Fiber

2. Project Structure

The project is divided into two main directories:

- **`backend/`**: Contains the server-side code, API, and database logic.
- **`frontend/`**: Contains the client-side React application.

3. Backend Architecture

Entry Point

- **src/server.js**: The main entry point. It initializes the Express app, connects to MongoDB, sets up middleware (CORS, Helmet, Compression), and defines API routes. It also handles initial data seeding (Admin and Classes).

Database

- **Configuration**:
  src/config/db.js handles the MongoDB connection.
- **Models**: Located in `src/models/`. Key models include:

- `User`: Stores user data and roles (student, teacher, parent, admin).
- `Class`, `Subject`, `Chapter`: Hierarchical structure for course content.
- `Lecture`, `Note`: Content types linked to chapters.
- `Quiz`, `Question`, `QuizAttempt`, `QuizResult`: Assessment system.
- `TeacherRequest`: Handles teacher registration requests.

API Routes

Routes are defined in `src/routes/` and mounted in
server.js. Key endpoints:

- `/api/auth`: Login, register, profile management.
- `/api/admin`: Administrative tasks (manage users, content).
- `/api/student`: Student-specific features.
- `/api/teacher`: Teacher-specific features.
- `/api/lectures`, `/api/notes`, `/api/quizzes`: Content management.

Authentication & Security

- **Middleware**: `src/middlewares/auth.middleware.js` (implied) protects routes using JWT verification.
- **Role-Based Access**: Routes are protected based on user roles (Admin, Teacher, Student, Parent).

4. Frontend Architecture

Entry Point & Routing

- **src/main.jsx**: Bootstraps the React app.
- **src/App.jsx**: Defines the application routes and layout. It uses
ProtectedRoute to restrict access based on user roles.

State Management

- **src/context/AuthContext.jsx**: Manages global authentication state (user login status, user data).

API Integration

- **src/lib/api.js**: Configures the Axios instance for making HTTP requests to the backend. It likely handles token attachment for authenticated requests.

Key Pages (`src/pages/`)

- **Public**:
Home, `Login`, `Register`.
- **Student**: `StudentDashboard`, `Notes`, `Lectures`, `QuizList`, `AttemptQuiz`.
- **Teacher**: `TeacherDashboard`, `UploadNotes`, `UploadLectures`, `CreateQuiz`.
- **Admin**: `AdminDashboard`, `ManageUsers`, `SystemOverview`.

5. How to Start

Prerequisites

- Node.js installed.
- MongoDB installed and running locally (or a remote connection string).

Backend Setup

1. Navigate to the backend directory:

   ```
   cd backend
   ```

2. Install dependencies:

   ```
   npm install
   ```

3. Configure environment variables:
   - Create a
     .env file based on
     .env.example.
   - Ensure `MONGO_URI` points to your MongoDB instance.
   - Set `JWT_SECRET` to a secure string.
4. Start the server:

   ```
   npm start
   # OR for development with auto-reload
   npm run dev
   ```

Frontend Setup

1. Navigate to the frontend directory:

   ```
   cd frontend
   ```

2. Install dependencies:

   ```
   npm install
   ```

3. Start the development server:

   ```
   npm run dev
   ```

4. Open your browser and navigate to the URL shown (usually `http://localhost:5173`).

6. Key Features

- **Role-Based Dashboards**: Customized views for Students, Teachers, Parents, and Admins.
- **Content Management**: Teachers can upload notes and lectures; Admins approve them.
- **Assessment System**: Quizzes with immediate feedback and result tracking.
- **AI Integration**: "Concept Master AI" for student assistance (integrated via backend/frontend).

- **Analytics**: Performance tracking for students and system usage stats for admins.
-