Week 19: LMS (EduHub) - Course Content Delivery & Student Enrollment

Day 1: Lesson Modules and Nested Routing

Theoretical Overview

The core of **EduHub** is the delivery of educational content. To provide a seamless learning experience, we implemented a hierarchical navigation system that allows students to progress through lessons within a specific course without losing their place.

Implementation Logic

- **Hierarchical Navigation**: Using **React Router**, we implemented nested routes where the URL structure reflects the content hierarchy (e.g., /course/:courseId/lesson/:lessonId).
- **The Outlet Component**: We utilized the <Outlet /> component to render the specific lesson content within a persistent course layout, which includes the course syllabus and progress bar.
- **Dynamic Data Fetching**: The useParams hook is used to extract the lessonId, which then triggers a useEffect call to fetch the specific lesson's text and video data from the MongoDB "Courses" collection.

Day 2: The Student Enrollment System

Managing Relationships in NoSQL

A Learning Management System requires a robust way to link students to the courses they purchase or join.

- **The Enrollment Collection**: We created a separate MongoDB collection called "Enrollments" to act as a bridge between the "Users" and "Courses" collections.
- **Enrollment Logic**: When a student clicks "Join Course," a POST request is sent to the Express backend. The server creates a new document containing the studentId, courseId, and enrollmentDate.
- **Duplicate Prevention**: We implemented a backend check to verify if a document with that specific studentId and courseId combination already exists before allowing the insert operation, preventing double enrollment.

Day 3: Multimedia Integration and Course Viewer

Delivering Diverse Content

To cater to different learning styles, the **EduHub** course viewer supports various media types, drawing from the foundations laid in **Week 1**.

- **Video Integration**: We used the HTML5 <video> tag and embedded YouTube IFRAMEs to display lecture videos.
- **Figure and Captions**: Technical diagrams and illustrations are wrapped in <figure> and <figcaption> tags to ensure SEO and accessibility standards are met.
- **Document Downloads**: We utilized the express.static middleware on the backend to allow students to download PDF resources associated with specific lessons.

Day 4: Communication State and Progress Tracking

Enhancing the Student Journey

- **Progress Tracking**: We implemented a "Mark as Complete" feature. When clicked, the frontend sends a PATCH request to the enrollment document to update the completedLessons array.

- **Communication State (Appendix B)**: We utilized global state to manage the "Communication State," showing success toast notifications when a student completes a module.
- **Loading States**: While the course content is being fetched, "Skeleton" components are displayed to improve the perceived performance of the app.

Day 5: Synthesis of Class and Functional Components

Integrating Legacy Knowledge

In accordance with previous lectures, we integrated class-based components for some of the more complex data tables in the student dashboard.

- **Lifecycle Methods**: We reviewed how componentDidMount in class components serves the same purpose as useEffect with an empty dependency array in our functional components.
- **Prop Types and Default Props**: We implemented propTypes to validate the data being passed into the course listing components, ensuring the LMS is robust and bug-free.