**Full Stack Development with MERN**

**Introduction:**

* *Project Title:* 
  + Online Learning Platform
* *Team Members:*
  + Syed Sahil A (2021503053): Team Leader, Documentation, Version Control.
  + Praveen P (2021503035): Frontend developer.
  + Shanthosh Kumar E (2021503049): Backend developer.
  + Rupesh A (2021503549): Frontend and backend integration.
  + Hemnath S (2021503513): Frontend and backend integration.

**Project Overview:**

* *Purpose:*
  + To provide a flexible, accessible, and user-friendly platform for online learning, enabling learners and instructors to connect, share knowledge, and achieve educational goals through interactive and self-paced courses.
* *Features:*
  + User-Friendly Interface: Simplified navigation for learners and instructors.
  + Course Management: Upload, organize, and track course materials and progress.
  + Interactivity: Discussion forums, live webinars, and real-time chat support.
  + Certification: Digital certificates upon course completion.
  + Accessibility: Multi-device compatibility for learning anytime, anywhere.
  + Self-paced learning: Freedom to progress through content based on individual schedules.
  + Payment Options: Free and premium courses with secure payment systems.

**Architecture:**

* *Frontend:* The frontend is built using React.js, employing a component-based architecture to ensure modularity and reusability. Key elements include:
  + Routing: React Router for navigating between pages such as course browsing, user profile, and course details.
  + UI Libraries: Bootstrap and Material-UI for responsive and user-friendly design.
  + State Management: Context API or Redux (optional for scalability) to manage application-wide states like user authentication and course enrollment.
  + API Integration: Axios is used for communication with backend RESTful APIs to fetch and send data in real time.
* *Backend:* The backend is developed using Node.js and Express.js, designed with a layered architecture:
  + Route Layer: Handles API endpoints for user actions (e.g., registration, course enrollment).
  + Controller Layer: Manages the business logic for processing requests and responses.
  + Service Layer: Handles data manipulation and interaction with the database.
  + Middleware: Implements features like authentication (JWT) and error handling.
* *Database*: The database uses MongoDB to store and retrieve structured data. The schema includes:
  + User Collection: Stores user details (e.g., name, email, password, role [student/instructor], enrolled courses).
  + Course Collection: Contains course information (e.g., title, description, instructor, modules, pricing).
  + Progress Collection: Tracks individual learner progress (e.g., completed modules, scores).
  + Payments Collection: Manages payment records and subscriptions for premium courses.
  + Database interactions use Mongoose, providing a clear schema definition and efficient query handling. The relationships are designed to ensure flexibility and scalability, allowing seamless addition of new features.

**Setup Instructions:**

* *Prerequisites:*
  + Node.js (v16.x or later) - For running the backend and managing dependencies.
  + MongoDB (v5.x or later) - For database storage.
  + Git - For cloning the project repository.
  + Web Browsers - Two installed browsers for testing (e.g., Chrome and Firefox).
  + Code Editor - Recommended: Visual Studio Code (VS Code).
  + Internet - Minimum bandwidth of 30 Mbps.
* *Installation:*
  + Clone the Repository:

git clone <repository-url>

cd <project-directory>

* + Install Dependencies:
    - For Frontend:

cd frontend

npm install

* + - For Backend:

cd backend

npm install

* + Set Up Environment Variables:
    - Create a .env file in the root of the backend directory with the following values:

PORT=8000

MONGO\_URI=<your-mongodb-connection-string>

* + Start the Application:
    - Run Backend:

cd backend

npm start

* + - Run Frontend:

cd client

npm start

* + Access the Application:
    - Open a browser and navigate to http://localhost:5173.
  + Testing and Usage:
    - Register as a user or instructor to explore features.
    - Upload courses, enroll, and simulate interactions.

**Folder Structure:**

A black screen with white text

Description automatically generatedA screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Running the Application:**

* To run the application locally, use the following commands:
  + Frontend:
    - Navigate to the client directory and start the frontend server:

cd frontend

npm run dev

* + - This will launch the React application at http://localhost:5173.
  + Backend:
    - Navigate to the server directory and start the backend server:

cd backend

npm start

* + - The backend server will run on http://localhost:8000 (or the port specified in the .env file).
  + Ensure both servers are running simultaneously for full functionality.

**API Documentation:**

* *User Registration:*
  + Endpoint: /register
  + Method: POST
  + Description: Registers a new user by providing name, email, and password.
* *User Login:*
  + Endpoint: /login
  + Method: POST
  + Description: Authenticates a user and returns a JWT token for access.
* *Add Course:*
  + Endpoint: /addcourse
  + Method: POST
  + Description: Adds a new course with uploaded content (requires authentication).
* *Get All Courses:*
  + Endpoint: /getallcourses
  + Method: GET
  + Description: Retrieves a list of all available courses.
* *Get All Teacher’s Courses:*
  + Endpoint: /getallcoursesteacher
  + Method: GET
  + Description: Retrieves all courses uploaded by the authenticated teacher.
* *Delete Course:*
  + Endpoint: /deletecourse/:courseid
  + Method: DELETE
  + Description: Deletes a specific course by its ID (requires authentication).
* *Enroll in a Course:*
  + Endpoint: /enrolledcourse/:courseid
  + Method: POST
  + Description: Enrolls the authenticated user in the specified course.
* *Get Course Content:*
  + Endpoint: /coursecontent/:courseid
  + Method: GET
  + Description: Retrieves the content of a specific course (requires authentication).
* *Complete Module:*
  + Endpoint: /completemodule
  + Method: POST
  + Description: Marks a course section as complete for the authenticated user.
* *Get User’s Enrolled Courses:*
  + Endpoint: /getallcoursesuser
  + Method: GET
  + Description: Retrieves all courses the authenticated user is enrolled in.
* *Get All Users:*
  + Endpoint: /getallusers
  + Method: GET
  + Description: Fetches details of all users (admin access required).
* *Delete User:*
  + Endpoint: /deleteuser/:cuserid
  + Method: DELETE
  + Description: Deletes a user by their ID (admin access required).

**Authentication:**

* *Authentication:* The project uses JWT (JSON Web Tokens) for secure authentication. A token is generated and sent back to the client when a user logs in. The token is included in the Authorization header for all protected routes.
* *Authorization:* Middleware (authMiddleware) verifies the token and grants access to protected resources based on the user's role (student, instructor, admin).
* *Token Handling:* Tokens are generated using jsonwebtoken with a secret key (JWT\_SECRET from the .env file). Expired or invalid tokens result in a 401 Unauthorized error.
* *Session Management:* Sessions are stateless, relying on the client to store and send tokens. To add security, tokens can be stored in HTTP-only cookies to prevent XSS attacks.

**User Interface:**

* *Home Page:*

A person and person sitting on steps with a computer

Description automatically generated

* *Signup Page:*

A screenshot of a computer

Description automatically generated

* *Login Page:*

A person sitting on steps looking at a screen

Description automatically generated

* *Teacher Home Page:*

A screenshot of a computer

Description automatically generated

* *Course Creation:*

A screenshot of a computer

Description automatically generated

* *Trending Courses:*

A screenshot of a computer

Description automatically generated

* *Student Home Page:*

A screenshot of a computer

Description automatically generated

* *Course Enrollment:*

A screenshot of a computer

Description automatically generated

* *Course Progression:*

A screenshot of a computer

Description automatically generated

* *Completion Certificate:*

A close-up of a certificate

Description automatically generated

* *Course Deletion:*

A screenshot of a chat

Description automatically generated

**Testing:**

* The project follows a comprehensive testing strategy to ensure the application's functionality, performance, and reliability. Testing is divided into the following stages:
* *Unit Testing:*
  + Focuses on testing individual components or modules in isolation.
  + Examples: Testing backend APIs (e.g., user registration, course creation).
* *Integration Testing:*
  + Verifies that different modules work together as expected.
  + Examples: Ensuring the frontend properly integrates with backend APIs.
* *Manual Testing:*
  + Performed for exploratory and UI/UX testing to ensure the user interface meets expectations.

**Screenshots or Demo:**

* *Demo Video:* [olp\_demo\_video.mp4](https://drive.google.com/file/d/1JdTaeTRxqj4nGSQ0hQK579PA0FPm8dP_/view?usp=sharing)

**Known Issues:**

* *File Upload Limitations:* 
  + Large files may take longer to upload or could fail due to server limitations.
  + Potential Fix: Optimize file upload configurations and enable cloud-based storage like AWS S3 or Google Cloud Storage.
* *Error Handling:* 
  + Some API responses might not provide detailed error messages, making debugging difficult.
  + Potential Fix: Improve error handling middleware for descriptive error responses.
* *Token Expiration:*
  + Users need to log in again when their JWT token expires, which might disrupt the learning experience.
  + Potential Fix: Implement token refresh functionality.

**Future Enhancements:**

* *AI-Powered Recommendations:* Use machine learning to recommend courses based on user interests, progress, and feedback.
* *Localization:* Add multi-language support to cater to a global audience.
* *Live Class Integration:* Allow instructors to schedule and conduct live classes or webinars directly on the platform using tools like Zoom or WebRTC.
* *Course Reviews and Ratings:* Enable students to leave feedback and ratings for courses, helping others make informed choices.
* *Third-Party Integration:* Integrate with platforms like LinkedIn for certification sharing and professional networking.