IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 9, September 2026

DOI: 10.17148/IJARCCE.2026.15908

DESIGN AND DEVELOPMENT OF A FULL-STACK LEARNING MANAGEMENT SYSTEM WITH SECURE AUTHENTICATION, ONLINE PAYMENT, AND COURSE PROGRESS TRACKING

Akshata Mahishawadagi

MCA Final Year, Department of Computer Applications, Bagalkot University, Jamkhandi, Karnataka, India

Abstract: This paper presents the design and development of a full-stack Learning Management System (LMS) intended to provide an integrated platform for course management, digital learning, authentication, online enrollment, payment processing, and learner progress tracking. The system is implemented using the MERN technology stack, with React.js and Vite on the client side, Node.js and Express.js on the server side, and MongoDB Atlas for persistent data storage. Clerk is used for user authentication, while Stripe supports online payments and webhook-based payment confirmation. Cloudinary is used for course thumbnail media storage. The platform supports educator-oriented course creation with chapters and lectures and student-oriented course discovery, purchase, enrollment, video learning, and progress management. RESTful APIs connect the client and server, while database models maintain course, purchase, enrollment, and progress information. Functional testing of the implemented workflow demonstrates that the major learning cycle—from authentication and course publication through payment, enrollment, lecture access, and progress tracking—can be handled within a unified web application. The work demonstrates how contemporary full-stack web technologies and third-party services can be combined to construct a practical and deployable LMS.

Keywords: Learning Management System, MERN Stack, React.js, MongoDB Atlas, Clerk Authentication, Stripe Payment, Webhook, Course Progress Tracking

I. INTRODUCTION

The growth of web-based education has increased the need for learning platforms that can combine course delivery, user management, content administration, and learner monitoring in a single environment. Traditional approaches that depend on separate tools for authentication, content delivery, payment collection, and progress management can create fragmented user experiences and increase administrative effort. A modern LMS can reduce this fragmentation by integrating these activities through a common web application.

The proposed Learning Management System is a full-stack web application developed with the MERN ecosystem. The application provides two major operational perspectives: learners can browse available courses, purchase courses, access enrolled learning content, watch lectures, and track completed lectures; educators can create and manage courses containing chapters and lectures and publish course content. Authentication is handled through Clerk, course thumbnails are stored through Cloudinary, and Stripe is integrated for online payment processing. MongoDB Atlas stores application data and Express.js exposes RESTful APIs to the React-based client.

The main objective of the project is to provide an accessible and maintainable learning platform while demonstrating a complete full-stack workflow. The system is designed so that a successful payment can be validated through a Stripe webhook before the corresponding purchase is treated as completed, helping connect financial transactions with course enrollment. The resulting architecture also separates presentation, business logic, and persistence responsibilities, making the application suitable for further enhancement.

II. LITERATURE REVIEW

Existing work on online learning systems shows that effective digital education platforms require more than static content delivery. Learning management systems commonly include user accounts, course organization, assessment or activity tracking, communication, and reporting. Cloud-based education platforms further emphasize scalability, accessibility, and integration with external services. Contemporary web application development also encourages modular client-server architectures in which a browser-based interface communicates with backend APIs.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 9, September 2026
DOI: 10.17148/IJARCCE.2026.15908

Open-source and commercial LMS platforms demonstrate the value of centralized course administration and learner tracking, but their breadth can also make them complex for small institutions or academic projects to customize. Research and technical literature on web application security emphasizes the need for authenticated access, controlled authorization, secure handling of credentials, and validation of externally generated events. Payment gateway integrations similarly require server-side verification rather than trusting only client-side status.

The present work applies these principles to a focused LMS implementation. Instead of attempting to reproduce the entire functionality of large institutional platforms, the project concentrates on a coherent learning cycle: identity management, educator course creation, learner purchase, verified payment, enrollment, lecture access, and progress tracking. This provides a practical basis for evaluating the integration of modern web technologies in an educational application.

III. EXISTING SYSTEM AND PROBLEM STATEMENT

A. Existing System

Many learning environments use separate systems for course presentation, user authentication, payment, and content management. In smaller implementations, course information may be maintained manually and payment confirmation may require administrative intervention. Such separation can lead to duplicated data, inconsistent enrollment status, and additional effort for educators.

B. Problem Statement

The problem addressed by this project is the development of a unified web-based learning environment that can securely manage users, courses, paid enrollment, lecture delivery, and learner progress. The system must maintain a consistent relationship between the user identity, selected course, successful payment, enrollment status, and completed learning activities.

C. Objectives

- To develop a full-stack LMS using React.js, Node.js, Express.js, and MongoDB.
- To provide authenticated access for learners and educators using Clerk.
- To enable educators to create courses with chapters, lectures, prices, discounts, and thumbnails.
- To allow learners to browse, purchase, enroll in, and access course lectures.
- To integrate Stripe payment processing with webhook-based payment confirmation.
- To maintain learner progress and completed lecture information in MongoDB.
- To provide a modular architecture that can be deployed and extended.

IV. PROPOSED SYSTEM

The proposed system is a centralized web application in which the major LMS operations are connected through RESTful APIs. The React client provides the user interface, while the Express.js server handles application logic and database interaction. MongoDB Atlas stores persistent entities such as users, courses, purchases, and course progress. Third-party services are integrated for authentication, payment processing, and media storage.

A. Major Modules

- Authentication Module: manages sign-up, sign-in, and authenticated application access through Clerk.
- Course Management Module: enables educators to create courses, chapters, lectures, thumbnails, prices, and discounts.
- Course Discovery Module: retrieves published courses and presents course information to learners.
- Purchase and Enrollment Module: records course purchases and associates successful purchases with learner enrollment.
- Payment Verification Module: receives Stripe webhook events and updates purchase state after server-side verification.
- Learning Module: provides enrolled learners with lecture/video access.
- Progress Module: stores completed lectures and course completion state for each learner.

B. Technology Stack

Layer	Technology	Purpose
Frontend	React.js, Vite, Tailwind CSS	User interface, routing, course and learning views
Backend	Node.js, Express.js	REST APIs and server-side application logic

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 9, September 2026

DOI: 10.17148/IJARCCE.2026.15908

Database	MongoDB Atlas, Mongoose	Persistent storage and data modeling
Authentication	Clerk	User identity and authenticated access
Payment	Stripe	Online payment processing and webhook events
Media	Cloudinary	Course thumbnail/media storage
Learning Video	React YouTube integration	Lecture video playback

TABLE I TECHNOLOGY STACK USED IN THE PROPOSED SYSTEM

V. SYSTEM ARCHITECTURE

The system follows a client-server architecture. The React/Vite client communicates with the Express.js backend through HTTP-based REST APIs. The backend validates requests, applies application logic, and accesses MongoDB through Mongoose models. Clerk provides identity services, while Stripe handles payment operations and sends webhook events to the backend. Cloudinary provides external media storage for course thumbnails.

A. Architecture Flow

User → React/Vite Client → Express.js REST API → Mongoose → MongoDB Atlas

Authentication: React Client ↔ Clerk

Payment: React Client → Express.js → Stripe → Stripe Webhook → Express.js → Purchase/Enrollment Data

Media: Educator → Application → Cloudinary → Course Thumbnail URL

B. Core Data Entities

Entity	Important Information	Role in System
Course	Course content, educator, thumbnail, price, discount, enrolled students	Stores published and managed course information
Purchase	Course ID, user ID, amount, payment status	Connects payment with course purchase
CourseProgress	User ID, course ID, completed state, completed lectures	Tracks learner progress
User	Identity and application user information	Associates learners/educators with LMS operations
Course Content	Chapters and lecture details including video URLs	Organizes learning material

TABLE II PRINCIPAL DATA ENTITIES

VI. METHODOLOGY

A. Development Approach

The system was developed incrementally, beginning with the identification of learner and educator requirements and followed by data modeling, API development, interface implementation, third-party service integration, and functional testing. The client and server were developed as separate applications so that their responsibilities remain clearly defined.

1) Requirement Analysis:

The requirements were derived from the learning workflow: account creation, course publication, course discovery, purchase, enrollment, lecture access, and progress tracking.

2) Database and API Design:

MongoDB collections and Mongoose schemas were defined for courses, purchases, and progress information. Express routes and controllers provide endpoints for retrieving courses, enrolled courses, user data, and related operations.

3) Frontend Implementation:

React components and pages provide navigation, course presentation, course creation, enrollment views, and the lecture player. Vite is used as the frontend build and development environment.

4) Service Integration:

Clerk is integrated for authentication, Stripe for payment processing and webhook events, and Cloudinary for course thumbnail storage. External service results are incorporated into the application's backend workflow.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 9, September 2026

DOI: 10.17148/IJARCCE.2026.15908

5) Testing and Validation:

The implemented workflow was tested by creating courses, adding lecture content, performing test purchases using Stripe test functionality, verifying enrollment updates, and accessing lecture videos. Database records were inspected to confirm that purchase and progress information corresponded to application behavior.

VII. IMPLEMENTATION

A. Authentication and User Access

Clerk is used to manage user identity and authentication. The application uses authenticated user information when performing learner-specific operations such as retrieving enrolled courses and recording progress. This reduces the need for the application to implement password storage and account credential management directly.

B. Course Creation and Content Management

Educators can add course information and organize learning material into chapters and lectures. Course records include information required for presentation and purchase, while lecture objects contain video-related information. Course thumbnails are handled through Cloudinary, allowing the application database to store references rather than large media files.

C. Payment and Webhook Processing

Stripe is integrated to support online course purchases. The client initiates the purchase workflow through the application backend, while the backend communicates with Stripe. A webhook endpoint receives Stripe events and allows the server to confirm payment-related state. This approach is important because the server should not rely only on a client-side success indication when determining whether a paid course should be treated as purchased.

D. Enrollment and Learning

After successful payment processing, the application associates the learner with the purchased course. The enrollment information is then used to provide access to course learning content. The lecture player supports video-based learning, allowing enrolled learners to view the lecture material.

E. Progress Tracking

Course progress is maintained using a dedicated progress model containing the learner identity, course identity, completion state, and completed lecture information. This allows the application to distinguish between a learner who has enrolled and a learner who has completed individual parts of the course.

VIII. RESULTS AND DISCUSSION

Functional validation of the implemented system demonstrates a complete end-to-end learning workflow. The application was able to support user authentication, educator course creation, lecture/video configuration, course presentation, test payment processing, purchase state handling, enrollment, and video access. During implementation testing, the course became available in the learner's enrollment area after successful payment processing, and the associated lecture video could be played.

The payment workflow also highlighted the importance of webhook-based confirmation. A purchase record can exist independently of final enrollment state, so the backend must process the payment event before treating the transaction as completed. This separation provides a clearer relationship between payment status and learning access.

The project demonstrates that a MERN-based architecture can integrate multiple external services without placing all responsibilities in a single component. React manages the presentation layer, Express.js handles server-side operations, MongoDB persists application data, and specialized services provide authentication, payment, and media capabilities. The modular arrangement provides a practical foundation for future LMS functionality.

Test Area	Expected Behavior	Observed Implementation
Authentication	User can authenticate and access protected LMS features	Supported through Clerk integration
Course Creation	Educator can create course and lecture content	Supported
Payment	Test payment can initiate purchase workflow	Supported through Stripe
Webhook Confirmation	Server receives payment event and updates purchase state	Implemented
Enrollment	Successful purchase results in learner enrollment	Validated

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 9, September 2026

DOI: 10.17148/IJARCCE.2026.15908

Video Learning	Enrolled learner can play lecture video	Validated
Progress	Completed lecture/course state can be stored	Implemented

TABLE III FUNCTIONAL VALIDATION OF THE IMPLEMENTED SYSTEM

IX. ADVANTAGES AND LIMITATIONS

A. Advantages

- Unified workflow for course creation, purchase, enrollment, learning, and progress tracking.
- Separation of frontend, backend, and database responsibilities.
- Third-party authentication reduces direct credential-management responsibility.
- Webhook-based payment confirmation provides a server-side payment event workflow.
- Cloud-based database and media services support deployment-oriented development.

B. Limitations

- The current implementation focuses primarily on course delivery and does not include a complete examination or certification subsystem.
- The learning workflow is centered on video lectures and does not yet provide comprehensive analytics or adaptive learning.
- Production-scale evaluation with a large number of concurrent learners was outside the scope of the current implementation.
- The current project does not claim measured performance benchmarks beyond the functional validation performed during development.

X. FUTURE SCOPE

The system can be extended with online quizzes, assignments, certificates, instructor dashboards, learner analytics, course ratings, notifications, search and recommendation features, and richer administrative controls. A future version can also incorporate automated content recommendations based on learner activity and course completion patterns. Performance testing under concurrent workloads could be conducted to evaluate API response time, database behavior, and scalability. Additional security controls, audit logging, and fine-grained authorization can further strengthen the production deployment.

XI. CONCLUSION

This paper presented the design and development of a full-stack Learning Management System that integrates authentication, course management, online payment, enrollment, video learning, and progress tracking. The MERN architecture provides a clear separation between the client, server, and data layers, while Clerk, Stripe, and Cloudinary provide specialized services for identity, payment, and media storage. Functional validation of the implemented system confirms that the principal learner and educator workflows can operate as an integrated web application. The project therefore provides a practical foundation for delivering paid digital courses and can be extended toward a more comprehensive learning ecosystem.

ACKNOWLEDGMENT

The author acknowledges the guidance and academic support received during the development of this project. The author also acknowledges the open-source frameworks and developer services used to implement and test the system.

REFERENCES

[1] R. S. Pressman and B. R. Maxim, Software Engineering: A Practitioner's Approach, 9th ed. McGraw-Hill, 2019.
[2] J. Duckett, JavaScript and JQuery: Interactive Front-End Web Development. Wiley, 2014.
[3] MongoDB, “MongoDB Documentation,” MongoDB Documentation.
[4] Express.js, “Express - Node.js web application framework,” Express Documentation.
[5] React, “React Documentation,” React Developer Documentation.
[6] Node.js, “Node.js Documentation,” OpenJS Foundation.
[7] Clerk, “Clerk Documentation,” Clerk Developer Documentation.
[8] Stripe, “Stripe Documentation,” Stripe Developer Documentation.
[9] Cloudinary, “Cloudinary Documentation,” Cloudinary Developer Documentation.
[10] Vite, “Vite Documentation,” Vite Documentation.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 9, September 2026
DOI: 10.17148/IJARCCE.2026.15908

BIOGRAPHY

Akshata Mahishawadagi is a postgraduate student pursuing the Master of Computer Applications (MCA). Her academic interests include full-stack web development, database-driven applications, and educational technology.

