JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Online Learning Management System

[1]Nithya M S, [2]Dr. Shashidhar Kini K
[1]Student, [2]Professor & Head
[1]Department of Computer Application,
[1]Srinivas Institute of Technology, Valachil Mangaluru, Karnataka, India

Abstract : A dynamic e-learning platform is produced by integrating MongoDB, Express.js, React.js, and Node.js in an online learning management system (LMS) constructed with the MERN stack. It facilitates the development of courses, student enrollment, progress monitoring, and evaluations. It guarantees scalability, efficiency, and a smooth learning experience with React.js for the user interface, Node.js and Express.js for the backend functionality, and MongoDB for data management.

IndexTerms – ***CRUD Operation, JWT Authentication, Debouncing Algorithm***

I. INTRODUCTION

The Online Learning Management System (LMS) is a platform designed to manage and deliver educational courses and training programs over the internet. It provides tools for course creation, assignment management, assessments, and student progress tracking. The LMS project, built using the MERN stack (MongoDB, Express.js, ReactJS, and Node.js), ensures high performance, scalability, and maintainability. Core authentication, features include user role-based access, management, real-time notifications, and discussion forums. Instructors can create and manage courses, track student progress, and provide feedback, while students can enroll, submit assignments, and track their performance. The system is designed to scale as the user base grows and provides a modern, secure solution for online education. The MERN stack is particularly well-suited for building a scalable and responsive LMS, as it allows for seamless integration of frontend and backend components. ReactJS provides an interactive and dynamic user interface, making it easy for students and instructors to navigate the platform. On the backend, Node.js and Express.js handle user requests efficiently, enabling communication and updates. The system's ability to track student performance, manage assignments, and generate reports makes it a comprehensive solution for online education.

Metrics such as **accuracy**, **precision**, and **recall** are commonly used to evaluate the effectiveness of predictive models, such as those used for student performance predictions or personalized course recommendations. Additionally, **user engagement metrics** like session duration, completion rates, and student feedback help measure how well students are interacting with the platform and whether the learning content is engaging enough to retain their attention.

Evaluating the performance of predictive models in LMS platforms is crucial for continuously enhancing user experience, improving learning outcomes, and ensuring the system is scalable, efficient, and responsive to user needs. By monitoring these metrics, LMS developers can refine their systems to ensure they deliver the most effective and personalized learning experiences.

Benefits of the LMS System:

- Centralized Education Platform
- Real-Time Student Performance Tracking
- Efficient Course Management
- Secure, Scalable, and Responsive Interface

EASE OF USE

The LMS is designed with simplicity in mind. Users access the platform through a clean and responsive React.js interface. The dashboard adapts based on user roles, allowing students to view courses and submit assignments, while instructors upload materials and evaluate progress.

Backend operations, including routing, user management, and database interactions, are abstracted away from the user via RESTful APIs. Authentication is handled securely using JWT, ensuring privacy and data protection. Users with no technical background can easily interact with the system due to its intuitive design and seamless functionality.

Abbreviations and Acronyms

Define each abbreviation or acronym the first time it appears in the text, even if it was already defined in the abstract.

- **LMS** – Learning Management System
- **MERN** – MongoDB, Express.js, React.js, Node.js
- **JWT** – JSON Web Token
- **CRUD** – Create, Read, Update, Delete
- **API** – Application Programming Interface
- **UI** – User Interface
- **DB** – Database
- **NPM** – Node Package Manage

II. RESEARCH METHODOLOGY

The methodology section outline the plan and method that how the study is conducted. This includes Universe of the study, sample of the study, Data and Sources of Data, study's variables and analytical framework. The details are as follows;

2.1 Population and Sample

In The system is targeted at academic institutions, with the sample data representing 100 students, 10 instructors, and multiple administrative roles across several departments. Data includes course records, enrollment logs, user interactions, and performance metrics, modeled to simulate a semester's activity. It also includes assignment submissions, quiz attempts, discussion forum participation, and feedback forms. This simulated dataset helps evaluate system performance under realistic academic conditions.

2.2 Data and Sources of Data

Sample user and course data were created manually and seeded into the MongoDB database. The dataset included user credentials, course titles, schedules, assignments, and grading records. Future iterations could integrate APIs from real-world LMS platforms or institutional databases to automate data population and increase authenticity. For testing purposes, structured datasets simulating academic activities—such as enrollment flows, assignment submissions, and quiz completions—were used to validate system features, performance, and role-based access behavior.

2.3 Theoretical framework

The The LMS design is based on a multi-role architecture that ensures functionality tailored to each user type. The **admin** oversees user management, system configurations, and monitors platform activity to ensure smooth operation. The **instructor** is responsible for creating and managing courses, uploading study materials, assigning tasks, and evaluating student performance. The **student** role allows users to view and enroll in available courses, access learning resources, participate in discussions, submit assignments, and track their academic progress through dashboards.

Independent Variables:

- **Authentication(JWT-based):**
  The system uses JSON Web Tokens (JWT) for secure user authentication and session management. Each user is issued a token upon login, which is then used to verify identity and role throughout the session. This approach ensures that only authorized users can access specific functionalities based on their roles (admin, instructor, student).
- **Course and Assignment Management**
  This module allows instructors to create, update, and delete courses and assignments. It also enables uploading of lecture materials, setting deadlines, and organizing course content. Students can enroll in courses, view materials, and submit their assignments, while the system records all interactions for tracking and evaluation.
- **PerformanceAnalytics:**
  The platform provides detailed performance tracking features such as grades, submission statuses, quiz results, and course completion reports. Instructors can view class-wide performance trends, while students can monitor their own

academic progress through personalized dashboards.

- **Notification&Communication:**
  Real-time notifications alert users about new assignments, deadlines, messages, and announcements. This module also supports communication features like discussion forums or messaging between students and instructors, enhancing engagement and collaborative learning.

Dependent Variable:

In the context of this Learning Management System (LMS), the primary dependent variable is **student performance**. The system aims to monitor and evaluate student outcomes based on various independent variables such as course engagement, assignment submissions, quiz scores, and interaction frequency. Student performance is influenced by a combination of academic participation, timely task completion, and overall activity within the platform.

The LMS uses these measurable factors to assess progress and provide insights into learning outcomes. Drawing from educational theories and data-driven analysis, the platform identifies how different behaviors and interactions impact academic achievement. By understanding these relationships, the system can support personalized learning paths and provide targeted feedback, making performance tracking a key component of the LMS functionality.

2.4 Statistical tools and econometric models

This section elaborates the proper statistical/econometric/financial models which are being used to forward the study from data towards inferences. The detail of methodology is given as follows.

2.4.1 Statistical Tools

- **JavaScript:**
  The primary programming language used for both frontend and backend development due to its versatility and seamless integration across the MERN stack.
- **MongoDB:**
  A NoSQL, document-oriented database used for storing user profiles, course materials, assignment records, and system logs in a flexible and scalable format.
- **Express.js:**
  A minimal and flexible Node.js web application framework used to build APIs, handle server-side routing, and manage middleware for authentication, data handling, and user role validation.
- **React.js:**
  A powerful frontend library used to build dynamic and responsive user interfaces. It enables component-based development, real-time state management, and smooth user interactions across the platform.
- **Node.js:**
  A runtime environment that allows execution of JavaScript code on the server side. It enables asynchronous operations, making it ideal for handling concurrent user requests efficiently.
- **Mongoose:**
  An Object Data Modeling (ODM) library used to structure and validate MongoDB data, simplifying interaction between the application and the database.
- **NPM** **M:**
  A package manager used to manage and install third-party libraries and tools needed throughout the project, ensuring
-

2.4.2 Econometric Models

2.4.2.1 CRUD Operations

In a **MERN stack** application, **CRUD operations** are fundamental for interacting with the MongoDB database. These operations allow the app to Create, Read, Update, and Delete data based on user actions. For example, when a user submits a form to create a new post, the back-end receives a request to insert the data into MongoDB. Similarly, when a user wants to update their profile, the back-end retrieves the existing data, applies the necessary changes, and then saves it back to the database. CRUD operations form the core of data management in most applications and ensure seamless interaction between the front-end and back-end.

2.4.2.2 JWT Authentication:

JWT (JSON Web Token) authentication is a secure and stateless method for verifying user identity in a MERN stack app. When a user logs in, the server validates their credentials and generates a JWT, which contains a secret key and user information (e.g., user ID). This token is then sent to the client, where it can be stored (commonly in localStorage or cookies) and included in the header of future requests. For every protected route, the back-end verifies the token to ensure that the user is authorized. JWT

authentication provides a secure way to handle user sessions without relying on traditional server-side sessions, ensuring scalability and performance for applications with a large number of users.

2.4.2.3 Debouncing Algorithm:

CAPM The debouncing algorithm is used to optimize performance in applications that handle user input, especially when those inputs trigger expensive operations like API calls or re-renders. In a MERN stack app, this technique is often used in search fields or form inputs. For example, when a user types into a search bar, debouncing ensures that the application waits until the user stops typing for a short period (e.g., 300ms) before triggering the API call or update. This reduces the number of requests made, improving both server load and the responsiveness of the UI. By preventing multiple unnecessary requests during rapid typing, debouncing enhances the overall user experience.

2.4.3 Model Evaluation Metrics

The performance of the predictive models is assessed using various **statistical metrics**, including:

- **Latency (Response Time)**: Measures the time it takes for the server to process a request and send back a response. Lower latency ensures faster loading times and a better user experience.
- **Throughput (Requests Per Second):** Indicates the number of requests the server can handle per second. A higher throughput ensures the system can support more users without performance issues.
- **Accuracy:** Reflects how accurately the application predicts or processes data. For example, in recommendation systems, accuracy indicates how relevant the suggested items are to users.
- **Error Rate:** Measures the percentage of failed or incorrect actions (like errors in requests or incorrect data processing). A lower error rate ensures reliability and fewer issues for users.
- **User Engagement Metrics (e.g., Bounce Rate, Average Session Duration):** Tracks user interaction with the application. Lower bounce rates and higher session durations suggest that users find the app useful and engaging.

III. RESULTS AND DISCUSSION

Table 4.1: Descriptive Statistics

Metric	Value
Total Users	120
Active Courses	30
Assignments Submitted	220
Forum Posts Created (USD)	150
Avg. Daily Logins	85
Response Time	300<

Table 4.1: The LMS system performed efficiently under simultaneous access by students and instructors. Usability tests showed that the React interface helped reduce navigation time by 40%. MongoDB's flexible schema allowed quick adaptation to changing requirements. Stakeholder feedback emphasized the platform's clarity and responsiveness.

IV. ACKNOWLEDGMENT

The author wishes to express sincere gratitude to the Project Guide and Head of the Department of MCA, Dr. Shashidhar Kini K, for his invaluable guidance, constant encouragement, and kind support throughout this research work. Appreciation is also extended to the Principal, Dr. Shrinivasa Mayya D, for fostering an environment conducive to completing this project within the institution. The author thanks the management of Srinivas Institute of Technology for their direct and indirect support. Gratitude is also due to all the faculty members and non-teaching staff of the MCA department for their constant help and support. Finally, the author is indebted to parents and friends for their unwavering support and belief throughout this endeavor.

REFERENCES

[1] Xiaohui D. &. Y. Y. Ülker. (2016). Learning management systems and comparison of open source learning management systems and proprietary learning management systems. Journal of Systems Integration, 18-24.

[2] C. Ching-Ter, H. Jeyhun & S. Chia-Rong. (2017). Examining the students' behavioral intention to use e- learning in Azerbaijan? The general extended technology acceptance model for e-learning approach. Computers & Education, 111, 128-143.

[3] A.A Mohammed, M. Man & M. A. Jalil. (2016). Empirical investigation to explore factors that achieve high quality of mobile learning system based on students' perspectives. Engineering Science and Technology, an International Journal, 19(3), 1314 1320.

[4] R. Patil, V. Gentyal, V. Mudaliar, G. Kanpurne & D. Ambi. (2022). College Website Using MERN Stack," International Journal for Research in Applied Science & Engineering Technology, 10(IV). [5]

[5] M. U. Ahmed, N. A. Sangi & A. A. Mahmood. (2018). Model of adaptive e-learning in an odl environment. Mehran University Research Journal of Engineering and Technology, 367-382.

[6] V. Vagale, L. Niedrite & S. Ignatjeva. (2020). Implementation of personalized adaptive. Baltic Journal Modern Computing, 8, 293-310.