Project Report

Learning Management System

Team Lead: Muhammad Hamza A.B
Project Contributers: Magesh D, Sujathan V, Stalin Santhosh G

Muthukumaran Institute of Technology
IVth Year
NaanMudhalvan Project Report

Table of Contents

Introduction:

Our project, *Lemida*, is a Learning Management System (LMS) created using the MERN stack. The name *Lemida* stands for learning and growth, representing the core idea of this platform. The LMS provides an online environment where instructors can upload and manage courses, students can learn and track their progress, and admins can oversee users and course content.

To make the platform user-friendly and efficient, we incorporated a Razorpay payment integration, enabling secure transactions for premium courses. Our focus was on delivering a responsive and elegant design with smooth functionality.

Project Description:

The LMS has three main roles:

1. **Students:**
    - Can register, log in, and browse available courses.
    - Track learning progress through their account dashboard.
    - Purchase premium courses using the Razorpay payment gateway.
2. **Instructors:**
    - Upload and manage course materials.
    - View student engagement with their courses.
3. **Admins:**
    - Monitor all users and activities.
    - Elevate users to admin roles.
    - Edit, delete, or add courses.

This structured approach ensures a seamless experience for all types of users.

Technology Stack:

- **MongoDB:** For storing user data, course details, and progress tracking.
- **Express.js:** Backend framework to handle API requests and business logic.
- **React.js:** For a dynamic and responsive user interface.
- **Node.js:** Server-side environment to connect backend and frontend.
- **Razorpay API:** Integrated for secure payment transactions.

System Features:

- **User Roles:**
  - Students, Instructors, and Admins, each with specific features.
- **Payments:**
  - Razorpay integration for course purchases.
- **Responsive Design:**
  - Optimized UI for desktop and mobile users.
- **Progress Tracking:**
  - Students can view and track their learning journey.
- **Admin Panel:**
  - Advanced tools for managing users and courses.

System Design and Architecture:

- **Frontend:** Developed using React.js for creating dynamic pages and user-friendly navigation.
- **Backend:** Node.js and Express.js handle API endpoints and server-side logic.
- **Database:** MongoDB stores all user roles, course data, and payment details securely.
- **Payment Flow:** Razorpay ensures secure payment transactions, with clear tracking of successful purchases.

OTP Feature with Gmail Integration:

Our LMS incorporates a secure and user-friendly OTP verification system using Gmail. Whenever a user registers or requests a password reset, a One-Time Password (OTP) is sent to their registered email. This ensures the authenticity of users and enhances security.

- **How It Works**:
  - The system generates a unique OTP and emails it to the user.
  - Users must enter the OTP within a limited time to complete the registration or password reset process.
  - This feature adds a layer of protection and ensures only verified users can access their accounts.
- **Why Gmail?**
  - Gmail's reliable and fast email delivery makes it an ideal choice for sending sensitive information like OTPs.

This feature ensures the LMS remains secure and compliant with modern user authentication standards.

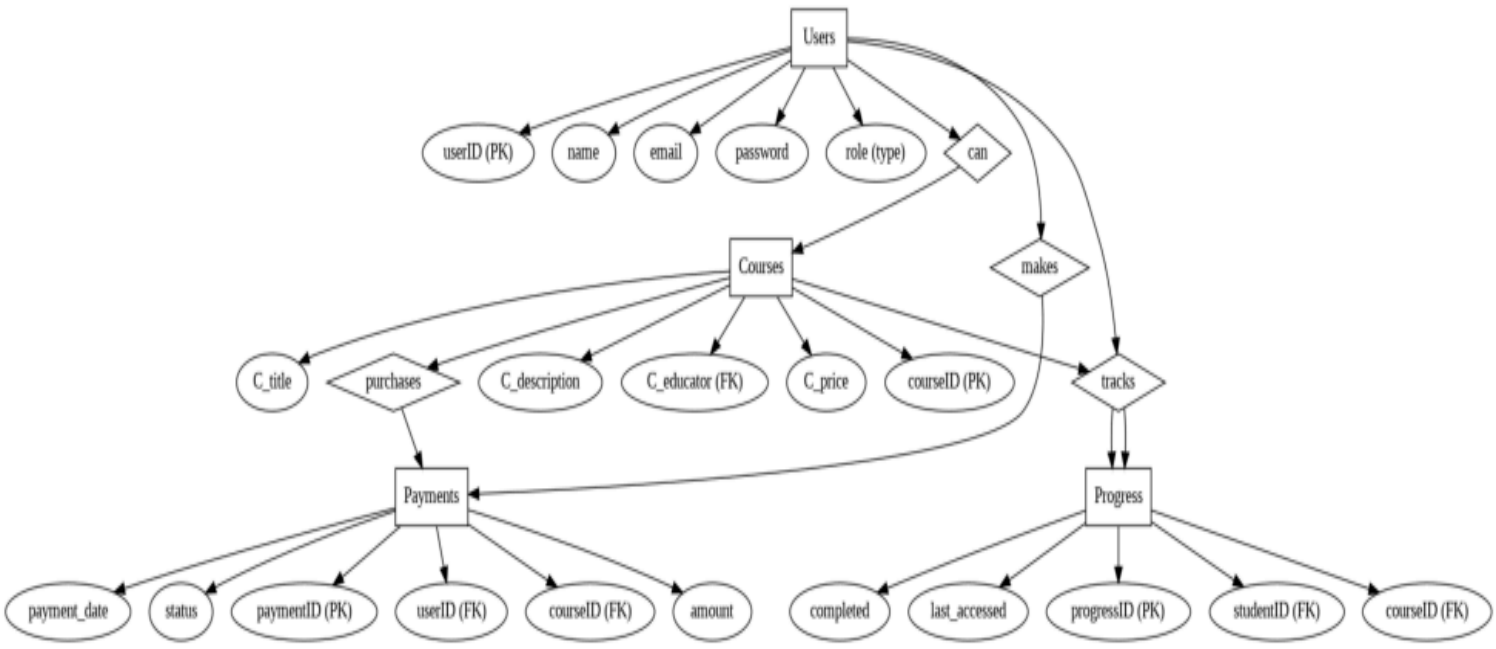
Development Process:

1. **Requirement Analysis:** Understanding project needs and breaking them into tasks.
2. **UI/UX Design:** Creating a clean and responsive interface using React.js.
3. **Backend Development:** Building APIs for user authentication, course management, and payment handling.
4. **Database Design:** Structuring collections for users, courses, and payments.
5. **Integration:** Connecting the Razorpay API and implementing role-based access.
6. **Testing and Debugging:** Ensuring each feature works smoothly across devices.

Team Contributions:

- **Muhammad Hamza A.B (Team Lead):**
  - Oversaw the entire project and implemented backend features like payments and role management.
- **Magesh D:**
  - Designed the frontend pages and ensured responsiveness.
- **Sujathan V:**
  - Worked on database management and authentication systems.
- **Stalin Santhosh G:**
  - Handled admin panel functionalities and testing.

ER Diagram:



Project Structure:

```
+frontend
|   .env
|   .gitignore
|   index.html
|   package-lock.json
|   package.json
|   vercel.json
|   vite.config.js
├───public
|   └───assets
└───src
    ├───admin
    |   ├───coursecard
    |   ├───Courses
    |   ├───Dashboard
    |   ├───Users
    |   └───Utils
    ├───components
    |   ├───coursecard
    |   ├───footer
    |   ├───header
    |   ├───loading
    |   └───testimonials
    ├───context
    └───pages
        ├───about
        ├───account
        ├───auth
        ├───coursedescription
        ├───courses
        ├───coursestudy
        ├───dashbord
        ├───home
        ├───lecture
        └───paymentsuccess
```

```
+server
|    .env
|    .gitignore
|    index.js
|    package-lock.json
|    package.json
|    vercel.json
|
├───controllers
|        admin.js
|        course.js
|        user.js
|
├───database
|        db.js
|
├───middlewares
|        isAuth.js
|        multer.js
|        sendMail.js
|        TryCatch.js
|
├───models
|        Courses.js
|        Lecture.js
|        Payment.js
|        Progress.js
|        User.js
|
├───routes
|        admin.js
|        course.js
|        user.js
|
└───uploads
```

Technical Details:

System Requirements

- **Hardware Requirements**:
  - Processor: Minimum dual-core
  - RAM: At least 4 GB
  - Storage: 1 GB free space
- **Software Requirements**:
  - Node.js (version >= 18)
  - MongoDB (cloud-based, e.g., MongoDB Atlas)
  - Git
  - Any browser (Chrome recommended)

Installation Guide

For Developers:

- Prerequisites: Node.js, npm/yarn, MongoDB, and Git installed.
- Steps:
  1. Clone the repository:
     ```
     gh repo clone ab-muhammad-hamza/nm-lms
     ```
  2. Navigate to the project directory:
     ```
     cd nm-lms
     ```
  3. Install dependencies for the frontend:
     ```
     cd frontend
     npm install
     ```
  4. Install dependencies for the backend:
     ```
     cd ../backend
     npm install
     ```
  5. Create MongoDB cluster and paste the URL in .env
  6. Run the application.

For Users:

- Provide a pre-built version or installation steps (e.g., hosted on a platform)

Running the Project

Frontend

```
cd frontend
npm run dev
```

Backend (Node.js + Express):

```
cd frontend
npm run dev
```

Environment Setup

```
MONGODB_URI = “Your Mongo Cluster URI”
Password = "Email app password from gmail"
Gmail = "your email Id associated with gmail"
RzpClient = "RazorPay Client ID"
RzpSecret = "RazorPay Secret ID"
```

Database Configuration

The schemes and configuration of the Database will be generated while running the script. It will be seeded and the content will be added automatically as MongoDB is an JSON based database.
Here are the tables (collections) which will be created:

- courses
- lectures
- payments
- progresses
- Users

To know more about the schema, you can get into **server > models** where there will be schema for all different collections of the project.

Razorpay Payment Integration

RazorPay is integrated in this project so that students can buy any courses and instructors will get paid from it. You can create an account in RazorPay, complete the KYC process and create a project to get the RazorPay client ID and the Secret key which then can be updated on .env file of server.

Authentication and Authorization

- JWT-based authentication implementation.
- Role-based access control (e.g., student, instructor, admin).

Features Overview

- **Instructor Features:**
  - Create, edit, and delete courses.
  - Track enrolled students.
- **Student Features:**
  - Enroll in courses.
  - Track progress.
- **Admin Features:**
  - Manage users (promote to admin, delete users).
  - Manage courses (add, edit, delete).

API EndPoints

- **POST /course/new** - Create a new course (Admin only).
- **POST /course/:id** - Add lectures to a course (Admin only).
- **DELETE /course/:id** - Delete a course (Admin only).
- **DELETE /lecture/:id** - Delete a lecture (Admin only).
- **GET /stats** - Fetch dashboard statistics (Admin only).
- **PUT /user/:id** - Update a user's role.
- **GET /users** - Get all registered users (Admin only).
- **GET /course/all** - Fetch all available courses.

- **GET /course/:id** - Get details of a specific course.
- **GET /lectures/:id** - Get all lectures for a course.
- **GET /lecture/:id** - Get details of a specific lecture.
- **GET /mycourse** - Get all courses the user has enrolled in.
- **POST /course/checkout/:id** - Initiate payment for a course.
- **POST /verification/:id** - Verify payment status.
- **POST /user/register** - Register a new user.
- **POST /user/verify** - Verify user email.
- **POST /user/login** - Login user.
- **GET /user/me** - Fetch logged-in user's profile.
- **POST /user/forgot** - Send a password reset link.
- **POST /user/reset** - Reset the password.
- **POST /user/progress** - Add progress for a course.
- **GET /user/progress** - Fetch the user's progress.

Gmail Integration for OTP Verification

The LMS project integrates Gmail to send One-Time Passwords (OTPs) for secure user authentication during registration, email verification, and password recovery.

- **Library Used**: nodemailer
- **How It Works**:
    1. When a user registers or requests a password reset, an OTP is generated on the backend.
    2. The nodemailer library is used to send the OTP to the user's registered email address using Gmail's SMTP server.
    3. The OTP is time-sensitive, ensuring additional security by expiring after a defined duration.

Security Considerations

- Protection against XSS, CSRF, and SQL/NoSQL injection.
- Secure handling of payment data and user authentication.

Challenges Faced:

1. **Payment Integration:** Learning and integrating Razorpay with the LMS.
2. **Authentication and Role Management:** Ensuring secure access for each user type.
3. **Responsive Design:** Adapting the UI to different devices without breaking functionality.

Testing and Results:

- **Unit Testing:** Tested individual features like course upload, progress tracking, and payments.
- **Integration Testing:** Verified proper communication between frontend and backend.
- **User Testing:** Feedback from classmates helped us refine the system.

All tests confirmed the system's stability and usability.

Future Scope:

1. Add interactive features like quizzes and certificates.
2. Expand the Razorpay integration to include multiple payment options.
3. Build analytics dashboards for instructors and students.

Conclusion:

The **Lemida LMS** project is a comprehensive learning platform that demonstrates the seamless integration of modern technologies, user-centric features, and secure functionality. Built on the MERN stack, this platform enables instructors to create and manage courses effortlessly while providing students with an engaging and interactive learning experience. The incorporation of essential features like Razorpay payment integration, OTP-based email verification via Gmail, and user progress tracking ensures a robust and scalable system.

This project highlights the importance of responsive design, ensuring compatibility across devices to cater to a diverse user base. The admin functionalities further enhance the platform's usability by allowing easy management of users and courses. The secure and efficient use of middleware for authentication and authorization ensures that the application upholds high standards of data privacy and access control.

From the technical perspective, the Lemida LMS demonstrates the ability to manage multiple stakeholders (admins, instructors, and students) within a unified system. The project showcases advanced backend capabilities, including CRUD operations, payment gateway integration, and dynamic routing for REST APIs, as well as frontend expertise with the use of React and Vite for a fast and seamless user interface.