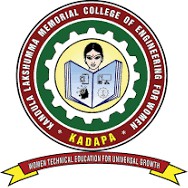
##### K.L.M COLLEGE OF ENGINEERING FOR WOMEN

(APPROVED BY AICTE AND AFFILIATED TO J.N.T.U. ANANTAPURAMU)

Kadapa - 516003

**Academic Year: 2025–2026**



DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

**A MAJOR PROJECT DOCUMENTATION ON**

**YOUR CENTER FOR SKILL ENHANCEMENT**

Submitted in partial fulfilment of the requirements for the award of the degree

**BACHELOR OF TECHNOLOGY**

in

**ELECTRONICS & COMMUNICATION ENGINEERING**

Submitted by

TEAM ID : LTVIP2025TMID49959

M. Sudha Rani 223h1a0423

G. Bharathi 223h1a0415

S. Sindhuja 223h1a0431

T. Nageswari 223h1a0439

M. Bindu Sai 233h5a0404

G.B.Dharani 223h1a0413

**Under the esteemed guidance of**

Mrs.C Sujana

Assistant Professor, Department of ECE

# **Online Learning Platform (OLP)**

## **Full-Stack MERN Application - Project Documentation**

## 1. Introduction

An **Online Learning Platform (OLP)** is a digital environment that provides tools and resources to facilitate education over the internet. These platforms have surged in popularity due to their flexibility and accessibility, catering to learners from diverse backgrounds.

Key characteristics of OLP include:

* **User-Friendly Interface**  
  Intuitive layouts ensure users can navigate and engage with minimal technical expertise.
* **Course Management**  
  Instructors can create, upload, and manage courses; students can enroll, consume content, and track progress.
* **Interactivity**  
  Features such as discussion forums, chat systems, and live webinars foster engagement and collaborative learning.
* **Certification**  
  Learners can earn verifiable certificates upon completing courses, enhancing career prospects.
* **Accessibility**  
  The platform supports a range of devices, including desktops, tablets, and smartphones.
* **Self-Paced Learning**  
  Learners proceed through content at their own speed.
* **Monetization Options**  
  Supports both free and paid content, with integrated payment solutions.

### Scenario-based Case Study

**Scenario: Learning a New Skill**

* **User Registration**  
  Sarah, a student eager to learn web development, registers on the OLP platform, providing her email and creating a password.
* **Browsing Courses**  
  Upon login, she explores a visually engaging catalog of courses, categorized by topic and difficulty.
* **Enrolling in a Course**  
  Sarah enrolls in Web Development Fundamentals after reviewing its description and syllabus.
* **Learning Progress**  
  The platform tracks Sarah’s progress across modules, allowing her to resume seamlessly.
* **Interactivity and Support**  
  She participates in discussion forums and live webinars for collaborative learning.
* **Certification**  
  After completing assignments and passing the final exam, she downloads her digital certificate.
* **Paid Courses**  
  Sarah upgrades to an advanced course via the integrated payment system.
* **Teacher’s Role**  
  John, an experienced web developer, uploads new content and manages enrollments.
* **Admin Oversight**  
  Admin users monitor activity, manage course listings, and handle support issues.

## 2. Project Overview

* **Purpose**  
  The OLP enables a seamless digital learning environment for students, teachers, and administrators. It facilitates course management, secure payments, progress tracking, and digital certification, promoting self-paced learning.
* **Key Features**
  + User authentication with JWT security.
  + Role-based dashboards for Students, Teachers, and Admins.
  + Course management (CRUD operations).
  + Interactive learning via discussion features.
  + Payment integration for premium courses using Stripe.
  + Automated PDF certificate generation for course completion.
  + Responsive and modern UI.

## 3. Technical Architecture

The OLP follows a robust **client-server architecture**, integrating modern web technologies:

### Frontend (React + Vite)

* Developed with **React.js** for component-driven architecture and modern UI development.
* Uses **Vite** as a build tool for rapid development and efficient production builds.
* Integrates libraries:
  + **Material UI**, **Bootstrap**, **Ant Design**, **mdb-react-ui-kit** for consistent UI components.
  + **Axios** for HTTP communication with backend APIs.
* Implements routing via **React Router**.
* Supports state management within components and potentially via Context APIs.

### Backend (Node.js + Express.js)

* Built using **Express.js**, providing scalable server-side logic.
* Handles:
  + API routing.
  + User authentication via JWT.
  + Course and user CRUD operations.
  + Certificate generation logic.
* Uses **Mongoose** to interface with MongoDB, enabling schema-based data modeling.
* Environment configurations managed via .env files.

### Database (MongoDB)

* Stores user and course data in a document-oriented format.
* Collections:
  + **Users**
    - \_id
    - name
    - email
    - password
    - type (student, teacher, admin)
  + **Courses**
    - \_id
    - C\_educator
    - C\_categories
    - C\_title
    - C\_description
    - sections
    - C\_price
    - enrolled (student IDs)

### ER Diagram

Relationships:

* Each course is linked to a user (teacher) via userID.
* Students enroll in courses, recorded via an array of enrolled IDs.

## 4. Prerequisites

### Frontend

* Vite
* React.js
* HTML/CSS/JavaScript
* UI Libraries:
  + Bootstrap
  + Material UI
  + Ant Design
  + mdb-react-ui-kit

### Backend

* Node.js & npm
* Express.js
* MongoDB (local or cloud via Atlas)
* Mongoose
* JSON Web Token (jsonwebtoken)
* bcryptjs (password hashing)
* Multer (file handling for uploads)
* dotenv (environment variables)
* Stripe API for payments

## 5. Installation and Setup

### MongoDB

* Run MongoDB locally:

bash

CopyEdit

"C:\Program Files\MongoDB\Server\8.0\bin\mongod.exe"

### Backend

1. Navigate to backend folder:

bash

CopyEdit

cd backend

1. Install dependencies:

bash

CopyEdit

npm install

npm install pdfkit stripe

1. Start backend server:

bash

CopyEdit

npm start

### Frontend

1. Navigate to frontend folder:

bash

CopyEdit

cd frontend

1. Install dependencies:

bash

CopyEdit

npm install

npm install @emotion/react @emotion/styled

1. Start frontend with Vite:

bash

CopyEdit

npm run dev

### Environment Configuration

Create a .env file in backend:

ini

CopyEdit

MONGO\_URI=<Your MongoDB connection string>

JWT\_SECRET=<Your JWT secret>

STRIPE\_SECRET\_KEY=<Your Stripe secret key>

## 6. Project Structure

pgsql

CopyEdit

dileep/

│

├── backend/

│ ├── models/

│ │ ├── User.js

│ │ └── Course.js

│ ├── routes/

│ │ ├── authRoutes.js

│ │ ├── courseRoutes.js

│ │ ├── userRoutes.js

│ │ ├── certificateRoutes.js

│ │ └── paymentRoutes.js

│ ├── utils/

│ │ └── certificate.js

│ ├── certificates/ # generated PDFs

│ ├── .env

│ ├── .env.example

│ ├── package.json

│ └── index.js

│

└── frontend/

├── src/

│ ├── pages/

│ │ ├── Home.jsx

│ │ ├── Login.jsx

│ │ ├── Register.jsx

│ │ └── Dashboard.jsx

│ ├── components/

│ │ └── Navbar.jsx

│ ├── App.jsx

│ ├── main.jsx

│ ├── routes.js

│ └── setupProxy.js

├── public/

├── index.html

├── package.json

## 7. Application Flow

### User Roles and Responsibilities

* **Teacher**
  + Add and edit courses.
  + Delete courses if no enrollments exist.
  + Upload sections and manage content.
* **Student**
  + Browse courses.
  + Enroll in free or paid courses.
  + Resume courses from previous progress.
  + Access and download completion certificates.
  + Filter courses by name, category, etc.
* **Admin**
  + Oversee users and courses.
  + Manage and delete users or courses as needed.
  + Monitor enrolled students and platform activity.

## 8. Running the Application

* **Frontend:**

bash

CopyEdit

cd frontend

npm run dev

* **Backend:**

bash

CopyEdit

cd backend

npm start

* **MongoDB:**

bash

CopyEdit

"C:\Program Files\MongoDB\Server\8.0\bin\mongod.exe"

## 9. API Documentation

### Authentication

* POST /api/auth/register  
  Registers a new user.
* POST /api/auth/login  
  Authenticates user, returns JWT.

### User Management

* GET /api/users  
  List users (admin only).
* PUT /api/users/:id  
  Update user info.
* DELETE /api/users/:id  
  Delete user.

### Course Management

* POST /api/courses  
  Create a new course.
* GET /api/courses  
  List all courses.
* GET /api/courses/:id  
  Fetch single course details.
* PUT /api/courses/:id  
  Update course.
* DELETE /api/courses/:id  
  Delete course.
* POST /api/courses/enroll/:id  
  Enroll a student.

### Certificates

* POST /api/certificates/generate  
  Generate PDF certificate.

### Payments

* POST /api/payments/create-checkout-session  
  Create Stripe payment session.
* POST /api/payments/webhook  
  Stripe webhook for payment events.

## 10. Authentication

* JWT-based authentication secures backend routes.
* Tokens stored client-side.
* Role-based authorization ensures proper access for admin, teacher, and student users.

## 11. User Interface

The platform offers a modern, responsive user experience:

* **Landing Page**  
  Attractive introduction and course browsing.
* **Register Page**  
  User signup with validations.
* **Login Page**  
  Secure login form.
* **Admin Dashboard**  
  Overview of users and courses.
* **Teacher Dashboard**  
  Tools for course creation and management.
* **Student Dashboard**  
  Enrolled courses, progress tracking, certificate downloads.

(Note: Screenshots and demo video references provided in your assets.)

## 12. Testing

### Backend

* Manual API testing via Postman or Insomnia.
* Potential for automated testing using Jest or Mocha.

### Frontend

* Manual end-to-end testing.
* Tools like React Testing Library or Cypress could be integrated for UI tests.

## 13. Screenshots or Demo

* **Landing page**  
  (Insert image or link)
* **Register page**  
  (Insert image or link)
* **Login page**  
  (Insert image or link)
* **Admin Dashboard**  
  (Insert image or link)
* **Teacher Dashboard**  
  (Insert image or link)
* **Student Dashboard**  
  (Insert image or link)
* **Demo Video**  
  project-implementation.mp4 (provide actual link if available)
* **Code Repository**  
  (Provide link to code drive or GitHub repo)

## 14. Known Issues

* Stripe webhook testing requires live environment or test mode setup.
* Certificate layouts may vary based on length of user names or course titles.
* Limited automated testing coverage at current stage.

## 15. Future Enhancements

* Implement automated unit and integration testing for backend and frontend.
* Add video upload and streaming features for courses.
* Enable social logins (Google, Facebook, etc.).
* Enhance analytics for course performance tracking.
* Improve certificate design for branding and personalization.
* Implement notification system for user engagement.
* Optimize for Progressive Web App (PWA) capabilities.