LearnHub: Your Center for Skill Enhancement

## 1. Introduction

**Project Title:** Online Learning Platform (OLP) using MERN Stack  
**Team ID:** LTVIP2025TMID54182  
**Team Size:** 4

**Team Members and Roles:**

| **Name** | | **Role** |
| --- | --- | --- |
| **Naveen Koppisetti** | UI/UX Designer & Tester, Full Stack Developer | | |
| **Mohammad Aqbal** | | Team Leader |
| **Nakkanaboyina Lakshmi Sunitha** | | Frontend Developer |
| **Nalluri Anusha** | | Backend Developer |
|  | |  |

## 2. Project Overview

### **Purpose**

The **Online Learning Platform (OLP)** is a modern, full-stack web application designed to bridge the gap between learners and educators by delivering educational content via the internet. Built using the MERN stack (MongoDB, Express.js, React.js, Node.js), the platform provides a scalable and interactive environment where students can access quality learning resources anytime, anywhere.

This platform was developed in response to the growing demand for flexible learning solutions, especially in the post-pandemic digital era. It empowers learners with self-paced modules, interactive content, and certification opportunities, while also giving educators robust tools for course creation, student tracking, and performance monitoring.

The primary goal is to simulate a real-world online classroom environment, making learning more accessible, engaging, and personalized. Whether it’s a student looking to acquire a new skill or an instructor wanting to share knowledge, this platform provides a structured, efficient, and intuitive medium to achieve those objectives.

**Objectives**

* To provide an intuitive and user-friendly interface for students and teachers.
* To implement role-based dashboards with different access controls for Admins, Educators, and Students.
* To facilitate course creation, enrollment, and certification with ease.
* To support both free and paid course models with integrated payment mechanisms.
* To ensure secure user authentication and session management.
* To support scalability, allowing thousands of users to access learning materials without performance degradation.

### **Key Features**

* **User Authentication and Role Management:**  
  JWT-based secure login system with role-based dashboards for Admin, Teacher, and Student.
* **Course Creation and Management:**  
  Teachers can create, update, and delete courses, add sections, and track student enrollments.
* **Student Interaction:**  
  Students can browse, enroll, track progress, participate in discussions, and complete courses at their own pace.
* **Certification System:**  
  Automatic generation of certificates upon successful course completion.
* **Admin Control Panel:**  
  Centralized dashboard for monitoring all users, courses, transactions, and system logs.
* **Responsive UI:**  
  Built using React with libraries like Bootstrap, Material UI, and MDB Kit for a responsive and mobile-friendly interface.
* **API-Based Architecture:**  
  RESTful APIs connect the frontend to the backend, enabling smooth and structured data communication.
* **MongoDB Integration:**  
  All application data, including user profiles, course information, enrollments, and certificates, are managed via MongoDB for fast and reliable storage.
* **Search and Filter Options:**  
  Advanced filtering capabilities allow students to discover courses based on category, name, or instructor.
* **Payment Gateway Integration (Optional/Future Scope):**  
  Supports enrollment into premium courses via a secure payment system.

3. Architecture

The architecture of the **Online Learning Platform (OLP)** follows a **modular, scalable, and maintainable client-server model**, utilizing the **MERN stack**—MongoDB, Express.js, React.js, and Node.js. This architecture is designed to ensure a seamless user experience, efficient data processing, and high system responsiveness even under heavy traffic. Each layer of the stack contributes to specific responsibilities, forming a cohesive and robust full-stack web application.

### **Frontend: React.js (Client-Side)**

The frontend is built using **React.js**, a powerful JavaScript library for creating dynamic and responsive user interfaces. To enhance development speed and performance, **Vite** is used as the build tool, offering lightning-fast hot module replacement and optimized bundling.

Key components of the frontend architecture include:

* **Component-Based Structure**: React components are reusable and modular, making development and maintenance easier.
* **Routing**: React Router is used to manage navigation across multiple views like home, login, register, dashboards, course pages, etc.
* **State Management**: React’s useState, useEffect, and Context API manage state across components.
* **HTTP Requests**: The frontend communicates with the backend via **Axios**, sending authenticated requests using JWT tokens.
* **Styling**: The user interface is built using **Bootstrap**, **Material UI**, and **MDB UI Kit**, ensuring responsiveness and an engaging user experience across devices.

### **Backend: Node.js & Express.js (Server-Side)**

The backend is developed using **Node.js** and **Express.js**, which provides a lightweight, high-performance environment for handling asynchronous operations and RESTful API development.

The responsibilities of the backend include:

* **API Routing**: Organized routes for authentication (/auth), courses (/courses), users (/users), and administrative tasks.
* **Authentication & Authorization**: Middleware using **JWT (JSON Web Tokens)** ensures secure access and role-based permissions (Admin, Teacher, Student).
* **Middleware**: Includes custom error handling, request logging, input validation, and authentication.
* **File Uploads**: Supports file uploads using **Multer** for media such as thumbnails or course documents (if applicable).
* **Environment Management**: Configuration variables such as database URLs, JWT secrets, and server ports are handled via **dotenv**.

### **Database: MongoDB (NoSQL Data Layer)**

The platform uses **MongoDB** as the database, accessed via **Mongoose**, an elegant ODM (Object Data Modeling) library for Node.js.

#### Key Collections:

* **Users Collection**:
  + Fields: \_id, name, email, password, type (student/teacher/admin), and timestamps.
  + Purpose: Stores all user data, authentication details, and role information.
* **Courses Collection**:
  + Fields: \_id, title, description, price, category, educator ID, sections, enrolled[].
  + Purpose: Stores course metadata, including enrolled students and modular content.
* **Enrollments (embedded or linked)**:
  + Tracks the relationship between users and the courses they have enrolled in.
  + Stores progress, certificate status, and timestamps.

MongoDB was chosen for its scalability, flexibility in handling dynamic schemas (like course modules and user progress), and seamless integration with JavaScript-based technologies.

### **Interaction Flow**

1. A user accesses the platform through the React frontend.
2. HTTP requests are sent via Axios to the Express backend.
3. The Express server processes requests and performs logic such as user validation, course fetching, or registration.
4. The server interacts with MongoDB to read/write data.
5. Responses are sent back to the frontend and rendered dynamically in the UI.

### **Benefits of This Architecture**

* **Separation of Concerns**: Each part of the stack is responsible for a specific set of tasks, making the system modular.
* **Scalability**: Easily scalable horizontally or vertically with cloud platforms like AWS, GCP, or Heroku.
* **Maintainability**: Modular design allows teams to independently update the frontend, backend, or database without breaking the entire application.
* **Security**: JWT, input validation, and proper error handling secure both user data and system integrity.

## 4. Setup Instructions (Short)

### **Prerequisites**

* **Node.js & npm** – For running backend and managing packages
* **MongoDB** – For database (local or cloud)
* **Vite** – For fast React frontend
* **Git** – For cloning the repository

### **Installation Steps**

1. **Clone the Repository**

git clone https://github.com/yourusername/online-learning-platform.git

cd online-learning-platform

1. **Install Dependencies**

cd frontend

npm install

cd ../backend

npm install

1. **Set Up Environment Variables**

Create a .env file inside backend/ with:

PORT=5000

MONGO\_URI=mongodb://localhost:27017/olp

JWT\_SECRET=your\_jwt\_secret\_key

1. **Run the Application**

* **Backend:**

cd backend

npm start

* **Frontend:**

cd frontend

npm run dev

## 5. Folder Structure

The project is divided into two main directories: **frontend** and **backend**, following a modular architecture that supports scalability, role separation, and easy maintenance.

### **5.1 Frontend Folder Structure (**/frontend**)**

frontend/

├── node\_modules/ # Dependency packages

├── public/ # Static files like favicon, index.html

├── src/

│ ├── assets/ # Logos, images, icons

│ ├── components/

│ │ ├── admin/ # Admin-specific components

│ │ │ ├── AdminHome.jsx

│ │ │ └── AllCourses.jsx

│ │ ├── common/ # Shared components

│ │ │ ├── AllCourses.jsx

│ │ │ ├── AxiosInstance.jsx

│ │ │ ├── Dashboard.jsx

│ │ │ ├── Home.jsx

│ │ │ ├── Login.jsx

│ │ │ ├── NavBar.jsx

│ │ │ ├── Register.jsx

│ │ │ └── UserHome.jsx

│ │ └── user/

│ │ ├── student/

│ │ │ ├── CourseContent.jsx

│ │ │ ├── EnrolledCourses.jsx

│ │ │ └── StudentHome.jsx

│ │ └── teacher/

│ │ ├── AddCourse.jsx

│ │ └── TeacherHome.jsx

│ ├── App.css

│ ├── App.jsx

│ ├── main.jsx

│ └── index.css

├── package.json

├── vite.config.js

### **5.2 Backend Folder Structure (**/backend**)**

backend/

├── config/ # DB connection

│ └── db.js

├── controllers/ # Handles business logic

│ ├── adminController.js

│ └── userControllers.js

├── middlewares/ # Auth middleware

│ └── authMiddleware.js

├── routers/ # API route handlers

│ ├── adminRoutes.js

│ └── userRoutes.js

├── schemas/ # Mongoose models

│ ├── courseModel.js

│ ├── coursePaymentModel.js

│ ├── enrolledCourseModel.js

│ └── userModel.js

├── uploads/ # For storing uploaded files (e.g., thumbnails)

├── .env # Environment variables

├── .gitignore

├── index.js # Main server entry point

├── package.json

└── package-lock.json

### **Highlights**

* Components are grouped by user roles: **admin**, **student**, and **teacher**.
* Shared UI logic is placed under /common for reusability.
* Backend cleanly separates **routes**, **controllers**, and **models**, improving code readability and scalability.
* Uses **Mongoose** models for MongoDB interactions and **JWT middleware** for route protection.

## 6. Running the Application

To run the **Online Learning Platform** locally, you need to start both the **backend server** and the **frontend development server**. Ensure that all dependencies are installed and MongoDB is running.

### **Step 1: Start MongoDB**

Make sure MongoDB is running on your local machine. You can use:

mongod

or if using MongoDB Compass or Atlas, ensure the connection string in .env is correct.

### **Step 2: Start the Backend Server**

Navigate to the backend/ directory and start the Express server:

cd backend

npm start

* The backend will run at: [**http://localhost:5000**](http://localhost:5000/)
* It connects to MongoDB and exposes API routes like /api/users, /api/courses, etc.

### **Step 3: Start the Frontend (React) Server**

Open a new terminal, then navigate to the frontend/ directory:

cd frontend

npm run dev

* The frontend will run at: [**http://localhost:5172**](http://localhost:5172/)
* It connects to the backend via Axios for data (e.g., login, course fetching).

### **Quick Summary of Commands**

# Terminal 1 - Backend

cd backend

npm install # First-time only

npm start

# Terminal 2 - Frontend

cd frontend

npm install # First-time only

npm run dev

## 7. API Documentation (Short)

The backend exposes REST APIs for **authentication**, **course management**, **user management**, and **enrollments**. All APIs use JSON and are secured with JWT where needed.

### 🔐 Authentication

* POST /api/auth/register – Register a new user
* POST /api/auth/login – Login and receive JWT token

### 👤 Users

* GET /api/users/profile – Get logged-in user profile
* GET /api/users – Get all users (Admin only)

### 📚 Courses

* POST /api/courses – Create new course (Teacher only)
* GET /api/courses – List all courses
* GET /api/courses/:id – Get course by ID
* DELETE /api/courses/:id – Delete course (Teacher/Admin)

### ✅ Enrollments

* POST /api/enroll/:courseId – Enroll in a course (Student)
* GET /api/enrollments/my-courses – View enrolled courses

## 8. Authentication

The platform uses **JWT (JSON Web Token)** based authentication to securely manage user sessions and enforce **role-based access** for Admin, Teacher, and Student functionalities.

### 🔐 How It Works

1. **User Registration/Login**
   * On successful login or registration, the server generates a **JWT token**.
   * The token is sent to the frontend and stored in **localStorage**.
2. **Protected Routes**
   * Certain backend routes (e.g., course creation, enrollments, dashboard) require a valid JWT token.
   * A custom **auth middleware** (authMiddleware.js) checks the token before granting access.
3. **Token Structure**
   * Encodes user ID and role (admin, teacher, or student).
   * Signed with a secret key from .env file.

### 🧠 Role-Based Access Control

| **Role** | **Access Permissions** |
| --- | --- |
| **Admin** | View all users, delete courses, platform control |
| **Teacher** | Create courses, view enrolled students |
| **Student** | Enroll in courses, track progress, get certificates |

### 🛡️ Example Token Usage

**Frontend Request Header:**

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

✅ This ensures only authenticated users can access secure resources and prevents unauthorized access to admin or teacher functionality.

## 9. User Interface

The Online Learning Platform provides a simple and responsive user interface developed using React.js, Bootstrap, Material UI, and MDB React UI Kit. The UI is designed for ease of use across different user roles: Student, Teacher, and Admin.

### Key Features

* Role-based dashboards with custom views for students, teachers, and admins.
* Responsive design compatible with desktops, tablets, and mobile devices.
* Clean layout using modern UI libraries.
* Navigation bar with dynamic links based on user login and role.

### Interfaces by Role

**Student Dashboard**

* Browse and enroll in courses
* View enrolled courses
* Track progress and download certificates

**Teacher Dashboard**

* Create and manage courses
* Add course content and sections
* View student enrollments

**Admin Dashboard**

* Manage all users and courses
* Monitor platform activity

## 10. Testing

Testing was conducted to ensure the core functionalities of the platform work as expected for all user roles: Admin, Teacher, and Student.

### Testing Methods

**1. Manual Testing**

* Tested all user flows including registration, login, course creation, enrollment, and certificate generation.
* Verified role-based access to dashboards and protected routes.

**2. API Testing**

* Used Postman to test backend API endpoints for authentication, course management, and enrollments.
* Checked status codes, response structures, and error handling.

**3. UI Testing**

* Verified responsive behavior on different screen sizes.
* Checked component rendering, form validations, and navigation across the app.

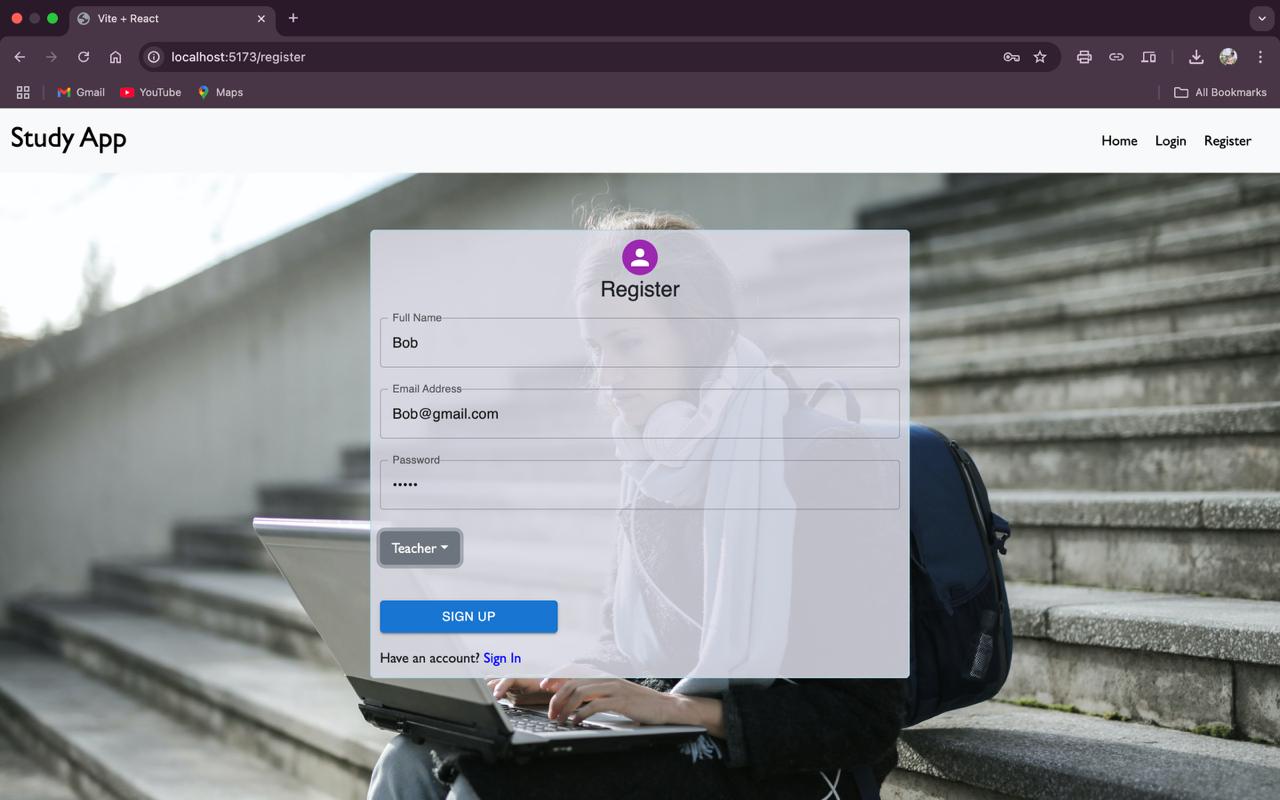
### Tools Used

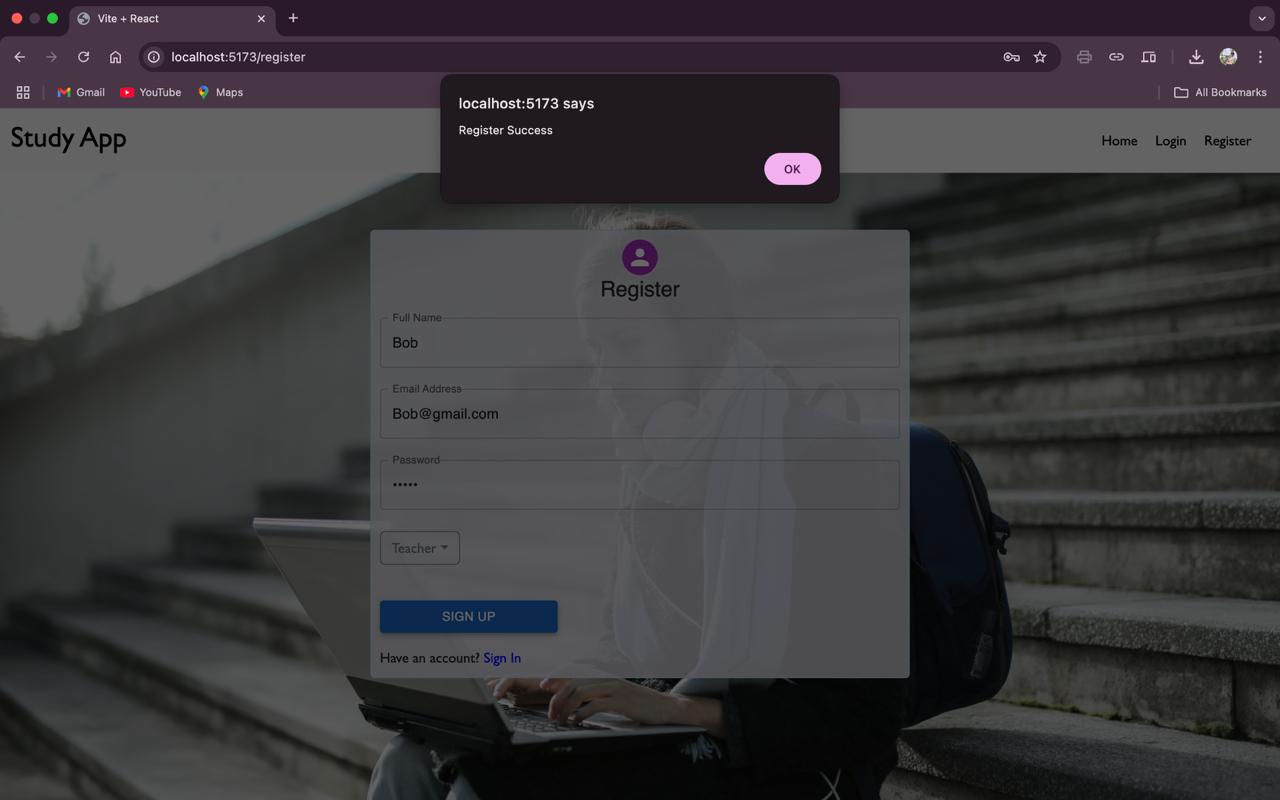
* Postman (API testing)
* Browser Developer Tools (UI/console testing)
* React DevTools (component testing)

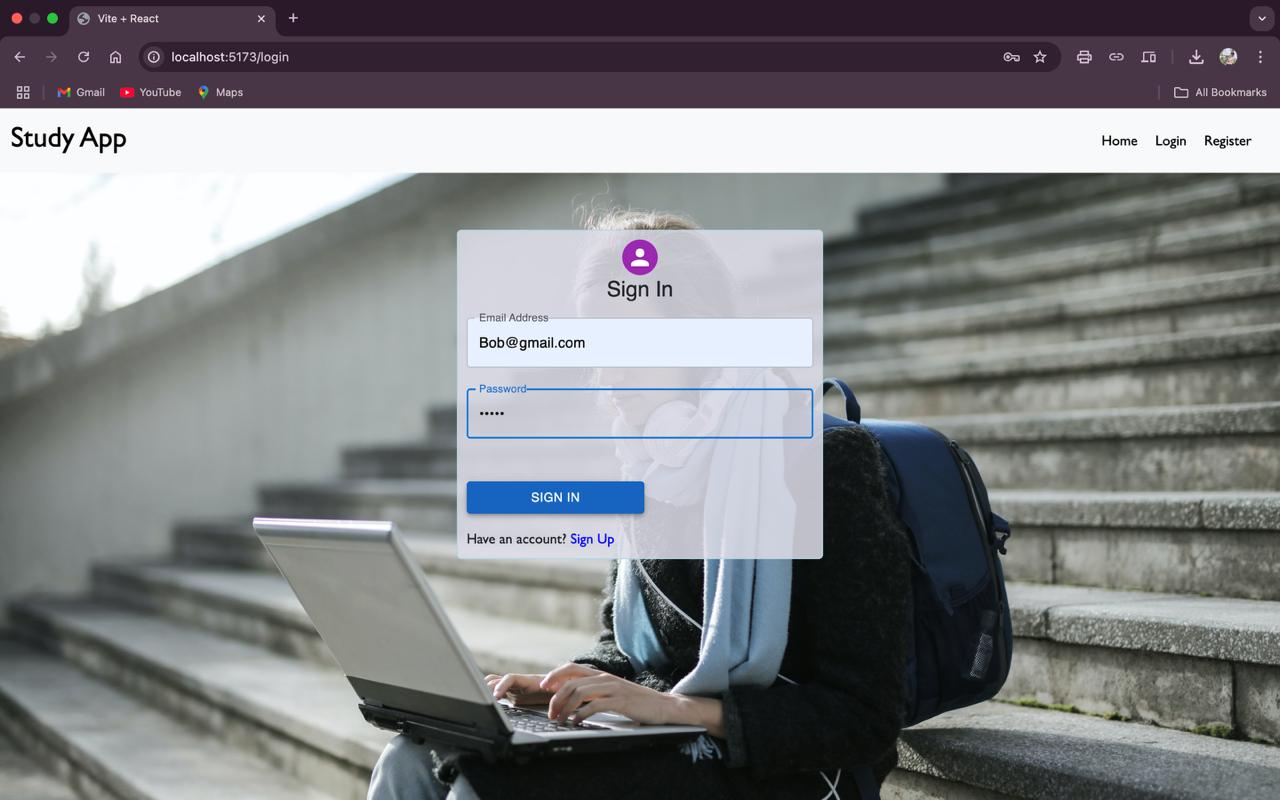
## 11. Screenshots or Demo

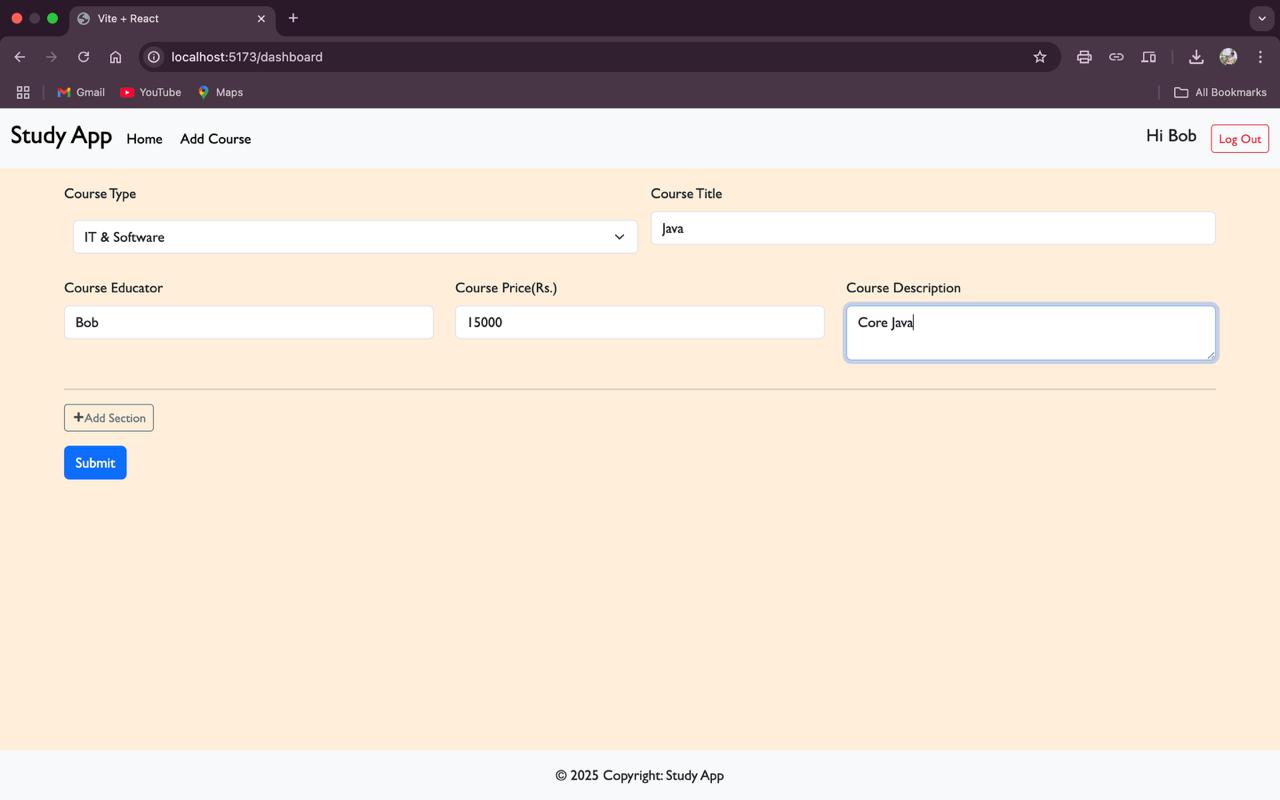
This section includes visual evidence of the platform’s functionality and user interface.

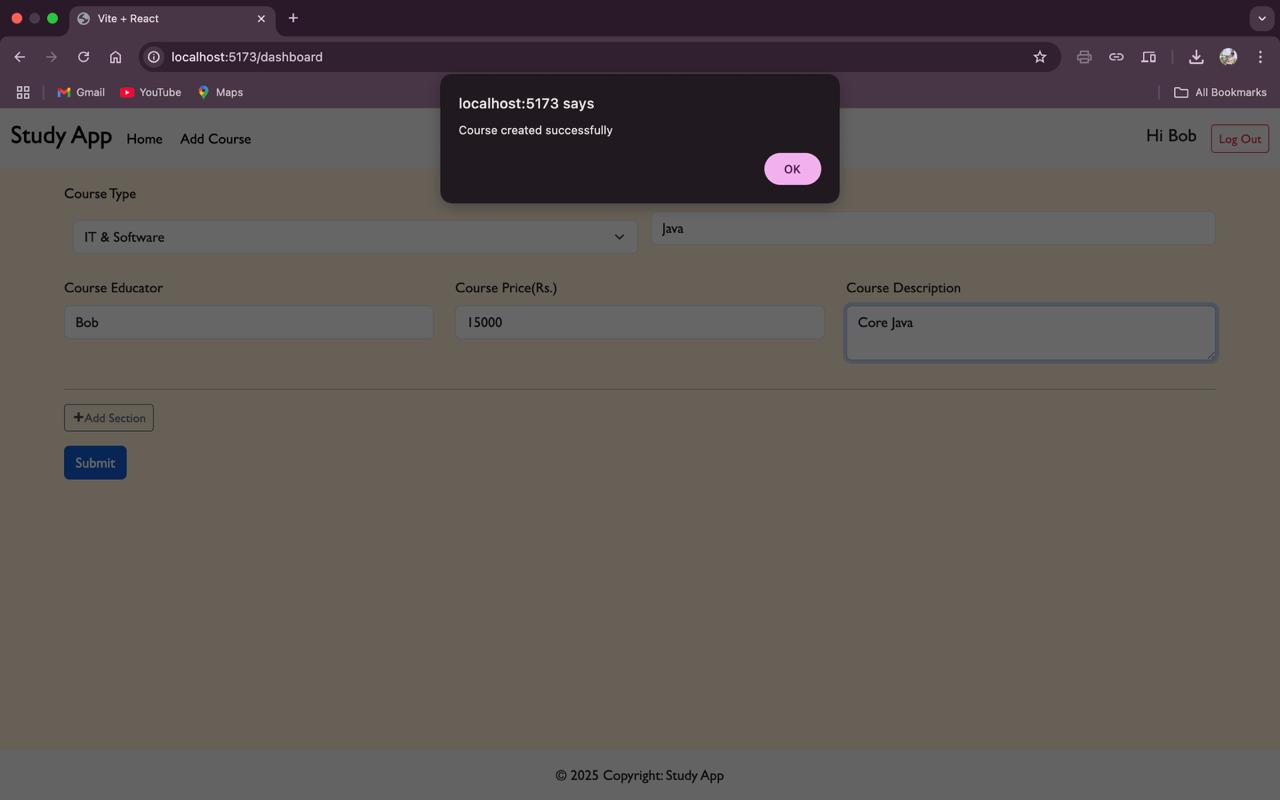
Screenshots:-

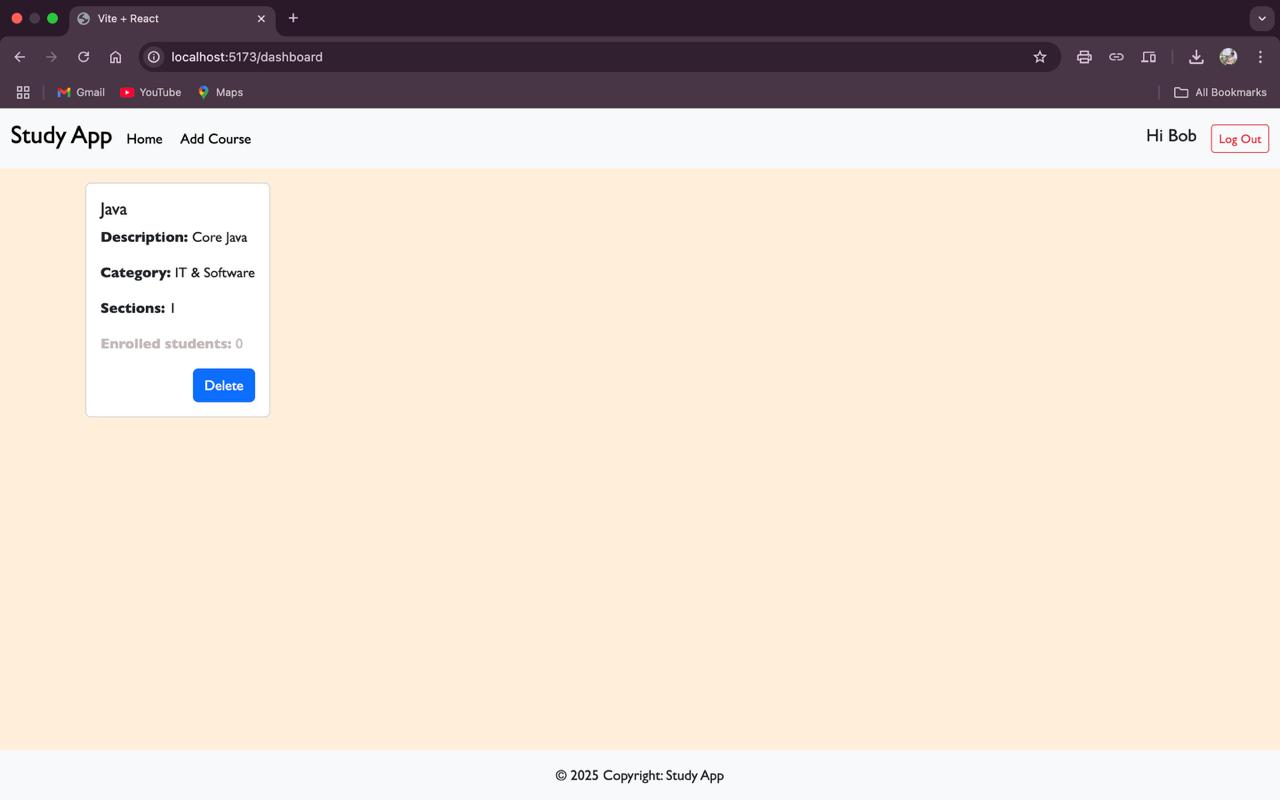




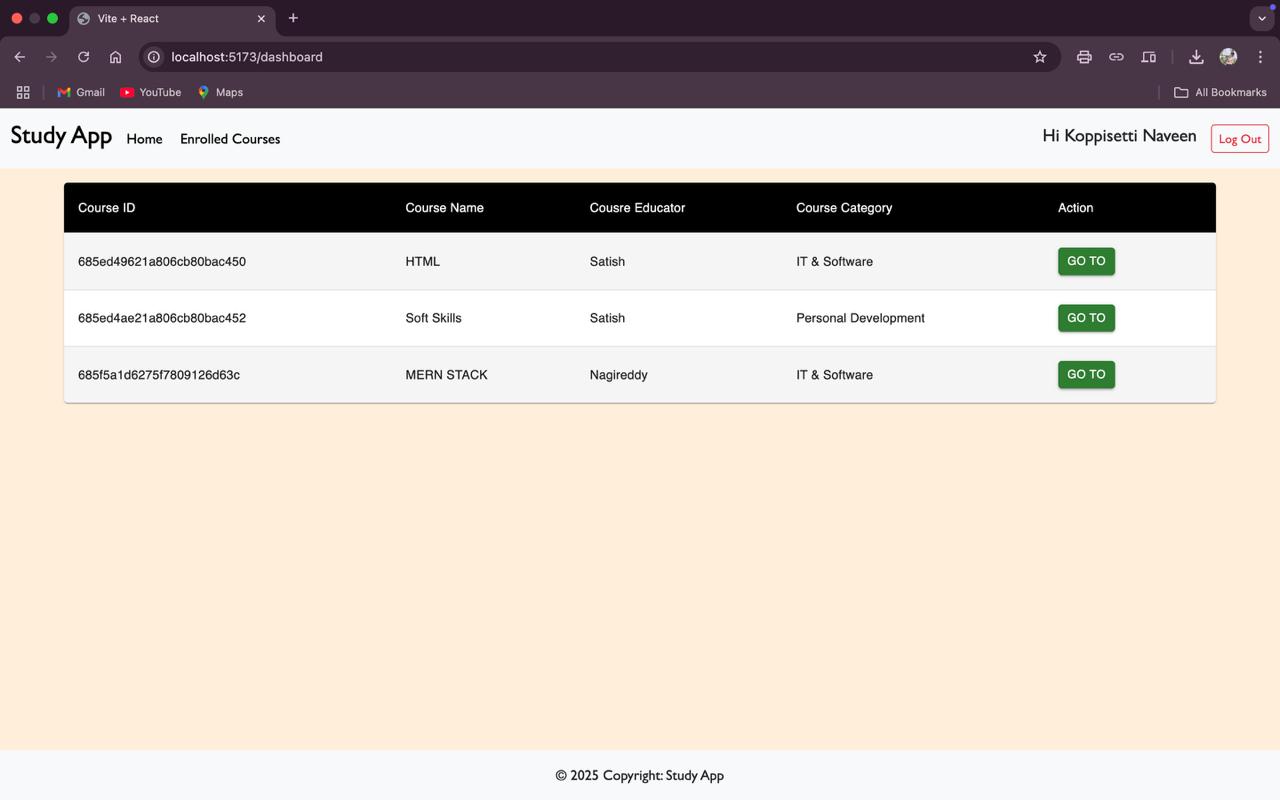




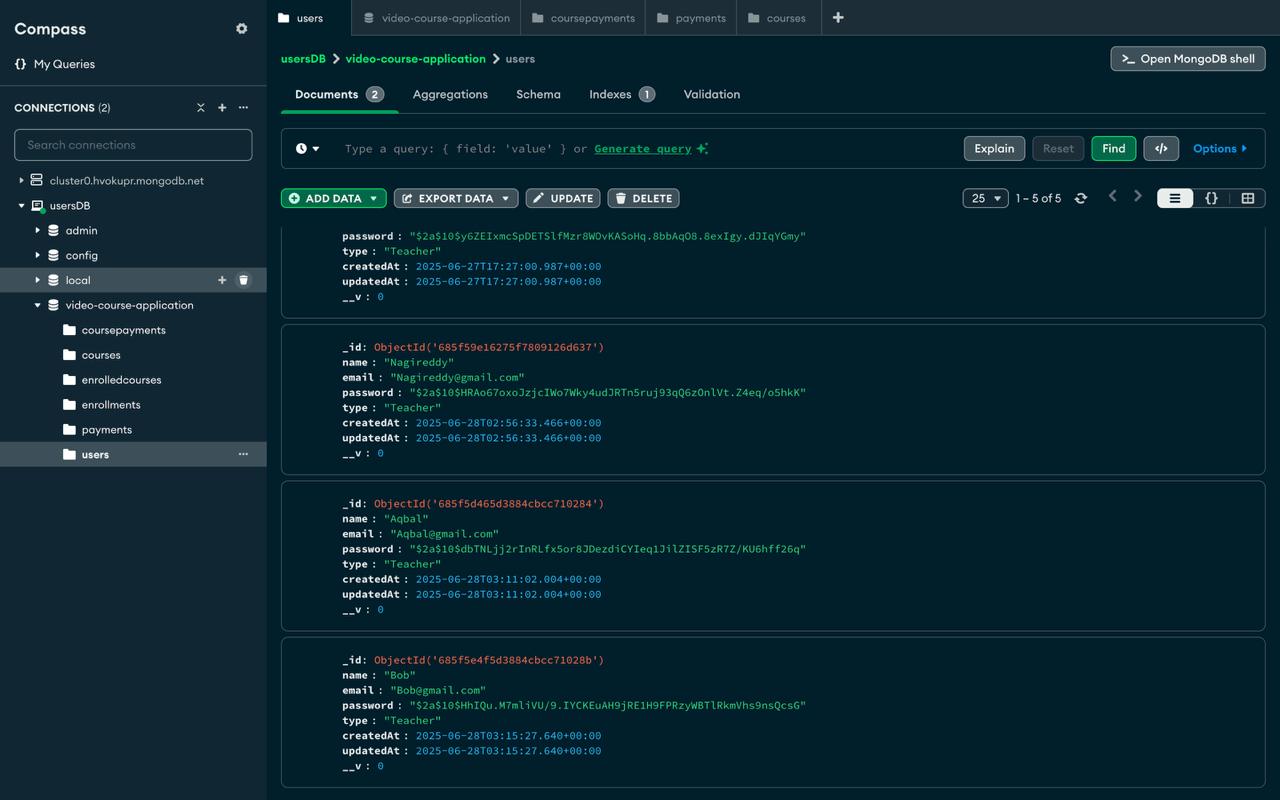




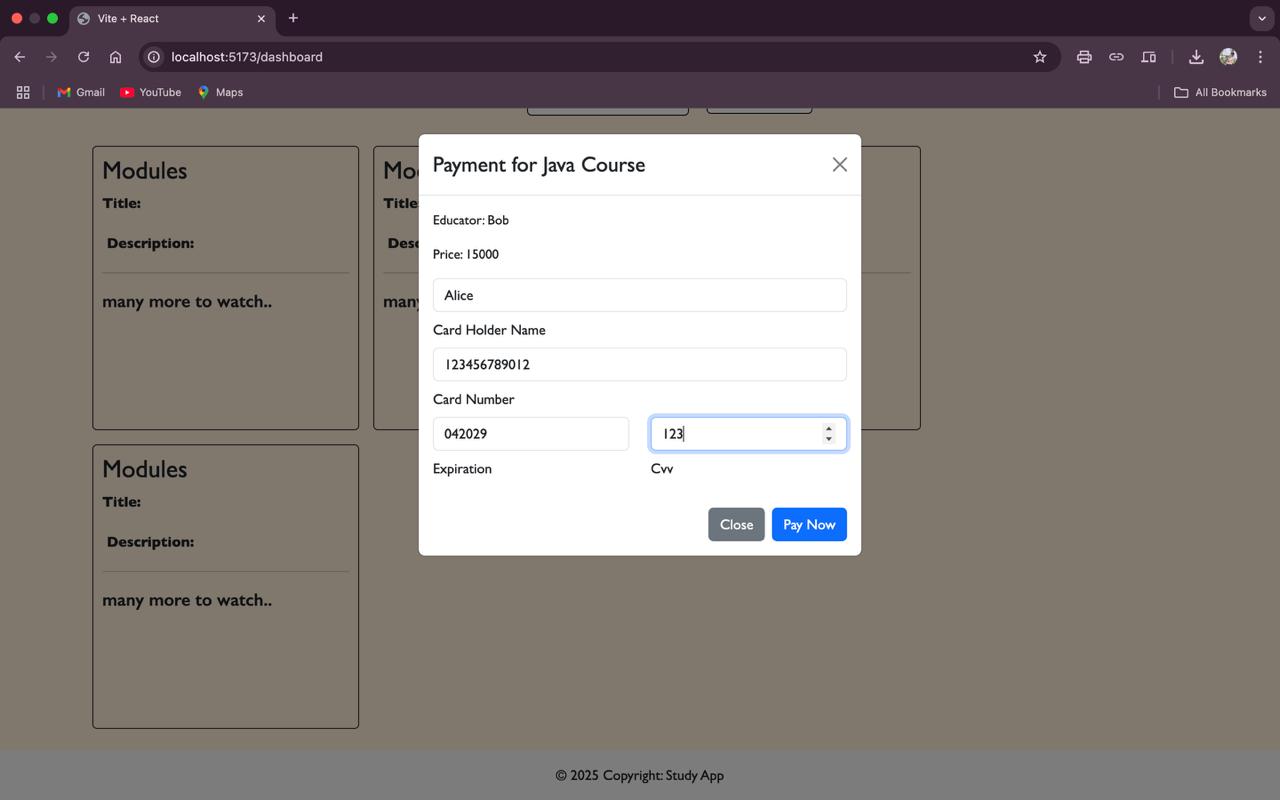
### C:\Users\AQBALL\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\TempState\A69D15DCF6D76D9D056662A38A11392D\WhatsApp Image 2025-06-28 at 08.47.49_9d8679b0.jpg



### C:\Users\AQBALL\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\TempState\F749025817F607FFF3160D1E5244A9F6\WhatsApp Image 2025-06-28 at 08.48.11_4f3bb2cd.jpg



### C:\Users\AQBALL\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\TempState\EAD97089AAE476D362A942D978947C32\WhatsApp Image 2025-06-28 at 08.49.36_8d318f52.jpg



### Demo Video

* [demo video](https://drive.google.com/file/d/1o1sah3qYJ8ZtkxhXYN8qpILiH2cIpkYV/view?usp=sharing)

## 12. Known Issues

The following issues were identified during development and testing:

* No password reset or "Forgot Password" functionality implemented
* Course media (e.g., videos, PDFs) are accessed statically; streaming is not supported
* Form validation is basic and can be improved for better user input handling
* User registration does not include email verification
* Payment integration is limited or not yet implemented for paid courses
* No real-time communication features such as chat or notifications

## 13. Future Enhancements

The following improvements and features are planned for future versions of the Online Learning Platform:

* Implement password reset via email with secure token validation
* Integrate payment gateways (e.g., Razorpay, Stripe) for premium course access
* Add live class support using video conferencing APIs like Zoom or Jitsi
* Enable real-time chat between students and instructors
* Introduce quizzes and assignments for course assessments
* Add progress tracking with visual indicators (e.g., progress bars)
* Implement email notifications and reminders for course activities
* Create a mobile-friendly version or a mobile app using React Native
* Improve admin analytics with charts and activity logs
* Add multilingual support for a wider audience reach