ONLINE LEARNING PLATFORM USING MERN

1.INTRODUCTION

An Online Learning Platform (OLP) is a web-based system that offers tools and resources to support flexible and accessible education. It enables users of all backgrounds to learn at their own pace from any device.

Project Title : Online Learning Platform Using MERN.

Team Members:

1. Devarakonda Poojanjali

2. Dhanyuni Harshitha

3. Killi Anusha

4. Nowdu Kanaka Mahalakshmi

5. Palivela Sesha Ratnam

2.PROJECT OVERVIEW

Purpose

The Online Learning Platform (OLP) is a web-based application designed to facilitate education over the internet. It allows learners to explore, enroll, and learn from a variety of online courses and provides tools for instructors to manage content.

Features

1.User registration/login for students, teachers, and admins

2.Course browsing and filtering

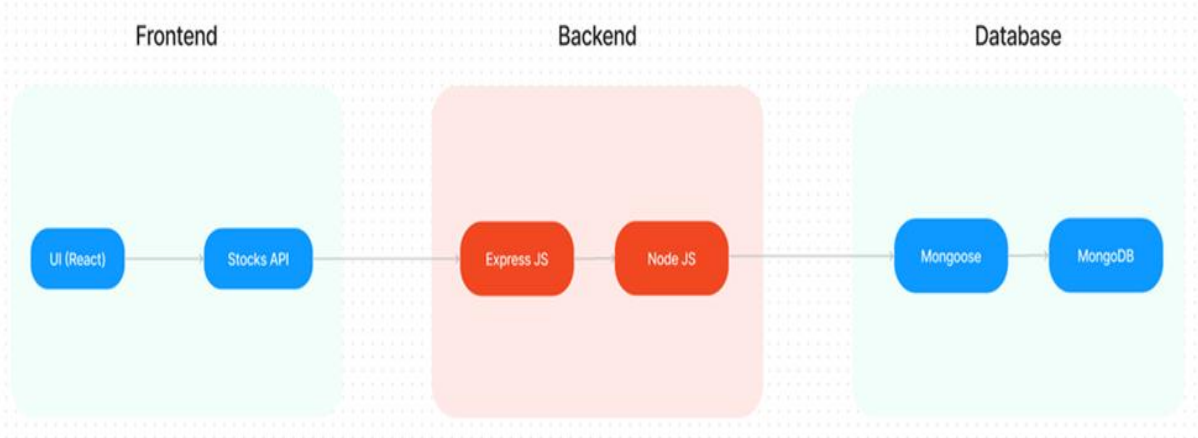
3.Enrolling in free and paid courses

4.Progress tracking and certification

5.Admin control for course/user management

6.Payment integration for premium content

3.TECHINICAL ARCHITECTURE



The technical architecture of OLP app follows a client-server model, where the frontend serves as the client and the backend acts as the server. The frontend encompasses not only the user interface and presentation but also incorporates the axios library to connect with backend easily by using RESTful Apis.

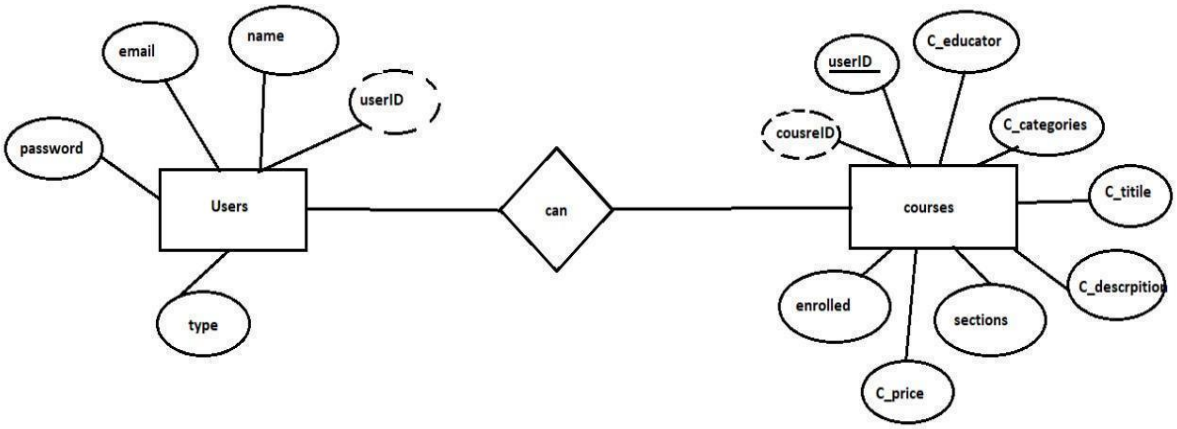
The front end utilizes the bootstrap and material UI library to establish a real-time and better UI experience for any user.

On the backend side, we employ Express.js frameworks to handle the server-side logic and communication.

For data storage and retrieval, our backend relies on MongoDB. MongoDB allows for efficient and scalable storage of user data and necessary information about the place.

Together, the frontend and backend components, along with Express.js, and MongoDB, form a comprehensive technical architecture for our OLP app. This architecture enables real-time communication, efficient data exchange, and seamless integration, ensuring a smooth and immersive blogging experience for all users.

4.ER DIAGRAM



Here there are 2 collections namely users, courses that have their own fields in

Users:

1. _id: (MongoDB creates by unique default)
2. name
3. email
4. password
5. type

Courses:

1. userID: (can act as a foreign key)
2. _id: (MongoDB creates by unique default)
3. C_educator
4. C_categories
5. C_title
6. C_description
7. sections
8. C_price
9. Enrolled

5.SETUP INSTRUCTIONS:

Perequisites

Here are the key prerequisites for developing a full-stack application using Node.js, Express.js, MongoDB, and React.js:

✓Vite:

Vite is a new frontend build tool that aims to improve the developer experience for development with the local machine, and for the build of optimized assets for production (go live). Vite (or ViteJS) includes a development server with ES _native_ support and Hot Module Replacement; a build command based on rollup.

npm create vite@latest

✓Node.js and npm:

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server side. It provides a scalable and efficient platform for building network applications.

Install Node.js and npm on your development machine, as they are required to run JavaScript on the server side.

Download: https://nodejs.org/en/download/

Installation instructions: https://nodejs.org/en/download/package-manager/

npm init

✓Express.js:

Express.js is a fast and minimalist web application framework for Node.js. It simplifies the process of creating robust APIs and web applications, offering features like routing, middleware support, and modular architecture.

Install Express.js, a web application framework for Node.js, which handles server-side routing, middleware, and API development.

Installation: Open your command prompt or terminal and run the following command:

npm install express

✓MongoDB:

MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-like format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data.

Set up a MongoDB database to store your application's data.

Download: https://www.mongodb.com/try/download/community

Installation instructions: https://docs.mongodb.com/manual/installation/

✓React.js:

React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

Install React.js, a JavaScript library for building user interfaces.

Follow the installation guide: https://reactjs.org/docs/create-a-new-react-app.html

✓**HTML, CSS, and JavaScript**: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

✓**Database Connectivity**: Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations. To Connect the Database with Node JS go through the below provided link:

https://www.section.io/engineering-education/nodejs- mongoosejs-mongodb/

Install Dependencies:

• Navigate into the cloned repository directory:

cd containment-zone

• Install the required dependencies by running the following commands:

cd frontend

npm install

cd ../backend

npm install

Start the Development Server:

• To start the development server, execute the following command:

npm start

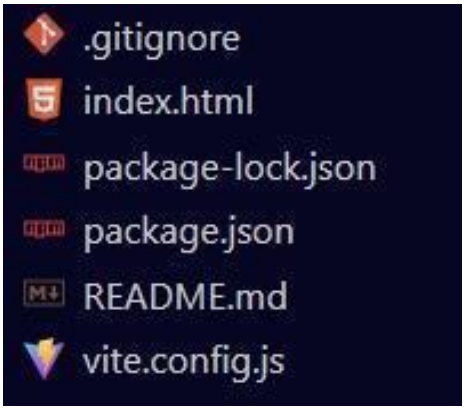
• The OLP app will be accessible at http://localhost:5172

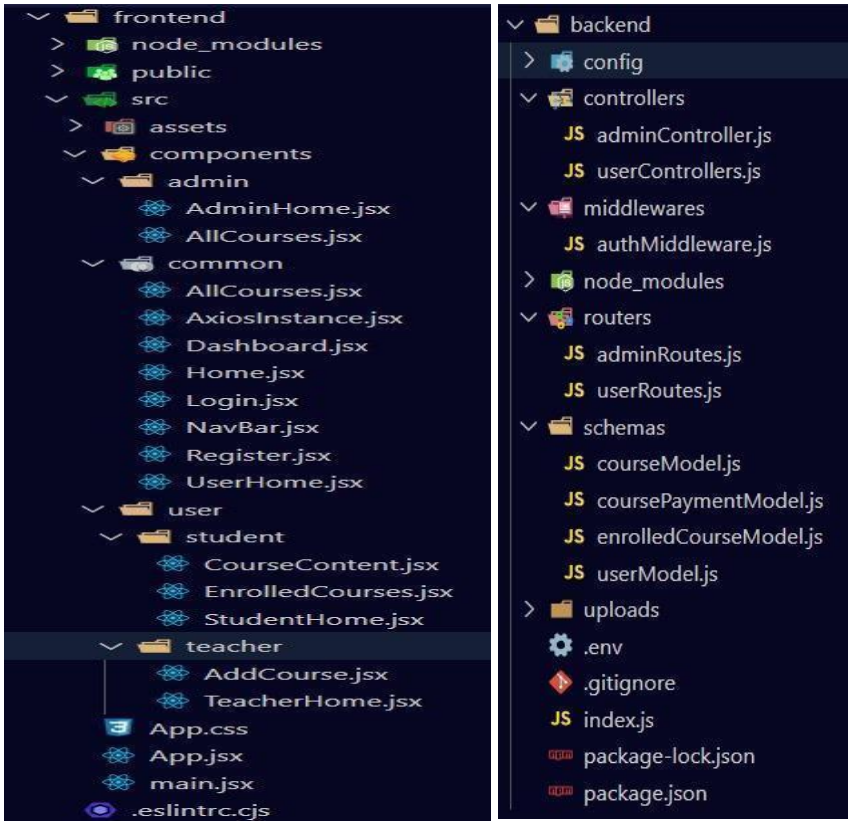
You have successfully installed and set up the Online learning app on your local machine. You can now proceed with further customization, development, and testing as needed.

6.FOLDER STRUCTURE

The first image is of the front part which shows all the files and folders that have been used in UI development

The second image is of the Backend part which shows all the files and folders that have been used in the backend development





7.RUNNING THE APPLICATION

To run the Online Learning Platform (OLP) on your local machine, you need to start both the frontend (React) and backend (Node.js/Express) servers. Below are the detailed steps to do that:

Step 1: Prerequisites (Make sure these are installed)

- Node.js and npm:
  Download and install from: https://nodejs.org/en/download
  Confirm installation using:

  node -v

  npm -v

- MongoDB:
  Download and install MongoDB Community Edition:
  https://www.mongodb.com/try/download/community
  Start MongoDB server locally (usually starts at port 27017).

- Vite (for frontend build):
  Vite is already included in the frontend dependencies.

Step 2: Clone the Project Repository

If the project is hosted on GitHub:

git clone https://github.com/DhanyuniHarshitha/LearnHub-MERN

cd online-learning-platform

Step 3: Setup the Backend (Server)

1.Navigate to the backend folder:

cd backend

2.Install backend dependencies:

npm install

3.Create a .env file in the backend directory and add the following environment variables

PORT=5000

MONGO_URI=mongodb://localhost:27017/olp_db

JWT_SECRET=your_jwt_secret_key

4.Start the backend server:

npm start

- The backend will run at: http://localhost:5000
- You should see a message like Server is running on port 5000

Step 4: Setup the Frontend (Client)

1. Open a new terminal window.
2. Navigate to the frontend folder:

   cd frontend

3. Install frontend dependencies:

   npm install

4. Start the frontend server using Vite:

   npm run dev

   - The frontend will run at: http://localhost:5172 (Vite default)
   - It should automatically open in your browser. If not, open your browser and go to:http://localhost:5172

Step 5: Connect Frontend with Backend

- Ensure that API requests from the React frontend (e.g., using Axios) are pointing to http://localhost:5000/api.
- You may need a proxy configuration in vite.config.js or use full URLs in API calls to link frontend and backend.

Step 6: Using the App

- Register/Login:
  Register as a student or teacher, or log in as admin.
- Explore Courses:
  View available courses, enroll, and track progress.
- Create/Edit Courses:
  Teachers can create or manage courses.
- Admin Panel:
  Admin can manage users, view enrollments, and control platform settings.

8.API DOCUMENTATION

Endpoint	Method	Description	Access	Request Body / Params	Response (Sample)
/api/auth/register	POST	Register a new user	Public	{ name, email, password, type }	{ token, user }
/api/auth/login	POST	Authenticate user and return token	Public	{ email, password }	{ token, user }
/api/users/profile	GET	Get logged-in user's profile	Authenticated	Header: Authorization: Bearer <token>	{ _id, name, email, type }
/api/users/update	PUT	Update user profile	Authenticated	{ name, password (optional) }	{ message: "Profile updated" }
/api/courses	GET	Get all courses with optional filters	Public	Query: ?name=xyz&category=abc	[{ course }, ...]
/api/courses/:id	GET	Get specific course by ID	Public	URL Param: id	{ course }
/api/courses	POST	Create a new course	Teacher	{ title, description, category, price, sections }	{ message: "Course created" }
/api/courses/:id	PUT	Update a course	Teacher	{ title, description, price, sections (optional) }	{ message: "Course updated" }

Endpoint	Method	Description	Access	Request Body / Params	Response (Sample)
/api/courses/:id	DELETE	Delete a course	Teacher	URL Param: id	{ message: "Course deleted" }
/api/courses/:id/enroll	POST	Enroll in a course	Student	URL Param: id	{ message: "Enrolled successfully" }
/api/courses/:id/sections	POST	Add section to a course	Teacher	{ title, content }	{ message: "Section added" }
/api/enrolled	GET	List enrolled courses for a user	Student	Header: Authorization: Bearer <token>	[{ course }, ...]
/api/enrolled/:id/progress	GET	Get progress in a specific course	Student	URL Param: id (Course ID)	{ progressPercentage }
/api/certificate/:id	GET	Download course certificate	Student	URL Param: id	{ certificateLink }
/api/admin/users	GET	List all users	Admin	Header: Authorization: Bearer <token>	[{ user }, ...]
/api/admin/courses	GET	List all courses (Admin view)	Admin	Header: Authorization: Bearer <token>	[{ course }, ...]

Endpoint	Method	Description	Access	Request Body / Params	Response (Sample)
/api/admin/course/:id/delete	DELETE	Force delete a course	Admin	URL Param: id	{ message: "Course force-deleted" }
/api/admin/overview	GET	Admin dashboard data summary	Admin	None	{ totalUsers, totalCourses, totalEnrollments }

9.AUTHENTICATION

It's the process of logging in and proving who you are — like using your email and password to access the platform.

How It Works in This Project:

1.Register (Sign Up)

- You enter your name, email, password, and role (student/teacher).
- Your password is encrypted (using bcryptjs) and saved in the database.
- A login token is generated (called a JWT) and sent to you.

2.Login

- You enter your email and password.
- The backend checks if:
  - The email exists
  - The password matches the saved (encrypted) one
- If both match, a new JWT token is created and sent to you.

3.Using the Token

- This token is like a pass that proves you're logged in.
- It must be sent with every request to access private parts of the site (like dashboards or course enrollment).

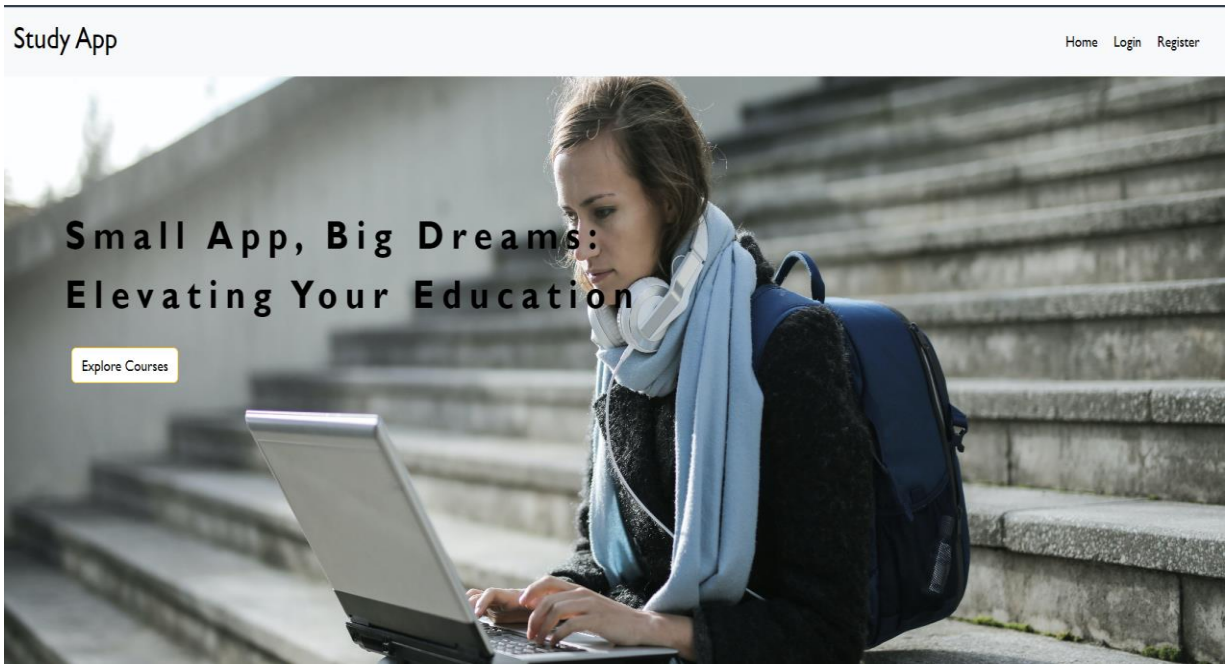
4.Middleware

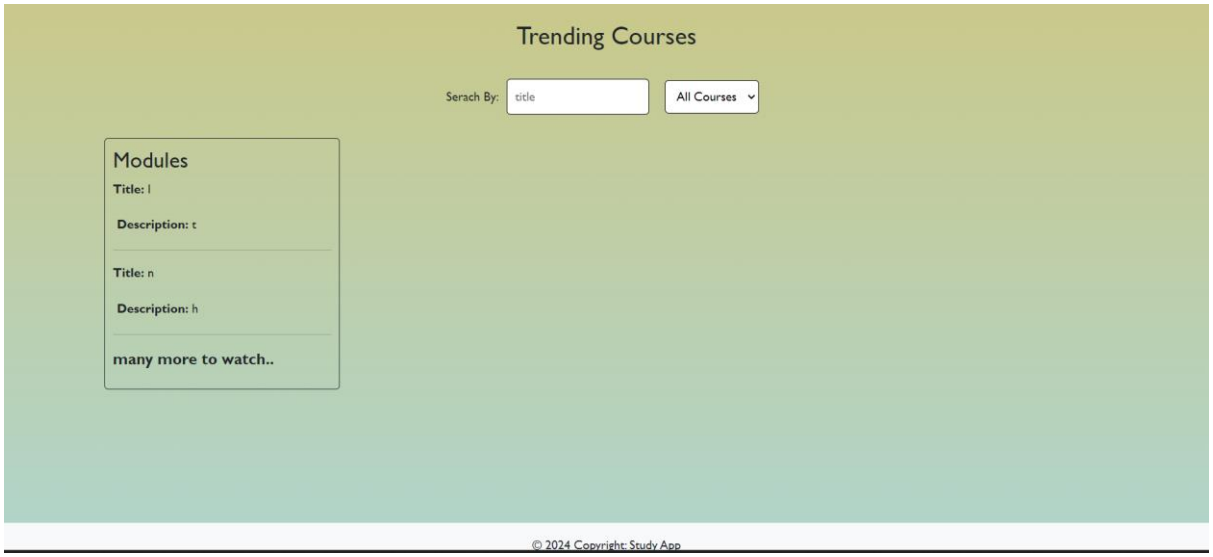
- A special function checks the token before allowing access to protected routes.
- If the token is valid, you're allowed in.
- If not, it says "Access Denied".

10.USER INTERFACE

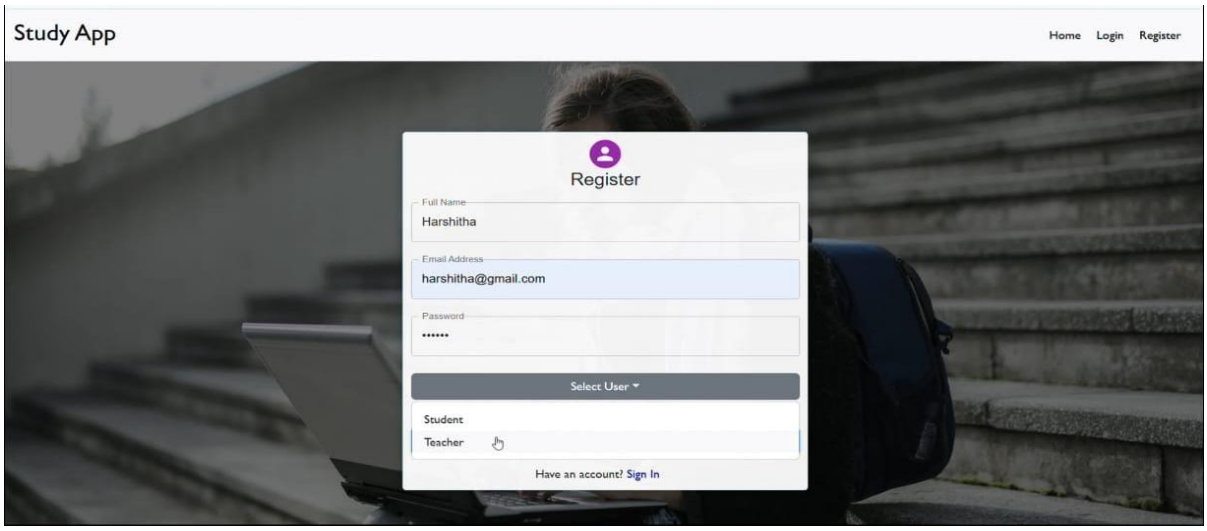
The user interface of the Online Learning Platform is designed to be clean, modern, and user-friendly, making it easy for students, teachers, and admins to interact with the system

1.Landing Page: Welcomes user and shows featured courses or platform highlights.

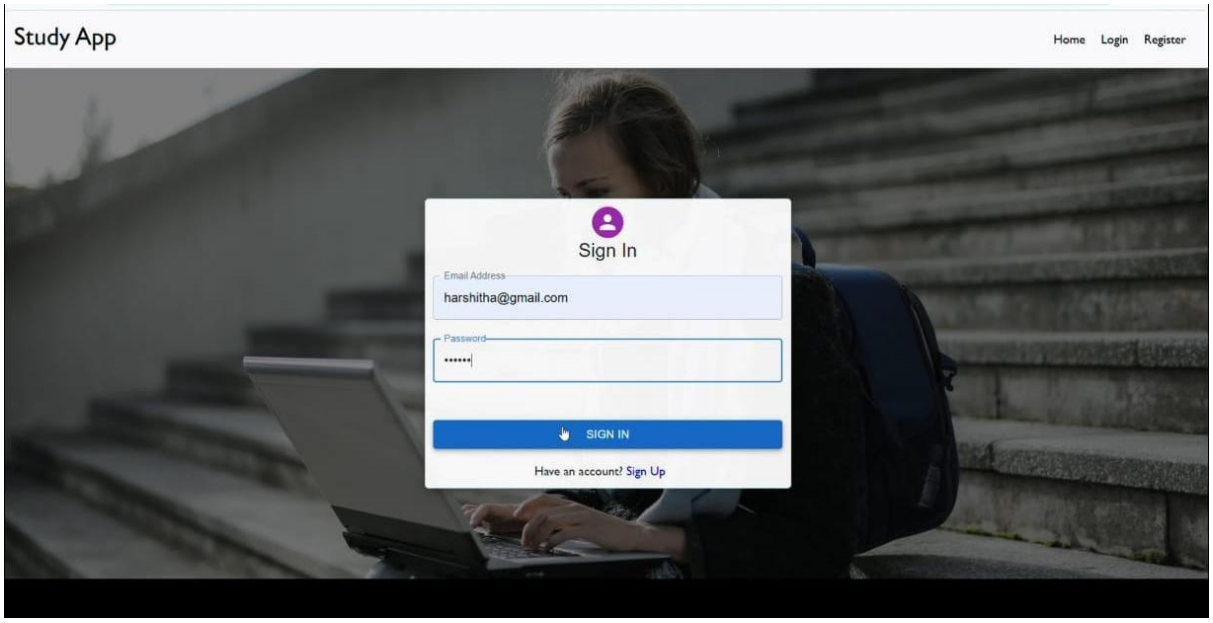




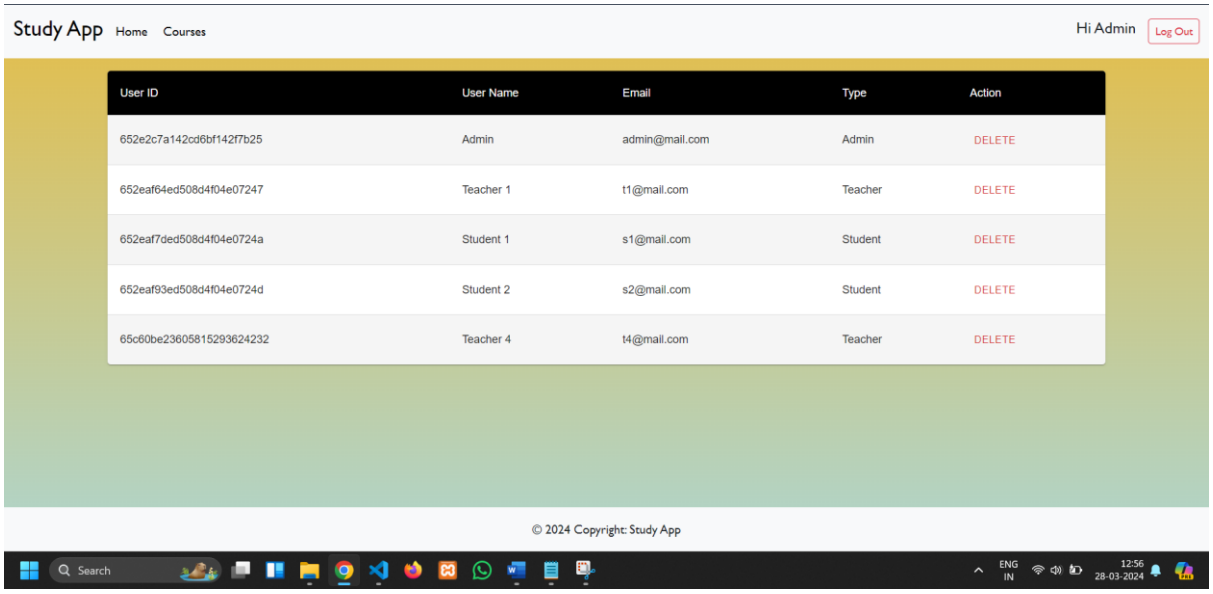
2.**Register Page**: Allows new users to sign up as a Student or Teacher.



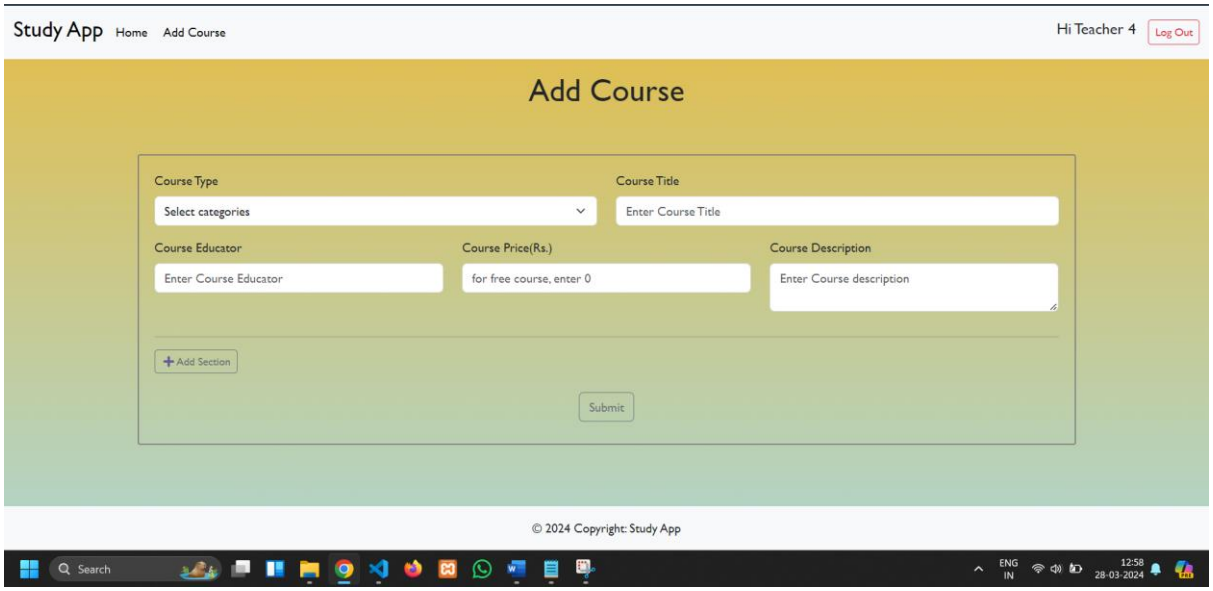
3.**Login Page**: Simple login form with email and password fields.



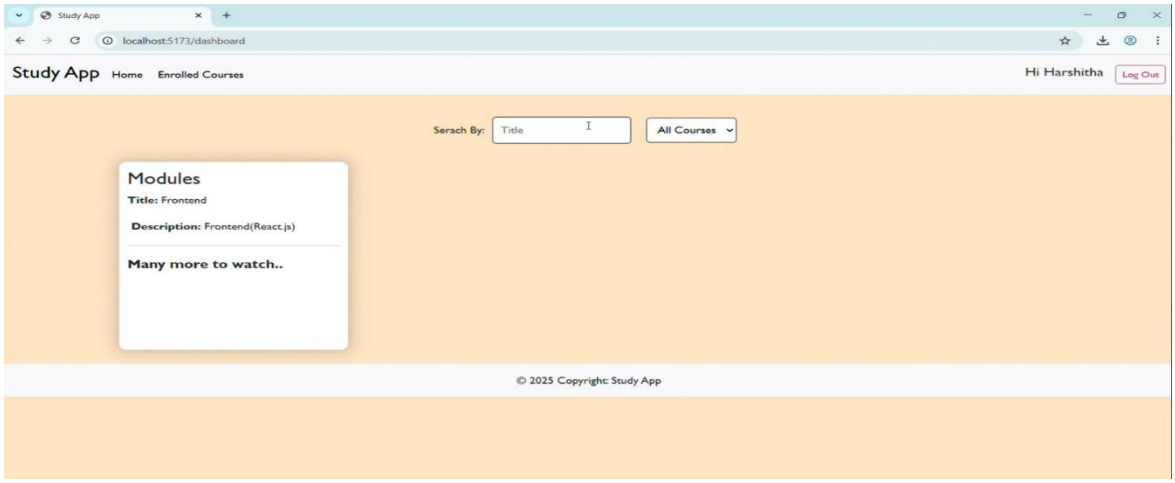
4.**Admin Dashboard:** Full control panel for managing users, courses, and monitoring activity.



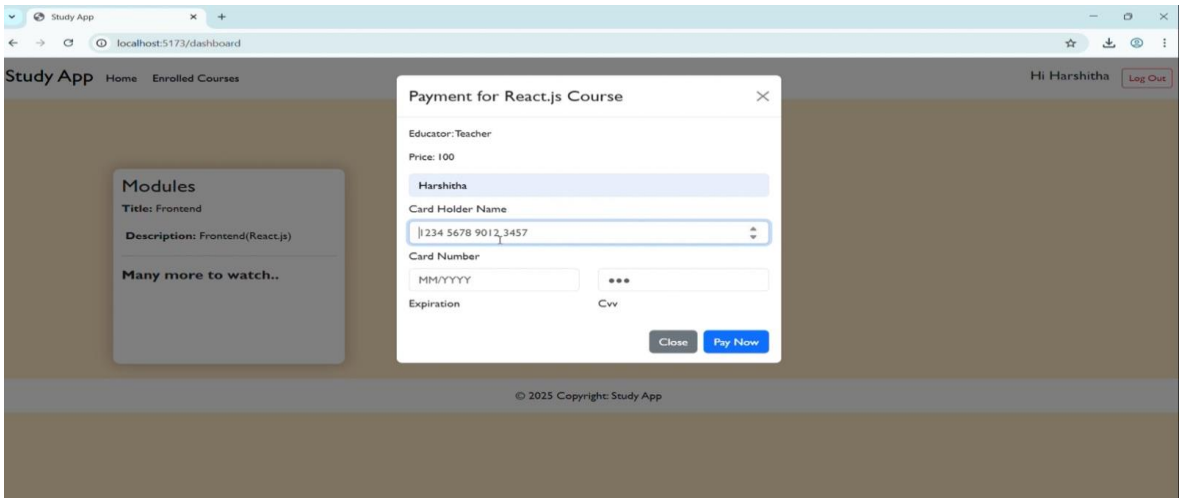
5.**Teacher Dashboard**: Lets teachers create, edit, and manage courses and sections



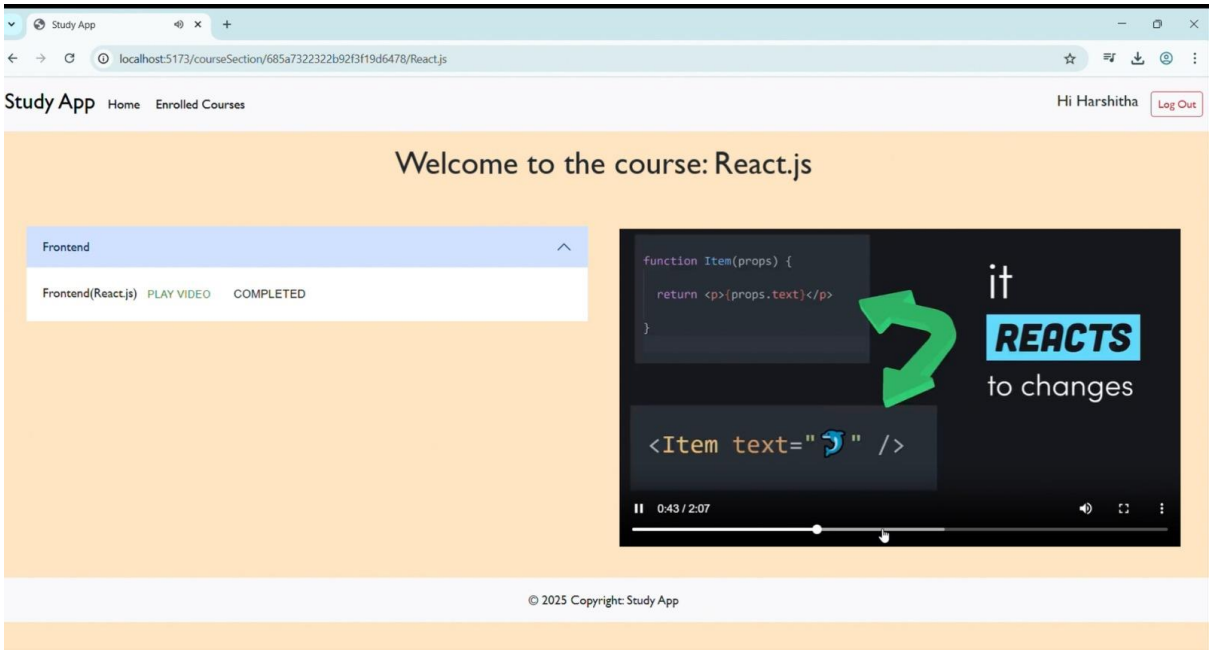
6.**Course Dashboard**: Lists enrolled courses, course progress, and certificates.



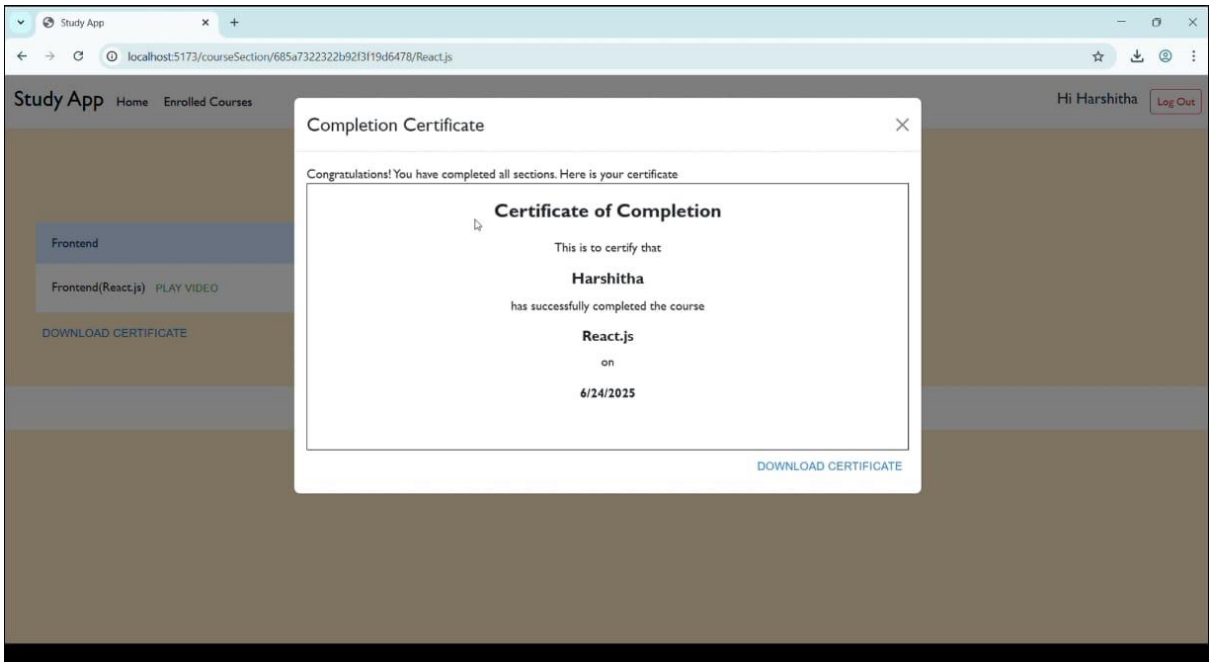
7.Payment For The Course:



8.Course Content:



9.Course Completion Certificate:



11.TESTING

Testing ensures that all features of the Online Learning Platform work correctly and provide a smooth experience for users.

Types of Testing Done:

1. Manual Testing
    - Each feature (registration, login, course creation, enrollment, etc.) was tested manually by developers.
    - Different user roles (Student, Teacher, Admin) were tested to verify correct access.
2. Functional Testing
    - Verified that:
        - Students can enroll and track progress.
        - Teachers can create and edit courses.
        - Admins can manage users and courses.
3. UI Testing
    - Checked that pages look good and work well on different screen sizes (mobile, tablet, desktop).

Tools Used:

- Postman – To test and verify API responses.
- Browser Dev Tools – For debugging and inspecting frontend behavior.

12. GITHUB AND DEMO

Github Link: https://github.com/DhanyuniHarshitha/LearnHub-MERN

Demo Link: https://drive.google.com/file/d/158a0xjYu5d2HZ7UQmuTCgeGZ_dzVR2rq/view?usp=drive_link

13. KNOWN ISSUES

While the platform is fully functional and tested, there are a few known limitations and areas for improvement:

1. Payment Gateway Not Integrated
    - Currently, paid course functionality is mocked and not connected to a real payment gateway like Razorpay or Stripe.
2. No Password Reset Option
    - Users cannot reset their password if forgotten. A “Forgot Password” feature needs to be added.
3. Limited Course Search/Filter Options
    - Course listing lacks advanced filtering (e.g., by rating, difficulty, or instructor name).
4. No Video Streaming Optimization
    - Course videos are displayed but not streamed via CDN or adaptive quality methods, which may affect performance on slow networks.
5. Basic Admin Dashboard
    - Admin panel provides only core features and lacks advanced analytics or filtering tools.
6. Minimal Mobile Optimization
    - While the UI is responsive, some components may not be fully optimized for small-screen devices.

14.FUTURE ENHANCEMENT

To improve the platform and provide a better user experience, the following features are planned for future development:

1. Payment Gateway Integration
    - Integrate real payment options using Razorpay or Stripe for secure paid course transactions.
2. Password Recovery Feature

- Add “Forgot Password” functionality with email-based reset options.

3. Course Ratings and Reviews
    - Allow students to rate and review courses to help others choose quality content.
4. Live Class Support
    - Enable teachers to schedule and conduct live classes using Zoom or WebRTC integration.
5. Advanced Admin Analytics
    - Add charts and statistics for admins to monitor course activity, user growth, and revenue.
6. Mobile App Development
    - Build Android/iOS apps to make learning more accessible on mobile devices.
7. Search and Filter Improvements
    - Add smart filters for courses by category, difficulty, instructor, and popularity.
8. AI-Based Course Recommendations
    - Suggest courses to users based on their activity, interests, and skill level.