

# **NAAN MUDHALVAN PROJECT**

## **MERN stack powered by MongoDB**



### **PROJECT TITLE**

## **Online Learning Platform using MERN Stack**

**Submitted by the team members of Final Year CSE 'A'**

**ACHYUTH SUDHARSHAN S      311421104002**

**ALINA MUSKHAN                311421104003**

**AZHAGAN B                      311421104008**

**MOHAMMED HASAN MUSTAFA R    311421104048**



**DEPARTMENT OF COMPUTER SCIENCE AND  
ENGINEERING**

**MEENAKSHI COLLEGE OF ENGINEERING**

**12, VEMBULIAMMAN KOVIL STREET, WEST K.K. NAGAR,**

**CHENNAI - 600 078, TAMILNADU.**

**(AFFILIATED TO ANNA UNIVERSITY)**

# **Online Learning Platform Using MERN Stack**

## **ABSTRACT :**

The **Online Learning Platform (OLP)** is a comprehensive educational solution leveraging the MERN stack (MongoDB, Express, React, Node.js) to provide accessible, flexible, and user-friendly learning opportunities for students worldwide. OLP incorporates a suite of features designed to streamline online education, including course management, interactive learning tools, and flexible accessibility across devices.

Through a client-server architecture, OLP enables real-time data exchange and supports robust functionalities such as user authentication, role-based access, and payment processing for premium content. By offering a seamless and engaging user experience, this platform empowers students to learn at their own pace, interact with peers and instructors, and receive certifications upon course completion.

The platform also supports instructors in creating and managing content and enables administrators to oversee the system effectively. This documentation details the requirements, architecture, features and implementation steps necessary to develop the OLP using MERN, serving as a foundational guide for an effective online education ecosystem.

**Keywords:** Online Education, Web Application, MERN Stack, Vite

# TABLE OF CONTENTS

ABSTRACT	
.....	
2	
<b>1. INTRODUCTION</b>	
.....	<b>4</b>
1.1. PURPOSE	
.....	
... 4	
1.2. SCOPE	
.....	
..... 5	
1.3. PROJECT OBJECTIVE	
.....	5
<b>2. SYSTEM</b>	
<b>REQUIRMENTS.....</b>	
<b>..... 6</b>	
2.1. HARDWARE	
.....	
6	
2.2. SOFTWARE	
.....	
6	
2.3. NETWORK	
.....	
. 6	
<b>3. PRE-REQUISITES</b>	
.....	<b>6</b>
<b>4. ARCHITECTURE.....</b>	
<b>..... 10</b>	
4.1. TECHNICAL ARCHITECTURE	
.....	11
<b>5. ER - DIAGRAM</b>	
.....	<b>11</b>
a. USERS ENTITY	
.....	12
b. COURSES ENTITY	
.....	12
c. RELATIONSHIP - “CAN”	
.....	12

<b>6. PROJECT STRUCTURE</b>	<b>13</b>
<b>7. APPLICATION FLOW</b>	<b>19</b>
Teacher	
.. 19 II. Student	
. 20	
III. Admin:	
. 20	
<b>8. PROJECT SETUP &amp; CONFIGURATION</b>	<b>20</b>
8.1. FRONT-END DEVELOPMENT	
DEVELOPMENT	
21	
8.2. BACK-END	
DEVELOPMENT	
21	
8.3. DATABASE DEVELOPMENT	
21	
<b>9. PROJECT IMPLEMENTATION &amp; EXECUTION</b>	<b>22</b>
9.1. SCREENSHOTS	
23	
<b>10. CONCLUSION</b>	<b>78</b>

## **1. INTRODUCTION :**

In recent years, the demand for online education has surged, driven by the need for flexible, accessible, and diverse learning options that cater to different schedules, locations, and learning preferences. An **Online Learning Platform (OLP)** is a digital system designed to meet this demand, providing a comprehensive educational environment that allows users to engage with content, track progress, and gain certification for completing courses. Built using the MERN stack, OLP leverages MongoDB for data storage, Express.js for server-side functionality, React for a dynamic front-end interface, and Node.js for back-end development, ensuring a scalable, efficient, and responsive learning platform.

OLP is built with a focus on accessibility and interactivity, allowing learners of all backgrounds and technical proficiency to navigate the platform easily. It

supports features like course enrollment, self-paced learning, discussion forums, live webinars, and progress tracking.

Moreover, the platform offers flexibility for both free and paid courses, catering to broad audience.

With roles for students, instructors, and administrators, OLP provides a structured yet adaptable framework for managing and delivering educational content, enhancing the learning experience, and supporting instructors in content management and user engagement. This documentation outlines the technical architecture, functional requirements, and user workflows of OLP, emphasizing how it integrates with modern educational needs to foster a robust online learning community.

## **1.1. PURPOSE :**

The purpose of the **Online Learning Platform (OLP)** is :

- ✚ To provide a centralized online platform that facilitates flexible, self-paced learning, accessible to users worldwide.
- ✚ To enable users to enroll in courses, track learning progress, and earn certifications, supporting personal and professional development.
- ✚ To offer an interactive and user-friendly interface that allows learners to navigate content easily and engage in discussion forums, live webinars, and assignments.
- ✚ To provide instructors with a streamlined system for creating, organizing, and managing course content effectively.
- ✚ To allow administrators to monitor platform operations, manage user activities, and ensure data security and integrity.
- ✚ To create a scalable, efficient, and cost-effective solution for educational institutions and independent educators looking to deliver content online.

## **1.2. SCOPE :**

The scope of the **Online Learning Platform (OLP)** encompasses the development, deployment, and maintenance of a comprehensive online education system using the MERN stack.

Key components and features within this scope includes :

- ✦ **User Management:** Support user roles for students, instructors, and administrators, including registration, login, and profile management.
- ✦ **Course Management:** Enable instructors to create, update, organize, and publish course materials such as video lectures, assignments, and reading materials.
- ✦ **Interactivity Tools:** Offer discussion forums, chat rooms, and live webinar functionality to foster communication between students and instructors.
- ✦ **Progress Tracking:** Implement features for learners to monitor their course progress and revisit content as needed.
- ✦ **Certification:** Issue digital certificates upon course completion to recognize student achievement and skill acquisition.
- ✦ **Payment and Subscription Options:** Provide a payment gateway for paid courses, offering both one-time purchases and subscription models.
- ✦ **Cross-Device Accessibility:** Ensure compatibility across various devices (PCs, tablets, smartphones) to allow access from any location with an internet connection.
- ✦ **Front-end & Back-end Integration:** Leverage MERN stack for efficient front-end and back- end communication and database management.
- ✦ **Admin Panel:** Include an administrative dashboard for monitoring user activity, managing course listings, and handling platform maintenance tasks.

### **1.3. PROJECT OBJECTIVE :**

The objective of the **Online Learning Platform (OLP)** project is to develop a versatile and accessible online education system using the MERN stack (MongoDB, Express.js, React, and Node.js). This platform aims to provide a seamless, user-friendly experience for students, instructors, and administrators by offering key features that facilitate effective online learning and course management.

## **2. SYSTEM REQUIRMENTS :**

### **2.1. HARDWARE :**

- ✦ **Operating System :** Windows 8 or higher
- ✦ **RAM :** 4 GB or more (8 GB recommended for smooth development experience)

### **2.2. SOFTWARE :**

- ✚ **Node.js** : LTS version for back-end and front-end development
- ✚ **MongoDB** : For database management using MongoDB Atlas or a local instance
- ✚ **React** : For front-end framework
- ✚ **Express.js** : For back-end framework
- ✚ **Git** : Version control
- ✚ **Code Editor** : e.g., Visual Studio Code
- ✚ **Web Browsers** : Two web browsers installed for testing compatibility (e.g., Chrome and Firefox)

### **2.3. NETWORK :**

#### ✚ **Bandwidth :**

30  
Mbps

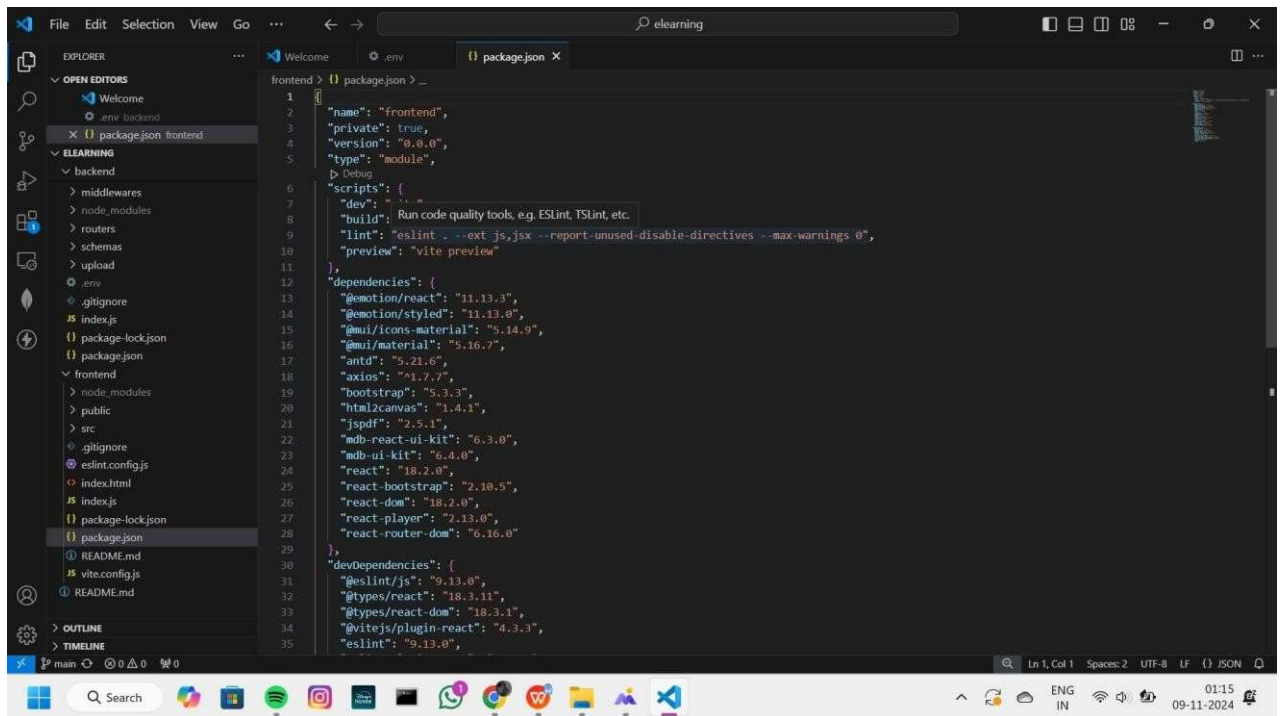
### **3. PRE-REQUISITES :**

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

#### □ **Vite:**

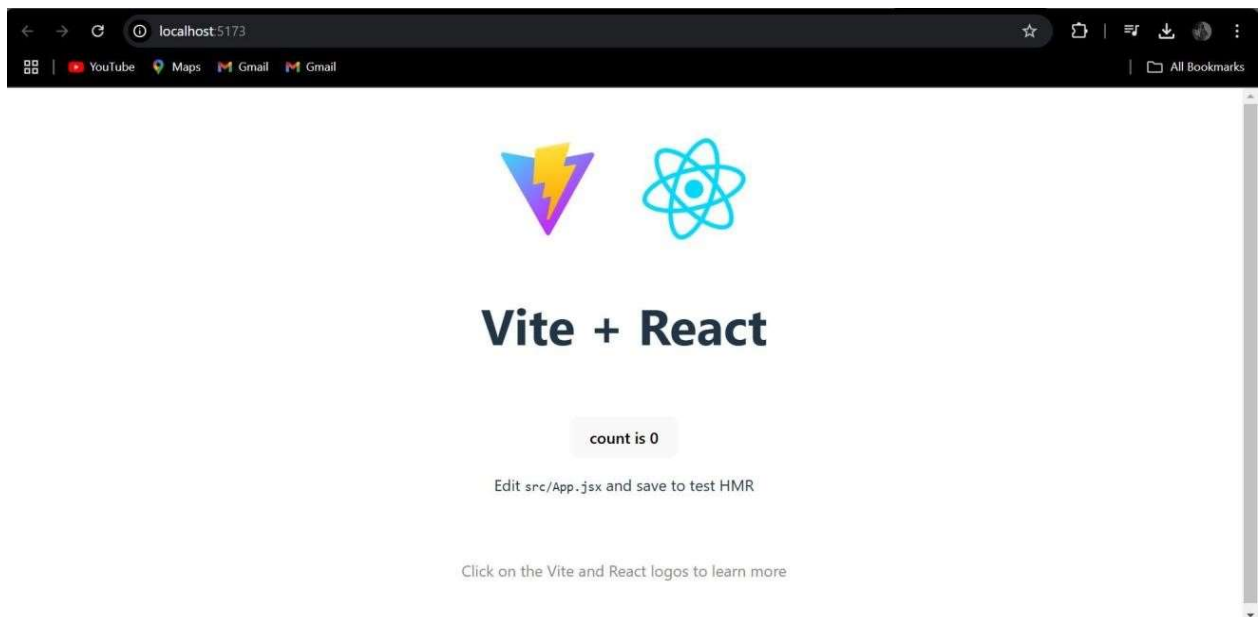
Vite is a new frontend build tool that aims to improve the developer experience for development with the local machine, and for the build of optimized assets for production (go live). Vite (or ViteJS) includes: a development server with ES \_native\_ support and Hot Module Replacement; a build command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

Installation : **npm create vite@latest**



The image shows a Visual Studio Code editor window with the 'package.json' file open. The file is for a project named 'frontend' and contains the following content:

```
1 {
2   "name": "frontend",
3   "private": true,
4   "version": "0.0.0",
5   "type": "module",
6   "scripts": {
7     "dev": "vite",
8     "build": "Run code quality tools, e.g. ESLint, TSLint, etc.",
9     "lint": "eslint . --ext js,jsx --report-unused-disable-directives --max-warnings 0",
10    "preview": "vite preview"
11  },
12  "dependencies": {
13    "@emotion/react": "11.13.3",
14    "@emotion/styled": "11.13.0",
15    "@mui/icons-material": "5.14.9",
16    "@mui/material": "5.16.7",
17    "antd": "5.21.6",
18    "axios": "1.7.7",
19    "bootstrap": "5.3.3",
20    "html2canvas": "1.4.1",
21    "jspdf": "2.5.1",
22    "mdb-react-ui-kit": "6.3.0",
23    "mdb-ui-kit": "6.4.0",
24    "react": "18.2.0",
25    "react-bootstrap": "2.10.5",
26    "react-dom": "18.2.0",
27    "react-player": "2.13.0",
28    "react-router-dom": "6.16.0"
29  },
30  "devDependencies": {
31    "@eslint/js": "9.13.0",
32    "@types/react": "18.3.11",
33    "@types/react-dom": "18.3.1",
34    "@vitejs/plugin-react": "4.3.3",
35    "eslint": "9.13.0",
36  }
37 }
```



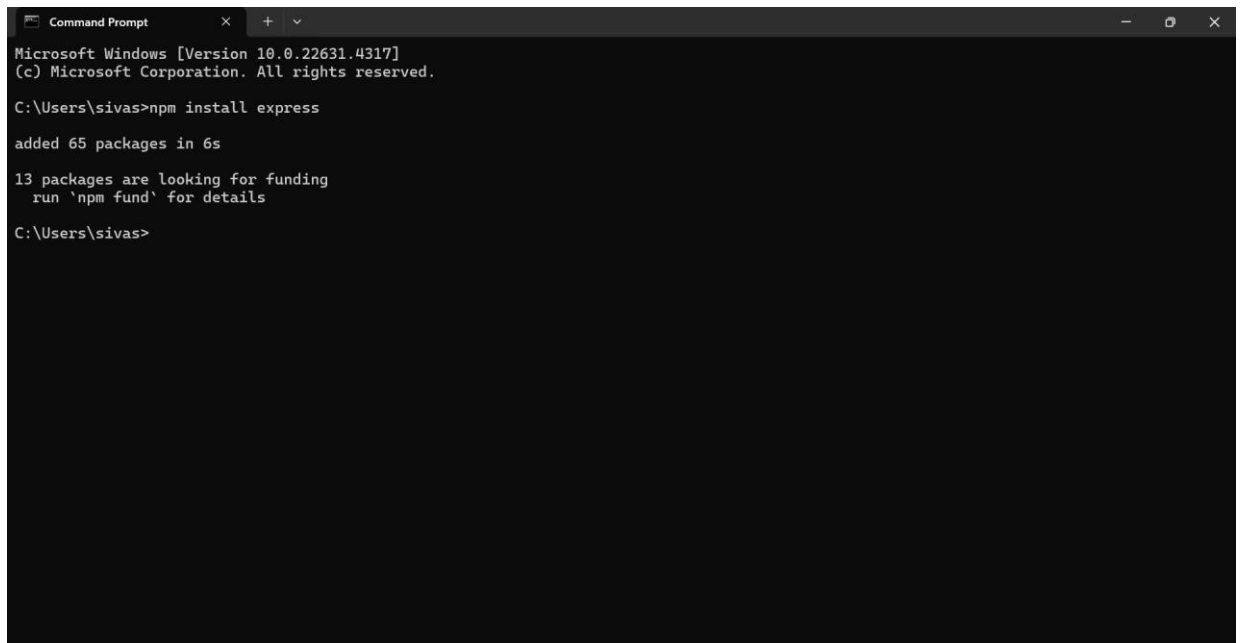


**Node.js and npm:** Node.js and npm: Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server-side. It provides a scalable and efficient platform for building network applications. Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side. Download: <https://nodejs.org/en/download/> Installation instructions:

### **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**

A screenshot of a Windows Command Prompt window. The title bar reads "Command Prompt". The window content shows the following text: "Microsoft Windows [Version 10.0.22631.4317] (c) Microsoft Corporation. All rights reserved. C:\Users\sivas>npm install express added 65 packages in 6s 13 packages are looking for funding run 'npm fund' for details C:\Users\sivas>".

```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sivas>npm install express

added 65 packages in 6s

13 packages are looking for funding
  run 'npm fund' for details

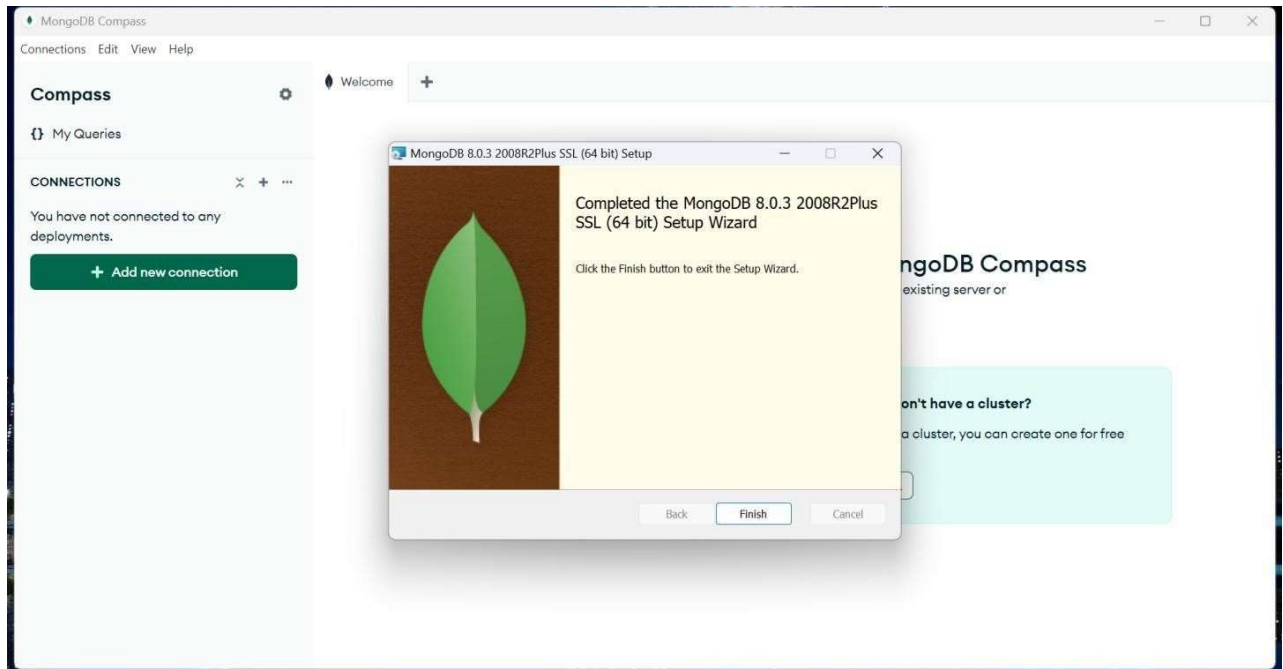
C:\Users\sivas>
```

## ❑ 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:**

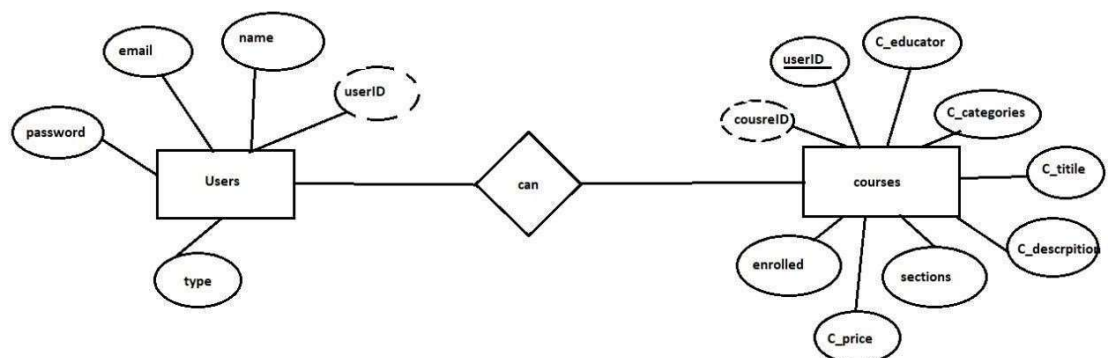
Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations.

### **4.1. TECHNICAL ARCHITECTURE :**

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

On the front-end side, it utilizes the bootstrap and material UI library to establish real-time and better UI experience for any user. On the back-end side, we employ Express.js frameworks to handle the server-side logic and communication. For data storage and retrieval, our back-end relies on MongoDB. MongoDB allows for efficient and scalable storage of user data and necessary information about the place.

For communication between the front-end and back-end, **RESTful API's** are implemented, facilitating smooth data exchange and modular integration. Additionally, **authentication and authorization** protocols, such as JWT (JSON Web



Tokens), are used to secure user access, ensuring role-based permissions for students, instructors, and admins. The platform includes **real- time communication channels** like chat and messaging, fostering interaction between students and faculty, while **adaptive learning algorithms** provide personalized learning paths.

For analytics, **data visualization tools** are incorporated to display metrics on course engagement and completion, aiding administrators in making data-driven decisions. Finally, third- party integration, such as **payment gateways** for premium content and **video streaming services** for course delivery, enhance the platform's functionality, while systematic updates and **security audits** maintain platform integrity and user data protection. This comprehensive set of technical components enables the OLP to offer a secure, scalable, and user-friendly online learning experience.

## **5. ER - DIAGRAM :**

Here there is 2 collections which have their own fields in :

### **a. USERS ENTITY**

□ Represents the users of the platform.

#### **Attributes:**

- ✚ **userID:** Unique identifier for each user (likely primary key).
- ✚ **name:** Name of the user.
- ✚ **email:** Email address associated with the user account.
- ✚ **password:** User's password for secure login. ✚ **type:** Indicates the type of user (e.g., student, educator, admin).

### **b. COURSES ENTITY**

□ Represents the various courses available on the platform.

#### **Attributes:**

- ✚ **courseID:** Unique identifier for each course.
- ✚ **C\_title:** Title of the course.

- † **C\_description:** Description of the course content.
- † **C\_educator:** The educator or instructor associated with the course.
- † **C\_categories:** Categories or tags to classify the course (e.g., Programming, Design).
- † **sections:** Indicates sections or modules within the course.
- † **C\_price:** Price of the course, which could relate to whether it's a free or premium course. † **enrolled:** Tracks the number of students or users enrolled in the course.

### c. RELATIONSHIP - "CAN"

- † This relationship shows the interaction between Users and Courses.
- † **Users (students) can:**
  - Browse courses.
  - Enroll in courses. ○ Track progress in courses they are enrolled in.
- † **Users (educators) can:**
  - Create, edit, and manage courses.
  - Upload course content such as videos, text, and interactive elements.
- † **Admin users can:**
  - Oversee all user activities. ○ Manage course listings, content, and enrollment data.

### Summary in

#### Use Case Points:

- † **User Registration/Login:** Users can register and log in with a unique userID, email, and password.
- † **Course Browsing and Enrollment:** Users can browse through available courses and enroll.
- † **Course Management for Educators:** Educators can create and manage courses, add content, and handle student enrollments.
- † **Content Interaction:** Enrolled students can interact with course materials and progress through sections/modules.

‡ **Payment Handling:** Courses may have prices, allowing for paid enrollment. ‡

**Data Tracking:** The platform tracks user enrollments and progress in each course.

## 6. PROJECT STRUCTURE :

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

### ○ Root Folders and Files :

#### ‡ **Front-end:**

The main project folder containing all files and folders for the front-end part of the application.

#### ‡ **node\_modules:**

Contains all the dependencies and modules installed via npm (Node Package Manager). This folder is automatically generated when dependencies are installed.

#### ‡ **public:**

Stores static files that can be served directly, such as index.html (the main HTML file for a React app), images, and other static assets. Files in this folder are accessible directly by the browser.

#### ‡ **src:**

This is the main source folder for the React components and other application code. All core logic, components, and styles are stored here. Folders and Files Inside src contains :

#### ○ **assets:**

This folder typically contains images, icons, or other media assets used throughout the frontend.

#### ○ **components:**

Houses all the components used in the application. Components are organized into

subfolders based on functionality or user roles (e.g., admin, common, user, student, teacher).

★ **admin Folder :**

Contains components specific to the admin functionalities of the application.

- a. **AdminHome.jsx**: The main dashboard or homepage component for admin users.
- b. **AllCourses.jsx**: Displays all courses available on the platform, likely for administrative oversight.

★ **common Folder :**

Contains shared components accessible by all users. **AllCourses.jsx**: Shows a list of all available courses (possibly a common view accessible by all users).

- a. **AxiosInstance.jsx**: Configures Axios for API requests, likely setting up base URLs or authorization headers.
- b. **Dashboard.jsx**: A general dashboard component that could serve as the main page for logged-in users.
- c. **Home.jsx**: The homepage or landing page for the application.
- d. **Login.jsx**: The login component for user authentication.
- e. **NavBar.jsx**: The navigation bar component, providing links to different parts of the application.
- f. **Register.jsx**: The registration component for new users.
- g. **UserHome.jsx**: The main page or dashboard for general users.

★ **user Folder :**

Contains subfolders for different user roles with specific components

✓ **student:**

- a. **CourseContent.jsx**: Displays the content of a course for students.
- b. **EnrolledCourses.jsx**: Lists the courses a student is enrolled in.
- c. **StudentHome.jsx**: The main page or dashboard for students.

✓ **teacher:**

- a. **AddCourse.jsx**: Component for teachers to add or create new courses.
- b. **TeacherHome.jsx**: The main page or dashboard for teachers, showing relevant information for instructors.

#### ○ **App.css:**

Contains global styles for the application, defining the look and feel of the entire frontend.

#### ○ **App.jsx:**

The root component of the application. It usually contains routes to various pages and loads other major components.

#### ○ **main.jsx:**

The main entry point for the application, where the React app is rendered into the HTML document (usually index.html in the public folder).

#### ○ **.eslintrc.js:**

Configuration file for ESLint, a tool used to enforce coding standards and style in JavaScript code.

This structure is well-organized, with separate folders for different user roles (admin, student, teacher), as well as shared components for common functionality. Each user role has its specific set of components to encapsulate role-based features, making it easier to manage and extend the code. This setup supports scalability and maintainability for a comprehensive web application.

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

#### ○ **Folder and File Structure**

##### ‡ **Config:**

- ★ This folder usually contains configuration files for setting up connections or other environmental variables needed by the application.



- ★ It likely has settings for connecting to the database, such as database URL and credentials, along with any other application-level configurations.

#### ‡ **Controllers:**

- ★ **adminController.js**: Contains functions to handle admin-specific actions, such as managing courses, users, or payments.
- ★ **userControllers.js**: Manages actions related to regular users, such as registration, login, enrolling in courses, or fetching user-specific information.

#### ‡ **Middlewares:**

- ★ **authMiddleware.js**: Contains middleware for handling authentication and authorization. It checks if users are authenticated before allowing access to certain routes, ensuring only logged-in users can access restricted areas of the application.

#### ‡ **Routers:**

- ★ **adminRoutes.js**: Defines API endpoints specific to admin functionalities and maps each endpoint to corresponding controller functions in adminController.js.
- ★ **userRoutes.js**: Defines API endpoints for user functionalities and maps them to the functions in userControllers.js. This could include endpoints for login, registration, course enrollment, etc.

#### ‡ **Schemas:**

- ★ This folder includes all the models that define the structure of different collections or tables in the database.
- ★ **courseModel.js**: Defines the schema for courses, including properties like title, description, price, educator, and other relevant course details.
- ★ **coursePaymentModel.js**: Handles the schema for course payment details, including fields for transaction IDs, payment status, amount, etc.
- ★ **enrolledCourseModel.js**: Manages information on users who have enrolled in specific courses, likely linking users to course IDs.

- ★ **userModel.js**: Defines the user schema, including fields like name, email, password, and role (e.g., student or teacher).

#### ✚ uploads

- ★ This folder may be used for storing files uploaded by users, such as profile pictures or course materials. It ensures that any uploaded assets are kept in an organized directory.

#### ✚ Environment and Configuration Files

- ★ **.env**: Stores environment variables such as database credentials, API keys, and other sensitive information that should not be hard-coded.
- ★ **.gitignore**: Specifies files and folders to be ignored by Git, such as node\_modules, .env, and other directories that do not need to be tracked.

#### ✚ Entry Points and Package Files

- ★ **index.js**: The main entry point for the backend application. It typically sets up the server, connects to the database, and initializes middleware, routes, and error handling.
- ★ **package.json**: Lists dependencies, scripts, and metadata for the project. It manages the backend libraries and tools required to run the server.
- ★ **package-lock.json**: Records the exact versions of dependencies installed in node\_modules, ensuring consistency across installations

This backend structure supports a modular approach to handling an online platform, where different sections are divided by purpose. Controllers handle specific actions, routes define accessible API endpoints, and middlewares enforce security and validation. Models structure the database, and configurations ensure easy setup across environments.

Here's an basic explanation for the third image which it includes:

#### ✚ .gitignore

- ★ Specifies files and directories that should be ignored by Git, preventing them from being tracked in version control. Common entries include `node_modules`, `.env`, and any other files that are either sensitive or unnecessary for sharing in the repository.

#### ✚ **index.html**

- ★ The main HTML file for the application. It serves as the entry point for the front-end application, where JavaScript bundles and stylesheets are injected. Typically, this file includes a root `<div>` where the front-end framework (e.g., React) mounts the application.

#### ✚ **package-lock.json**

- ★ Automatically generated by npm to track the exact versions of dependencies and sub-dependencies installed in the project. This file ensures that other developers or environments installing the project will have the exact same dependency versions, which improves consistency and reliability.

#### ✚ **package.json**

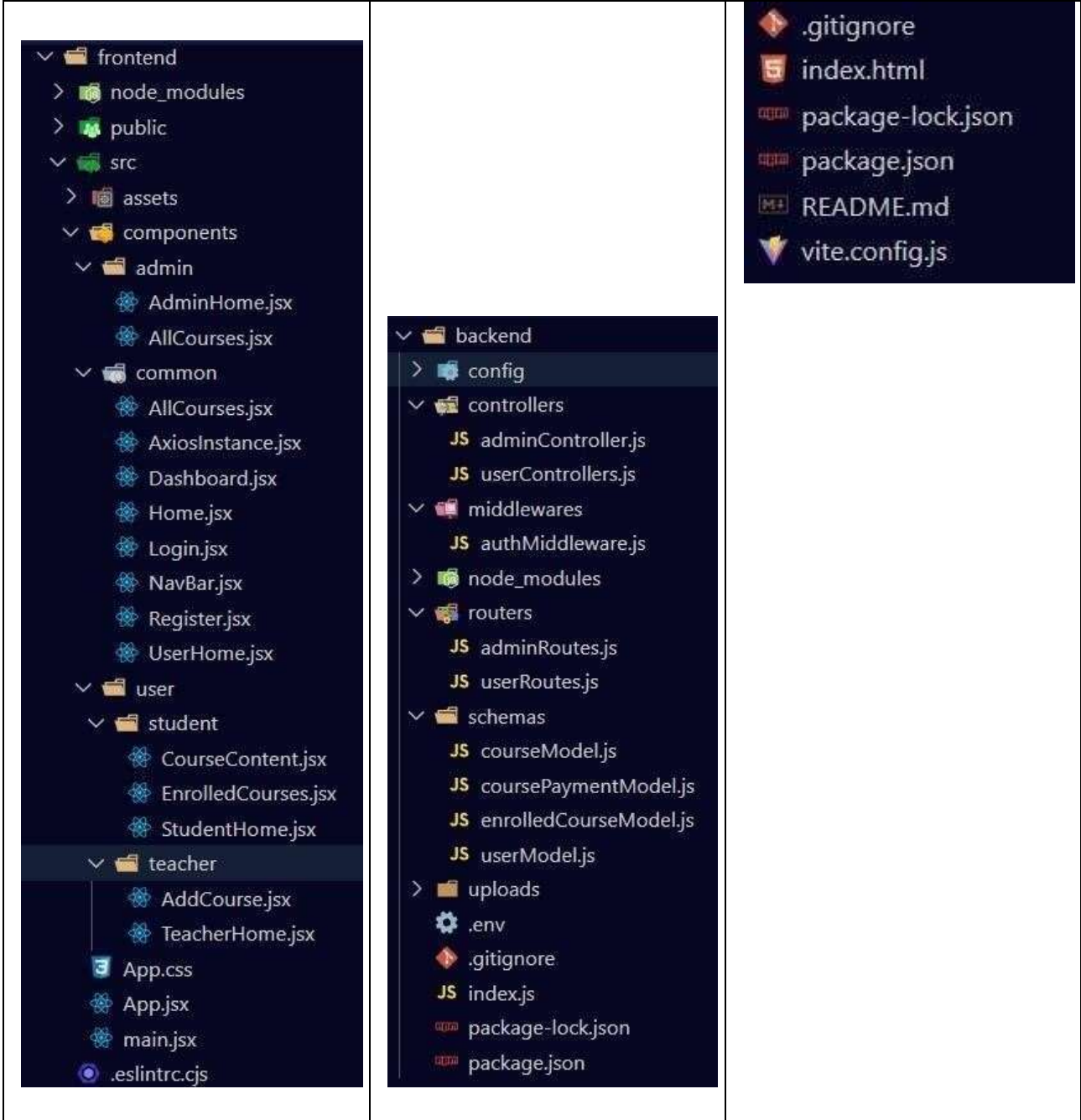
- ★ The main configuration file for the Node.js project. It includes metadata about the project (like its name and version), as well as scripts, dependencies, and other configurations. This file is essential for installing and managing packages and for running tasks like `npm start` or `npm build`.

#### ✚ **README.md**

- ★ A markdown file that typically contains documentation for the project. It often includes an introduction, setup instructions, usage information, and any other details that help developers understand and contribute to the project.

#### ✚ **vite.config.js**

- ★ The configuration file for Vite, a fast build tool and development server for modern web projects. This file is used to customize Vite's behavior, such as setting up plugins, defining alias paths, and configuring the development and build environments.



This set of files represents the foundational setup for a Vite-based front-end project, including configuration, documentation, and project dependencies.

## 7. APPLICATION FLOW :

The project has a user called– teacher and student and other will be Admin which takes care of all the user. The roles and responsibilities of these users can be inferred from the API endpoints defined in the code. Here is a summary:

### I. Teacher:

- ✚ Can add courses for the student.
- ✚ Also delete the course if no student enrolled in it or any other reasons.
- ✚ Also add sections to courses.

## II. Student:

- ✚ Can enroll in an individual or multiple course.
- ✚ Can start the course where it has stopped.
- ✚ Once the course is completed, they can download their certificate of completion of the course.
- ✚ For paid course, they need to purchase it and then they can start the course ✚ They can filter out the course by searching by name, category, etc

## III. Admin:

- ✚ They can alter all the course that are present in the app.
- ✚ Watch out all kind of users in app. ✚ Record all the enrolled all the student that are enrolled in course.

## 8. PROJECT SETUP & CONFIGURATION :

### ✚ Folder Setup

- Create Frontend and Backend folders

### 8.1. FRONT-END DEVELOPMENT :

For Frontend, we use require dependencies as follows, they are:

- ✚ React, Material , UI, Bootstrap and React - bootstrap
- ✚ Axios, Antd, Mdb-react-ui-kit

```
{
  "name": "frontend",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint . --ext js,jsx --report-unused-disable-directives --max-warnings 0",
    "preview": "vite preview"
  },
  "dependencies": {
    "@emotion/react": "^11.11.1",
    "@emotion/styled": "^11.11.0",
    "@mui/icons-material": "^5.14.9",
    "@mui/material": "^5.14.9",
    "axios": "^1.5.0",
    "bootstrap": "^5.3.2",
    "html2canvas": "^1.4.1",
    "jspdf": "^2.5.1",
    "mdb-react-ui-kit": "^6.1.0",
    "mdb-ui-kit": "^6.4.0",
    "react": "^18.2.0",
    "react-bootstrap": "^2.8.0",
    "react-dom": "^18.2.0",
    "react-player": "^2.13.0",
    "react-router-dom": "^6.16.0"
  },
  "devDependencies": {
    "@types/react": "^18.2.15",
    "@types/react-dom": "^18.2.7",
    "@vitejs/plugin-react": "^4.0.3",
    "eslint": "^8.45.0",
    "eslint-plugin-react": "^7.32.2",
    "eslint-plugin-react-hooks": "^4.6.0",
    "eslint-plugin-react-refresh": "^0.4.3",
    "vite": "^4.4.5"
  }
}
```

## 8.2. BACK-END DEVELOPMENT :

### ✚ Setup express server

- Create index.js file in the server (backend folder).
- define port number, mongodb connection string and JWT key in env file to access it.
- Configure the server by adding cors, body-parser.

### ✚ Add authentication:

- For this you need to make middleware folder and in that make authMiddleware.js file for the authentication of the projects and can use in.

For Backend, we use require dependencies as follows, they are:

✚ Cors, bcryptjs, multer,dotenv

✚ Express, mongoose, nodemon, jsonwebtoken

```
{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1",
    "start": "nodemon index"
  },
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "cors": "^2.8.5",
    "dotenv": "^16.3.1",
    "express": "^4.18.2",
    "jsonwebtoken": "^9.0.2",
    "mongoose": "^7.5.2",
    "multer": "^1.4.5-lts.1",
    "nodemon": "^3.0.1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

### 8.3. DATABASE DEVELOPMENT :

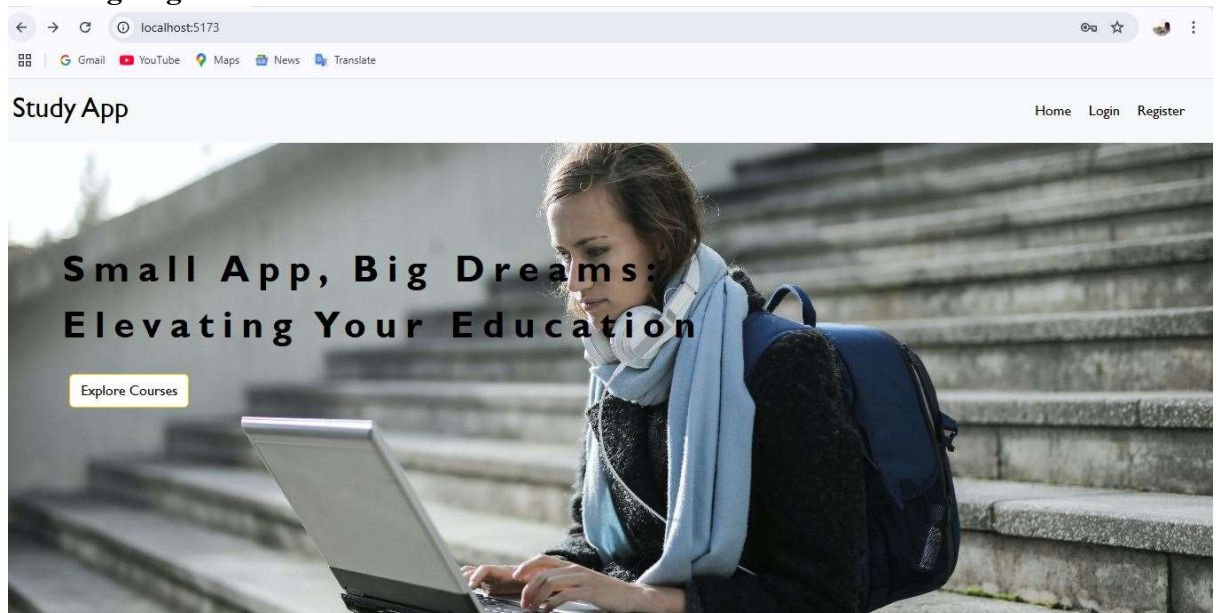
#### ✚ **Configure MongoDB**

- Import Mongoose
- Add database connection from config.js file present in config folder.
- Create a model folder to store all the DB schemas.

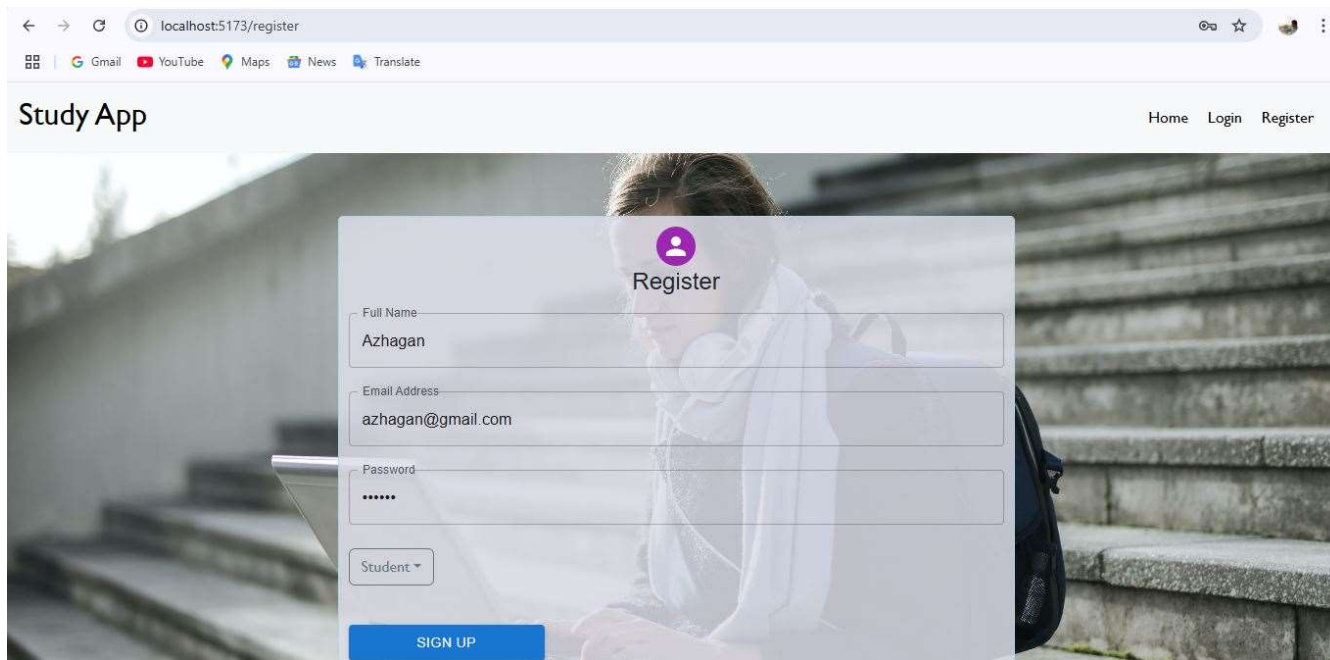
### 9. PROJECT IMPLEMENTATION & EXECUTION :

#### 9.1. SCREENSHOTS :

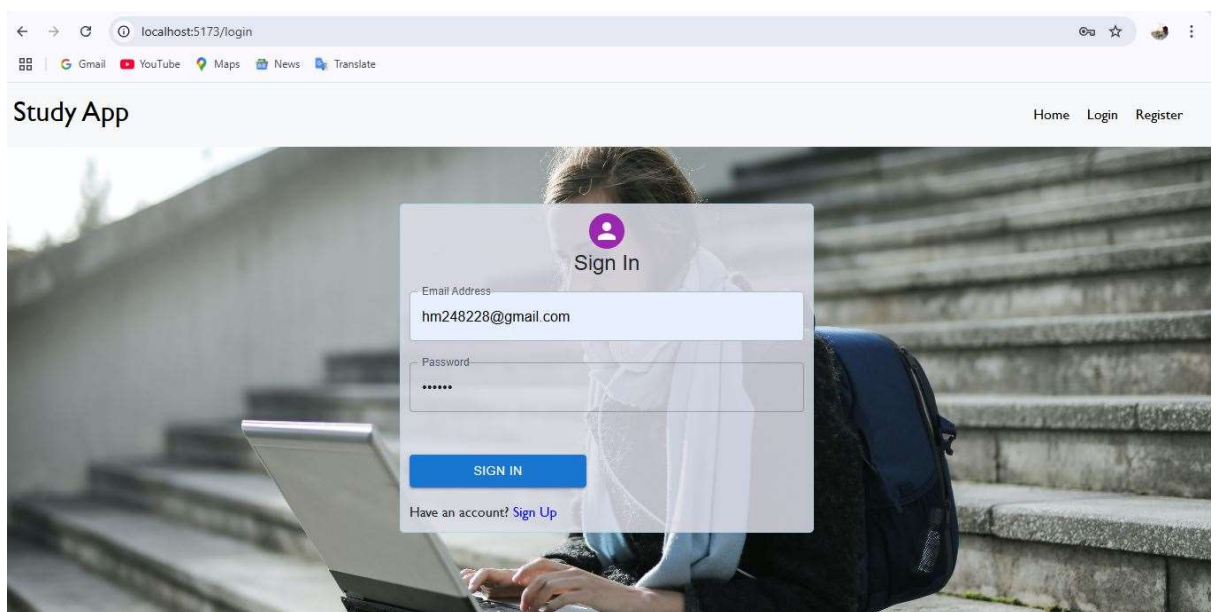
##### **Landing Page**



##### **Student Register Page**



## Student Login Page

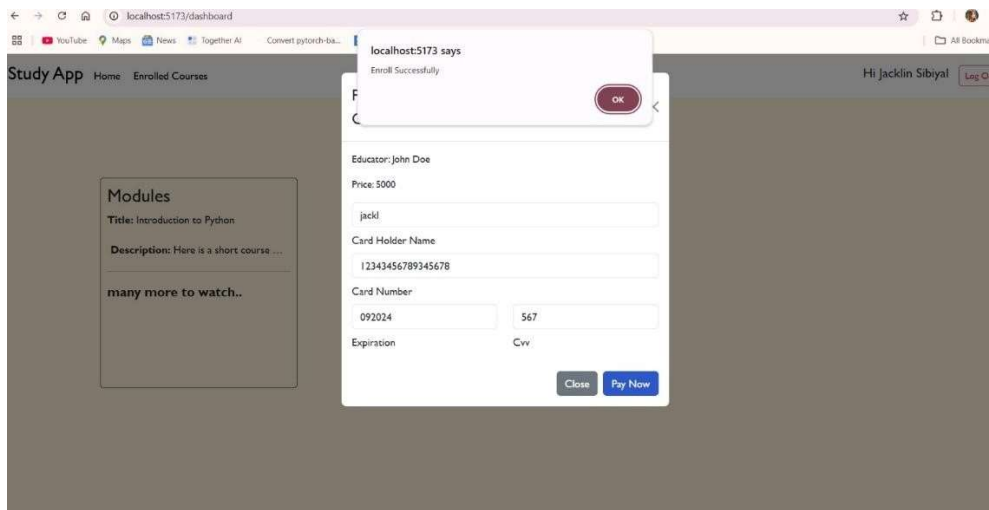


## Student Home Page





## Payment Details




## Student Course start



## Teacher Register

localhost:5173/register

Gmail YouTube Maps News Translate



### Register

Full Name

Teacher1

Email Address

teacher1@gmail.com

Password

\*\*\*\*\*

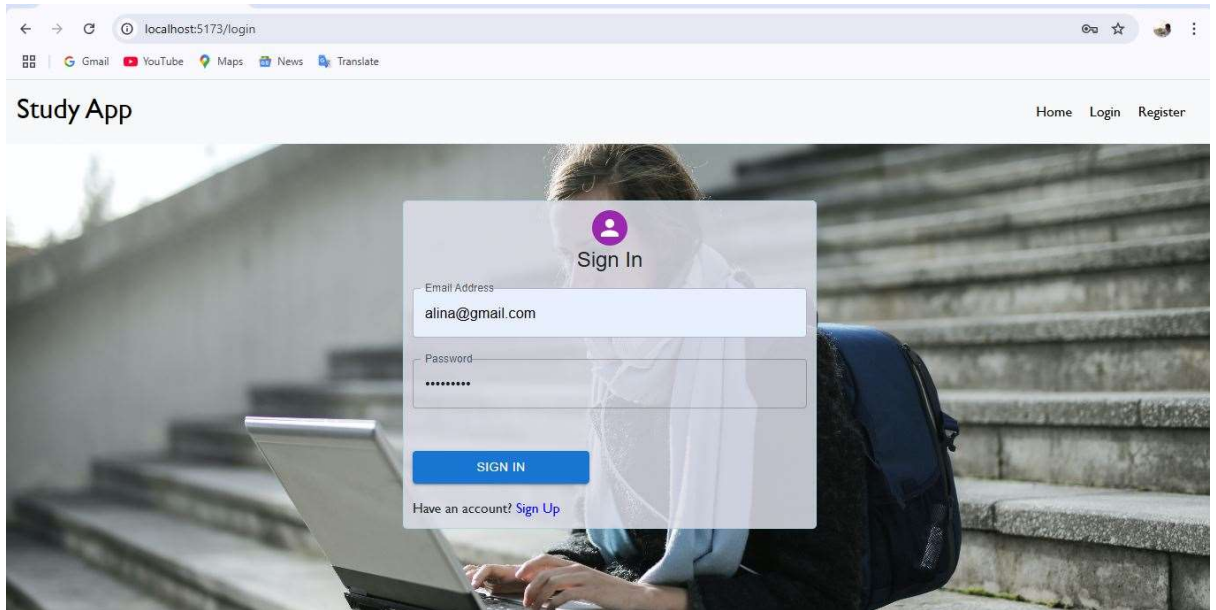
Teacher

SIGN UP

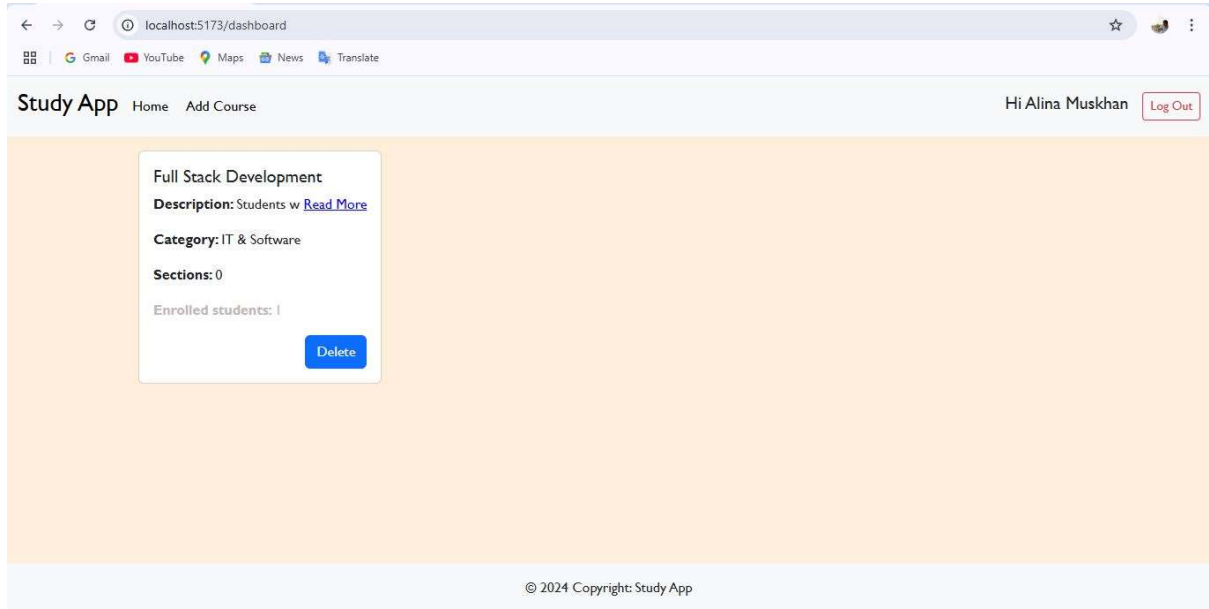
Have an account? [Sign In](#)

© 2024 Copyright: Study App

## Teacher Login Page



## Teacher Home Page



# Add Course

Study App

HomeAdd Course

Hi Mohanapriya

Log Out

Course Type

Select categories

Course Title

Enter Course Title

Course Educator

Enter Course Educator

Course Price(Rs.)

for free course, enter 0

Course Description

Enter Course description

+ Add Section

Submit

© 2024 Copyright: Study App

22°C

Search

ENG

IN

05:42

13.11.2024

## Database Details

The image displays two screenshots of the MongoDB Compass interface, showing database details for two different databases: 'E\_learning' and 'flightBookingMERN'.

**Top Screenshot (E\_learning / flightBookingMERN / users):**

- The interface shows the 'users' collection in the 'flightBookingMERN' database within the 'E\_learning' project.
- The 'Documents' tab is active, displaying two user documents.
- The first document is a JSON object with fields: `_id`, `username`, `email`, `usertype`, `password`, and `approval`.
- The second document is a JSON object with fields: `_id`, `username`, `email`, `usertype`, `password`, and `approval`.

**Bottom Screenshot (E\_learning / ims / users):**

- The interface shows the 'users' collection in the 'ims' database within the 'E\_learning' project.
- The 'Documents' tab is active, displaying three user documents.
- The first document is a JSON object with fields: `_id`, `name`, `email`, `password`, `type`, `createdAt`, and `updatedAt`.
- The second document is a JSON object with fields: `_id`, `name`, `email`, `password`, `type`, `createdAt`, and `updatedAt`.
- The third document is a JSON object with fields: `_id`, `name`, `email`, `password`, `type`, `createdAt`, and `updatedAt`.

MongoDB Compass - E\_learning /ms/enrolledcourses

Connections Edit View Collection Help

Compass

My Queries

CONNECTIONS (1)

Search connections:

E\_learning

- FlightBookingMERN
  - bookings
  - flights
  - users
- admin
- config
- lms
  - coursepayments
  - courses
  - enrolledcourses
  - users
- local

E\_learning > lms > enrolledcourses

Documents Aggregations Schema Indexes Validation

Type a query: { field: 'value' } or Generate query

Explain Reset Find Options

ADD DATA EXPORT DATA UPDATE DELETE

25 1-2 of 2

```
{
  "_id": ObjectId("6733ce0f1b013ac28a67131d"),
  "courseId": ObjectId("6733ce571b013ac28a671315"),
  "userId": ObjectId("6733ce7d1b013ac28a671310"),
  "courseLength": 1,
  "progress": Array (empty),
  "createdAt": 2024-11-12T21:52:15.442+00:00,
  "updatedAt": 2024-11-12T21:52:15.442+00:00,
  "__v": 0
}
```

```
{
  "_id": ObjectId("6733ce9c1b013ac28a671337"),
  "courseId": ObjectId("6733ce571b013ac28a67132f"),
  "userId": ObjectId("6733ce7d1b013ac28a671310"),
  "courseLength": 0,
  "progress": Array (empty),
  "createdAt": 2024-11-12T21:54:36.054+00:00,
  "updatedAt": 2024-11-12T21:54:36.054+00:00,
  "__v": 0
}
```

26°C Rain showers

MongoDB Compass - E\_learning /ms/courses

Connections Edit View Collection Help

Compass

My Queries

CONNECTIONS (1)

Search connections:

E\_learning

- FlightBookingMERN
  - bookings
  - flights
  - users
- admin
- config
- lms
  - coursepayments
  - courses
  - enrolledcourses
  - users
- local

E\_learning > lms > courses

Documents Aggregations Schema Indexes Validation

Type a query: { field: 'value' } or Generate query

Explain Reset Find Options

ADD DATA EXPORT DATA UPDATE DELETE

25 1-3 of 3

```
{
  "C.educator": "Jackline",
  "C.title": "Data science",
  "C.categories": "IT & Software",
  "C.price": "free",
  "C.description": "Enhance your skills",
  "sections": Array (1),
  "enrolled": 1,
  "createdAt": 2024-11-12T21:49:06.953+00:00,
  "updatedAt": 2024-11-12T21:52:15.442+00:00,
  "__v": 0
}
```

```
{
  "_id": ObjectId("6733ce4d1b013ac28a671328"),
  "userId": "6733ce5c1b013ac28a67133c",
  "C.educator": "Jackline",
  "C.title": "Stock market",
  "C.categories": "Finance & Accounting",
  "C.price": "4999",
  "C.description": "n",
  "sections": Array (empty),
  "enrolled": 0,
  "createdAt": 2024-11-12T21:53:17.617+00:00,
  "updatedAt": 2024-11-12T21:53:17.617+00:00,
  "__v": 0
}
```

```
{
  "_id": ObjectId("6733ce571b013ac28a67132f"),
  "userId": "6733ce5c1b013ac28a67133c",
  "C.educator": "Jackline",
  "C.title": "Data science",
  "C.categories": "IT & Software",
  "C.price": "free",
  "C.description": "Enhance your skills",
  "sections": Array (1),
  "enrolled": 1,
  "createdAt": 2024-11-12T21:49:06.953+00:00,
  "updatedAt": 2024-11-12T21:52:15.442+00:00,
  "__v": 0
}
```

26°C Rain showers

## **10. CONCLUSION :**

The conclusion of this project focuses on integrating a full-stack application, combining a React front end with a Node.js and Express back end, connected to a MongoDB database. By implementing modular code structure, secure authentication, and efficient data handling, this project successfully delivers an online learning platform (OLP) that allows different user roles— admin, teacher, and student—to manage and access course-related functionalities.

Through the integration of various tools and libraries like **Axios** for HTTP requests, **Multer** for handling file uploads, **JWT** for authentication, and **Mongoose** for database management, the platform provides a seamless and responsive user experience. Additionally, security measures, such as using **bcryptjs** for password hashing and **CORS** for controlling cross- origin access, enhance the robustness and integrity of the system.

In conclusion, this project showcases an effective implementation of a scalable and interactive learning management system by utilizing modern web development practices and technologies. It serves as a foundation for further enhancements, such as incorporating additional features, refining the user interface, or optimizing performance for large-scale deployment.

### **Github Project Link :**

<https://github.com/MohammedHasanMustafa/OLP>