Smart Internz

NAAN MUDHALVAN

நான் முதல்வன்
உலகை வெல்லும் இளைய தமிழகம்

ONLINE LEARNING PLATFORM USING **MERN** STACK

SMARTBRIDGE
Let's Bridge the Gap

By Team Members Of Final Year IT 'A'

TEAM K

APARNA K	311421205008
DHANUSH MS	311421205016
DINESH M	311421205019
LAVANYA B	311421205042

ABSTRACT :

The **Online Learning Platform (OLP)** is a **MERN-based web application** designed to offer accessible, self-paced, and interactive learning opportunities. It provides a range of features, including user-friendly navigation, course management, certification, and payment options, making it suitable for learners and instructors alike. This platform enables students to gain knowledge in a structured, engaging manner while instructors can efficiently create, organize, and manage courses. With flexible accessibility across devices, OLP delivers a modern learning experience tailored to individual schedules and needs.

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

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.

- It is 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.

- 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.

PURPOSE :

- 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 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.

SCOPE :

- **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.

SYSTEM REQUIRMENTS :

❖ HARDWARE :

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

❖ 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)

❖ NETWORK :

- ✓ **Bandwidth :** 30 Mbps

PRE-REQUISITES :

✓ **Vite + React:**

- Vite + React 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**

✓ Node.js

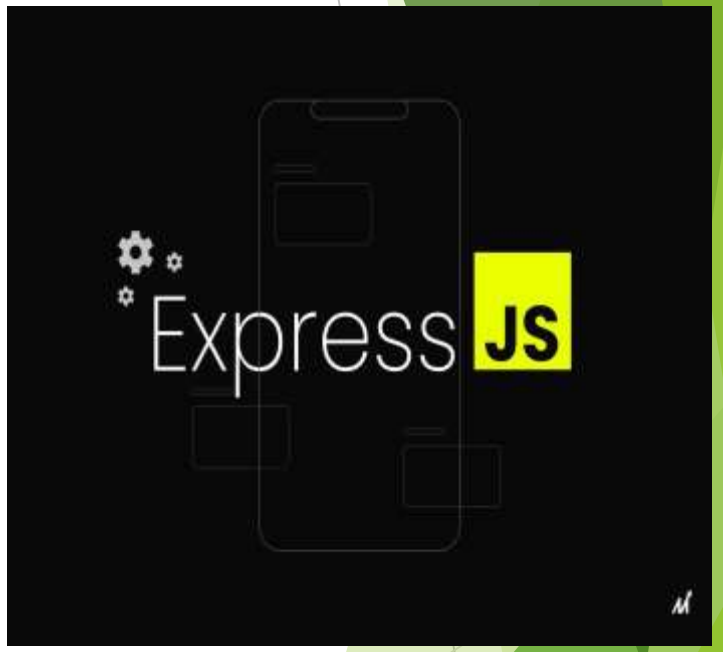
- Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server-side. Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

- Download: **https://nodejs.org/en/download/**
- Installation instructions: **https://nodejs.org/en/download/package-manager/**
- Run "**npm init**" to get default dependencies

✓ Express

- Express. j s i s 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. It **handles server-side routing, middleware, and API development.**

- Installation: Open your command prompt or terminal to run the following command:
  npm install express



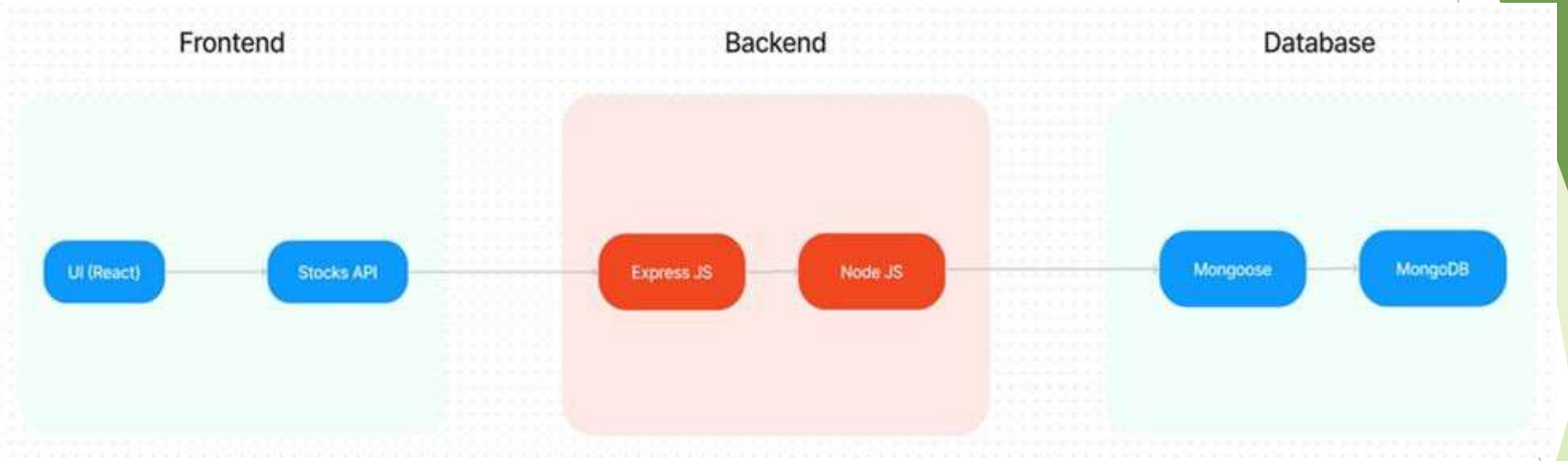
✓ Mongo DB

- 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.**

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

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

ARCHITECTURE :



TECHNICAL 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.

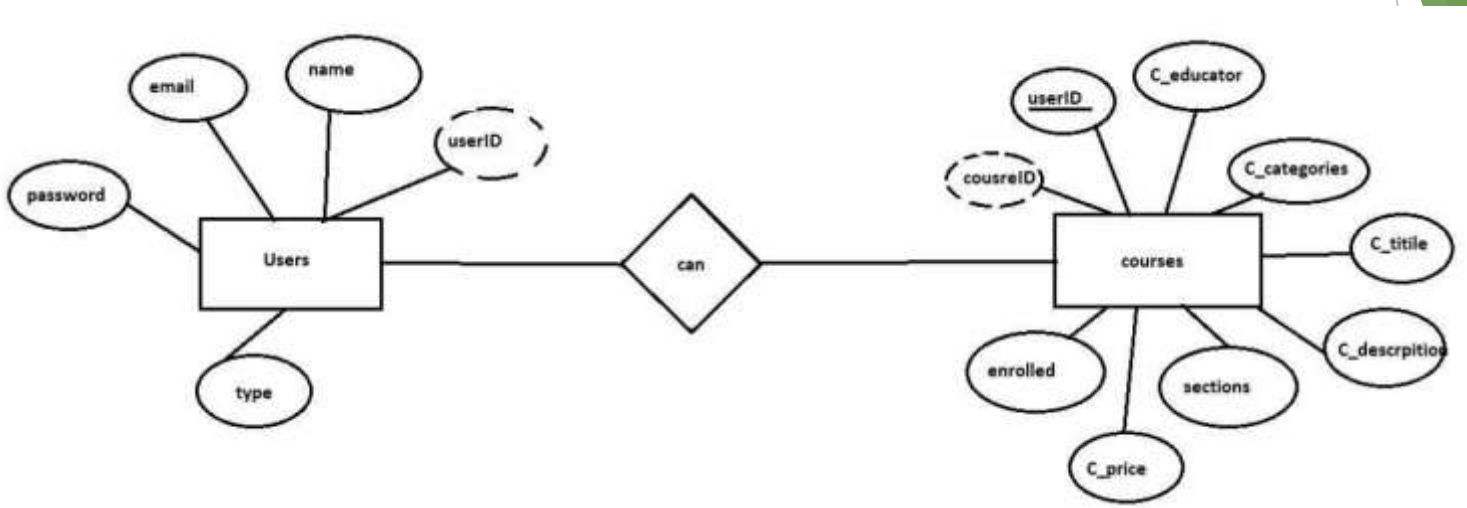
- The frontend utilizes the bootstrap and material UI library to establish real-time and better UI experience for any user.

- 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.

ER - DIAGRAM :



- 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.

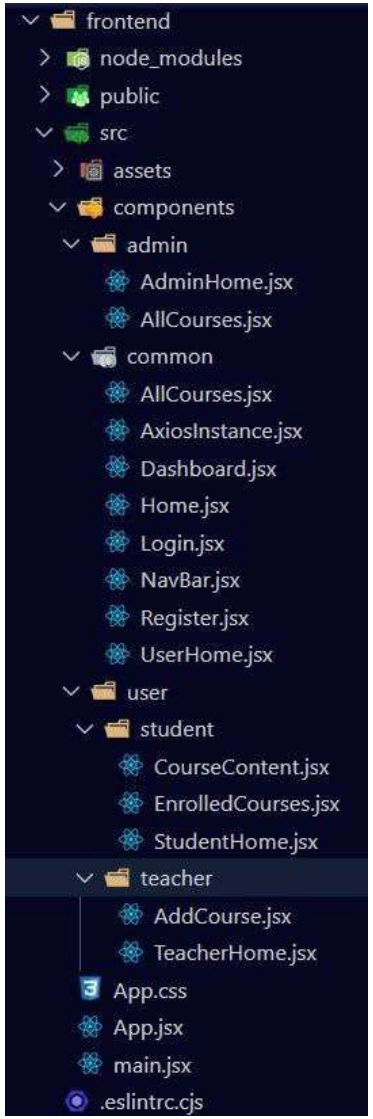
❖**C_categories:** Categories or tags to classify the course (e.g., Programming, Design).

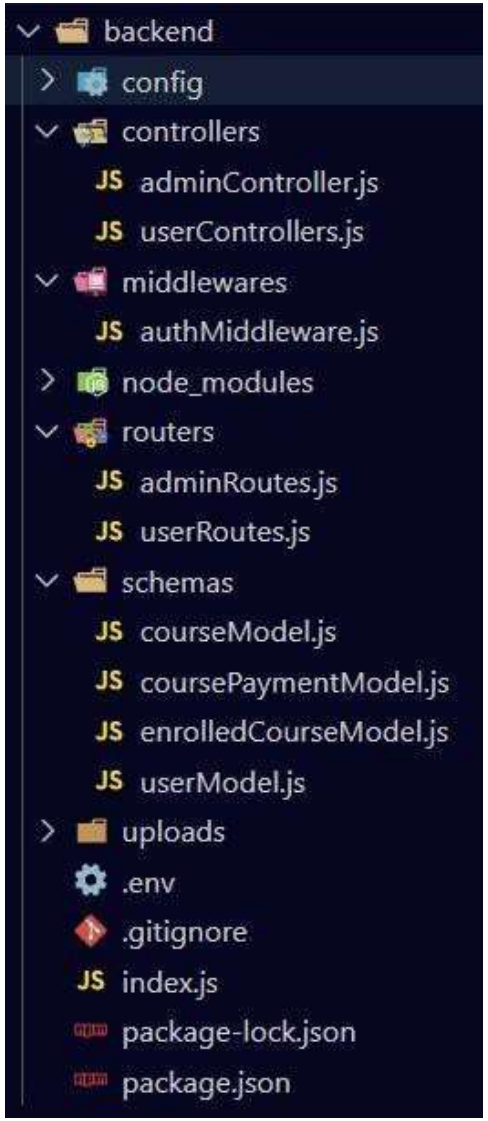
❖• **sections:** Indicates sections or modules within the course.

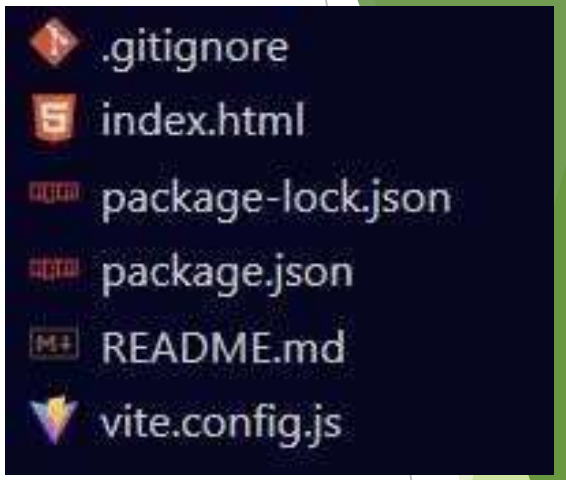
❖• **C_price:** Price of the course, which could relate to whether it’s a free or premium course.

❖• **enrolled:** Tracks the number of students or users enrolled in the course.

PROJECT STRUCTURE :







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

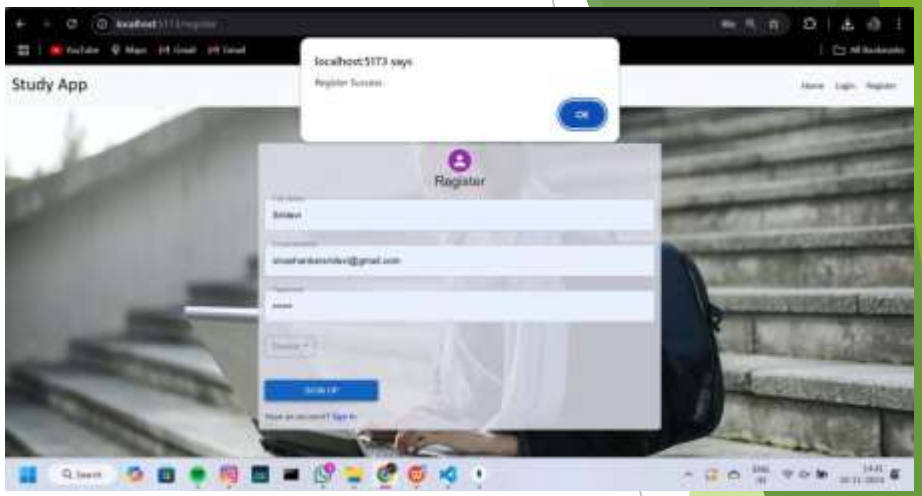
III. Admin:

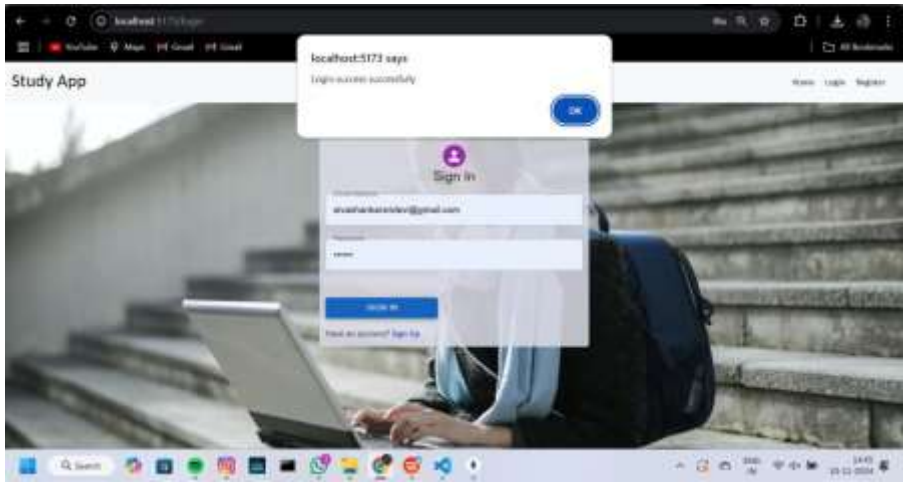
- They can alter all the course that are present in the app.
- Watch out all kind of users in app.
- Record all the enrolled all the student that are enrolled in course.

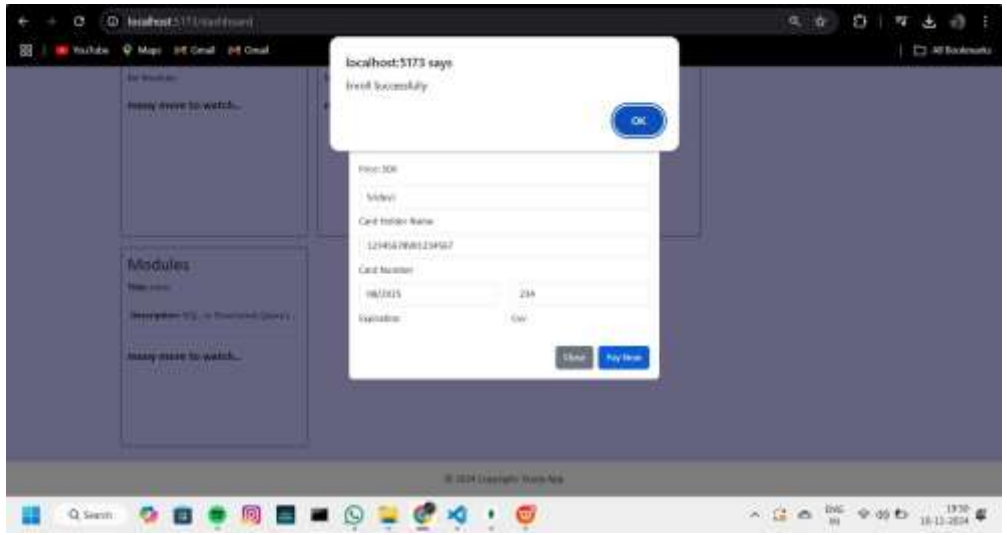
To Run the code use the following command:

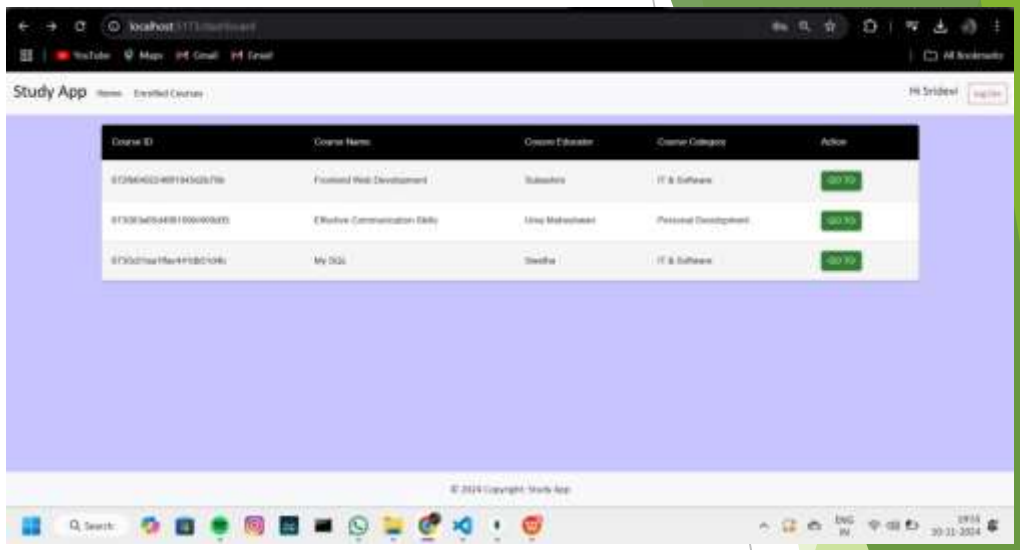
- **Frontend :** cd frontend > npm install > npm run dev
- **Backend :** cd backend > npm install > npm start

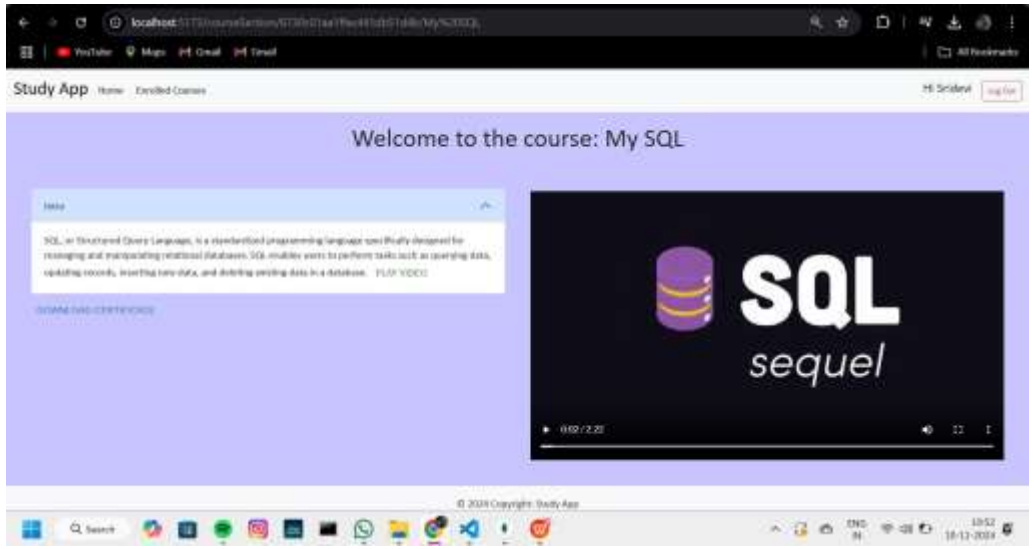
PROJECT EXECUTION SCREENSHOTS (Student Side) :

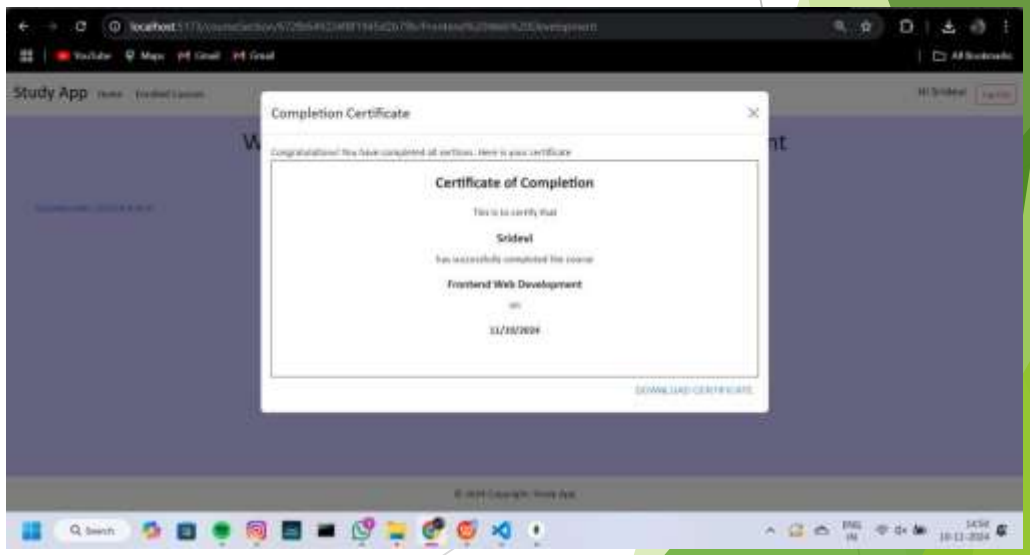




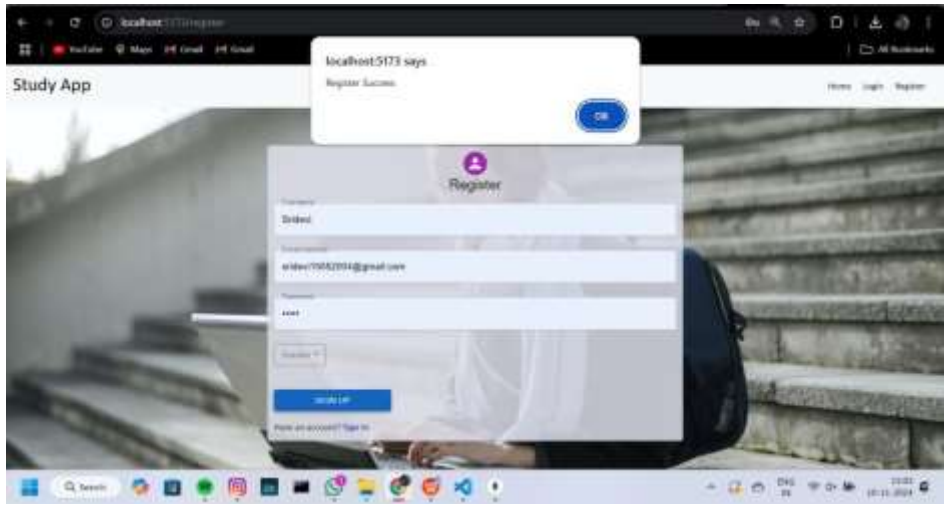

localhost:5173 says
Enroll Successfully
OK
Modules

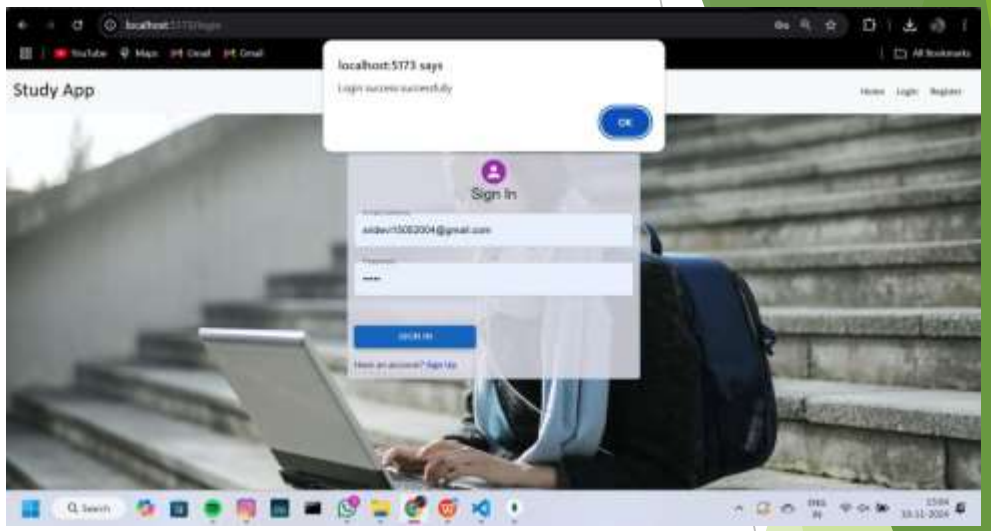

Study App
Hi Sridevi
Course ID
Course Name
Course Educator
Course Category
Action

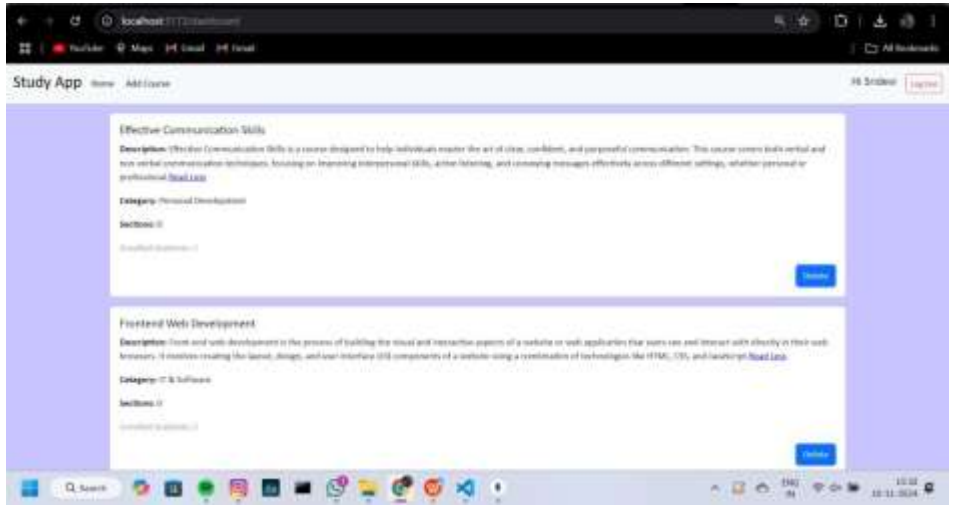

Study App
Hi Sridevi
Welcome to the course: My SQL
SQL
sequel

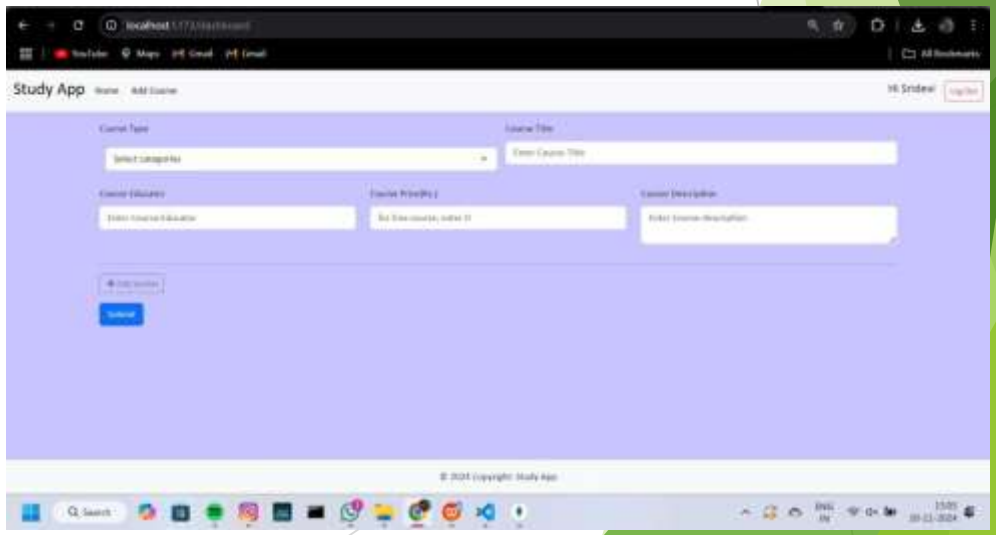

Study App
Hi Sridevi
Completion Certificate
Certificate of Completion
Sridevi
Frontend Web Development

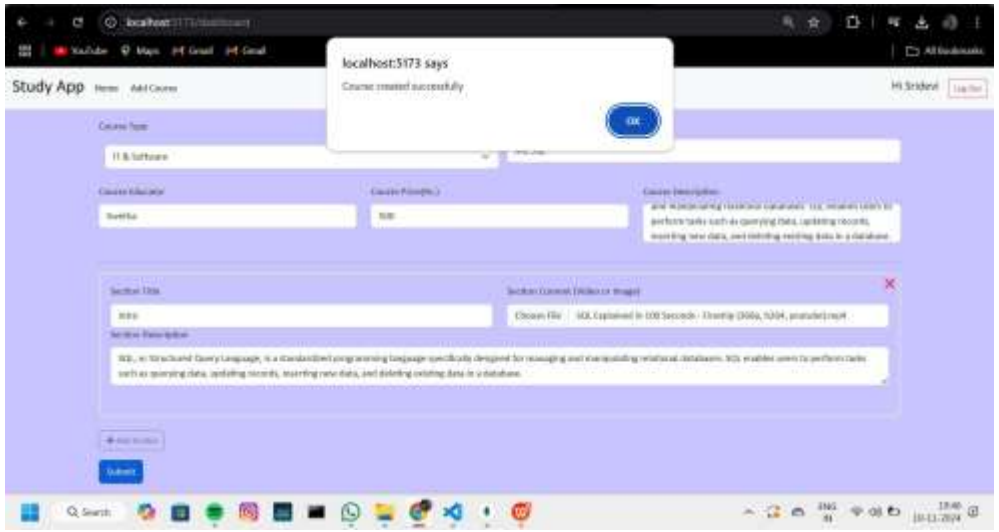
PROJECT EXECUTION SCREENSHOTS (Teacher Side) :

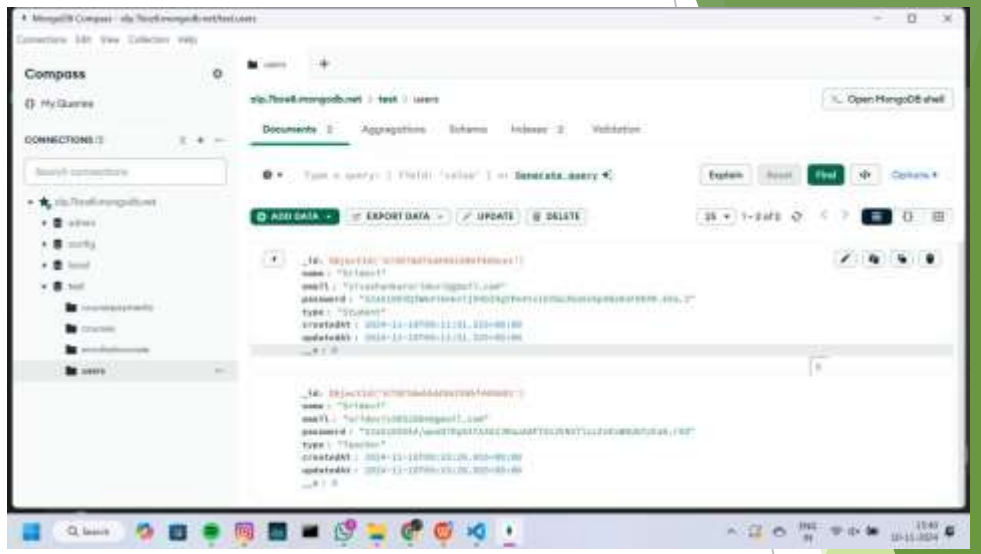


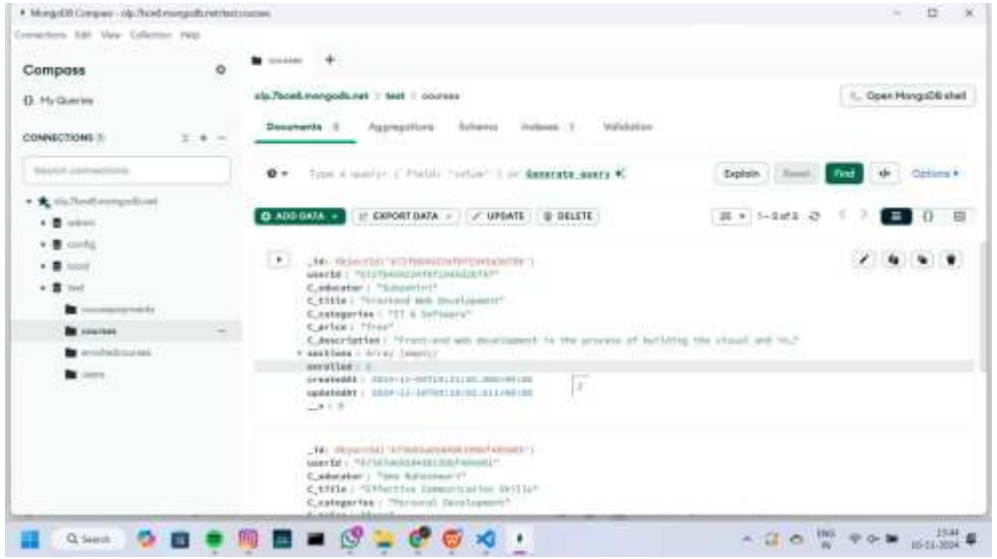


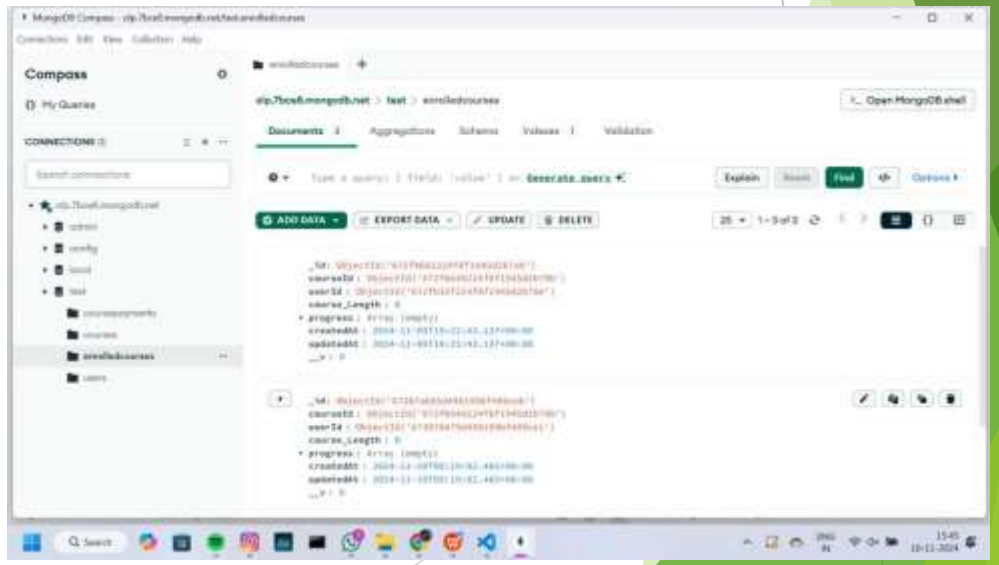





localhost:5173 says
Course created successfully
OK
Study App


Compass


Compass


Compass

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.

DEMO LINKS :

Github Link :
https://github.com/Aparnadivakaran/Online_learning_platform

Demo Video Link :
https://drive.google.com/file/d/15UELKcdRCkgHwP1dZG5mDVaZsHs4cMse/view?usp=drivesdk

THANK YOU