**Online Learning Platform using MERN**

Project Title: Learn Hub

Team ID: LTVIP2025TMID54005

Team Size: 4

Team Leader: Kokilampati Bhavani

Team member: Gudipudi Lakshmi Prasanna

Team member: Gandikota Kalyan Kumar

Team member: Jangam Sri Sreya

**INTRODUCTION**

An online learning platform (OLP) is a digital platform that provides a variety of tools and resources to facilitate learning and education over the internet. These platforms have become increasingly popular, especially in recent years, as they offer flexibility and accessibility for learners of all ages and backgrounds. Here are some key features and a description of an online learning platform:

**Certification:** Learners can earn certificates or badges upon completing courses or meeting certain criteria, which can be valuable for employment or further education.

**Accessibility:** Content is often accessible on various devices, including computers, tablets, and smartphones, making learning possible from anywhere with an internet connection.

**Self-Paced Learning:** Learners can typically access course materials at their own pace. This flexibility allows for learning that fits into individual schedules and preferences.

**Payment and Subscription Options**: There may be free courses, but some content may require payment or a subscription. Platforms often offer multiple pricing models.

**PROJECT OVERVIEW**

1. Facilitating Access to Knowledge and Resources:

Consolidated Learning Materials:

The Learning Hub aims to centralize various learning resources, making them easily accessible to users.

* Personalized Learning**:**

Features like user authentication and tailored content delivery ensure that each learner receives information relevant to their needs and preferences.

2. Fostering Collaboration and Active Learning:

* Interactive Features:

Platforms often include discussion forums, live courses, and other interactive tools to encourage learner participation and knowledge exchange.

* Community Building:

By facilitating interaction, the Learning Hub aims to build a sense of community among learners, promoting collaborative learning and peer support.

3. Enhancing the Learning Experience:

* Effective Teaching Tools:

For educators, the Hub provides tools for content creation, student monitoring, and performance assessment.

**TECHNICAL ARCHITECTURE**

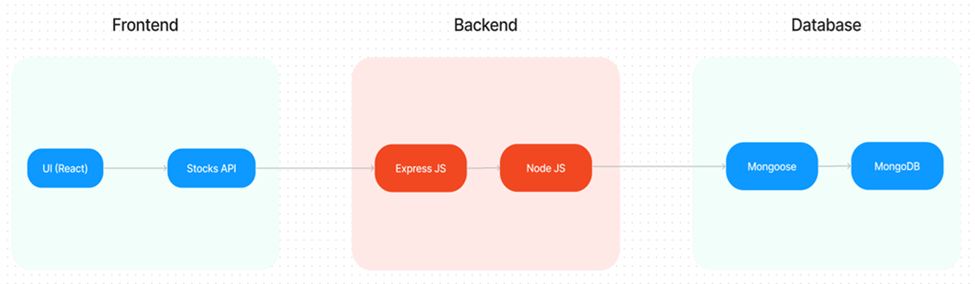
The technical architecture of OLPapp follows a client-server model, where the frontend serves as the client and the backend acts as the server. The frontend encompasses not only the user interface and presentation but also incorporates the axios library to connect with backend easily by using RESTful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

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

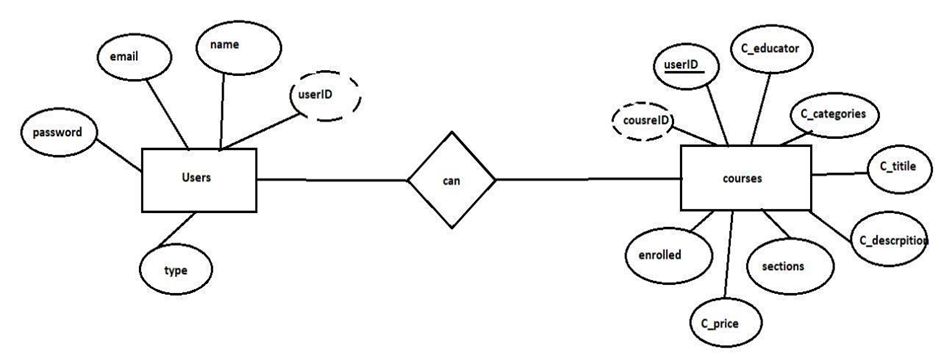
On the backend side, we employ Express.js frameworks to handle the server-side logic and communication.

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 OLPapp. This architecture enables real-time communication, efficient data exchange, and seamless integration, ensuring a smooth and immersive blogging experience for all users.

****

**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

**SETUP INSTRUCTIONS**

PRE-REQUISITES:

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/](https://www.section.io/engineering-education/nodejs-%20mongoosejs-mongodb/)

**INSTALLATION**

**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:

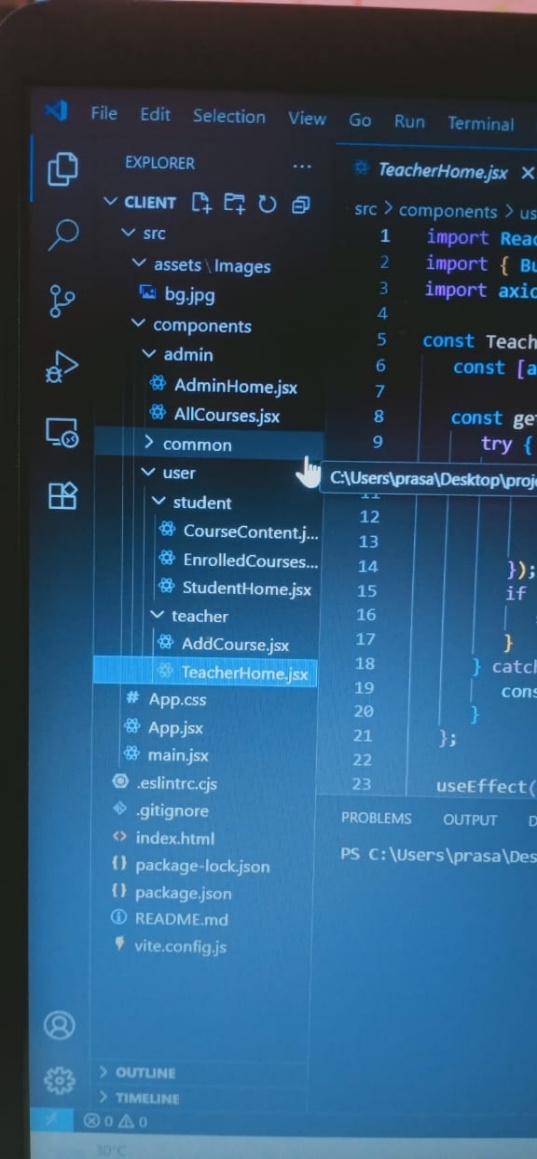
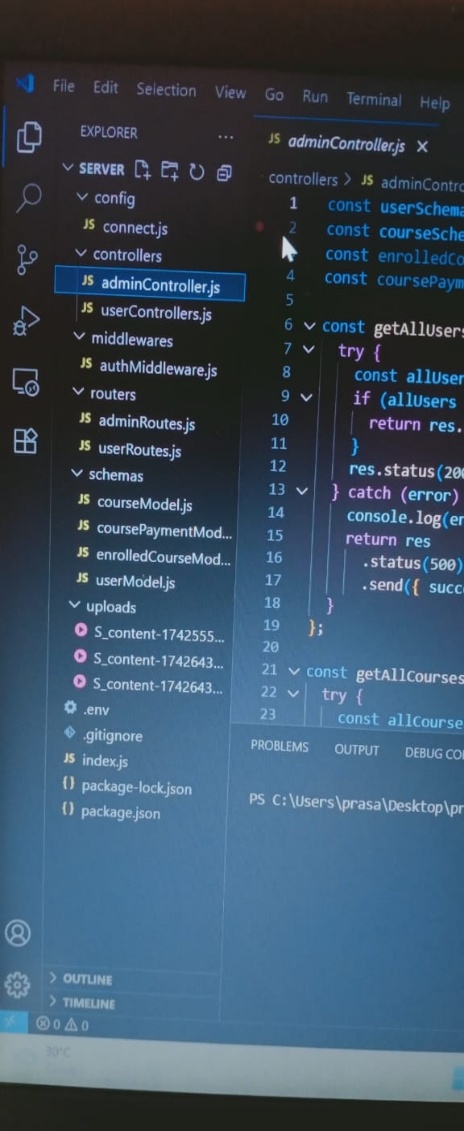
npm start

• The OLPapp will be accessible at [http://localhost:5172](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.

**PROJECT 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

** **

**Application Flow:**

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

**Teacher:**

1. Can add courses for the student.
2. Also, delete the course if no student enrolled in it or for any other reasons.
3. Also, add sections to courses.

**Student:**

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

**Admin:**

1. They can alter all the courses that are present in the app.
2. Watch out for all kinds of users in the app.
3. Record all the enrolled students that are enrolled in the course.

**RUNNING THE APPLICATION**

**Milestone 1- Setup & configuration**

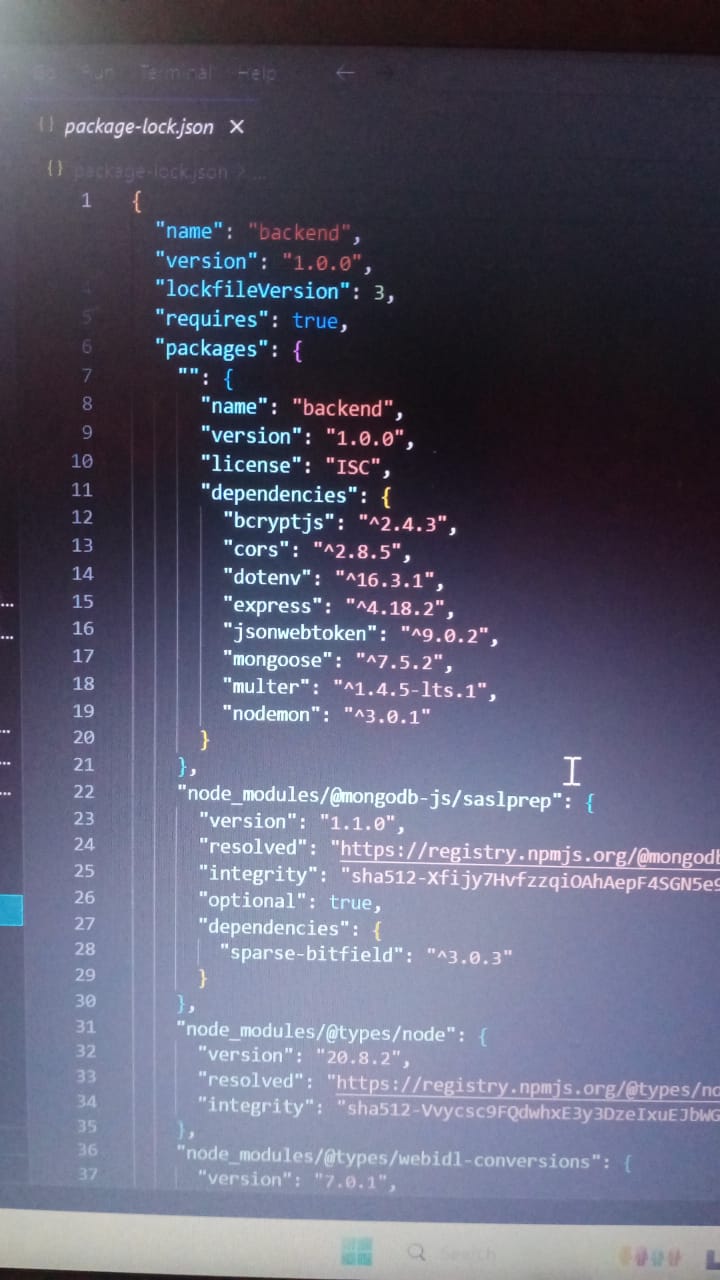
* **Folder setup:**

1. Create frontend and
2. Backend folders

. Open the backend folder to install the necessary tools

For backend, we use:

* cors
* bcryptjs
* express
* dotenv
* mongoose
* Multer
* Nodemon
* jsonwebtoken



**Milestone 2- Backend Development**

* **Setup express server**

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

* **Add authentication:**

for this,

1. You need to make a middleware folder and, in that make, authMiddleware.js file for the authentication of the projects and can use in.

**REF**: [](BACKEND%20LINK)

**Milestone 3- Database**

* **Configure MongoDB**

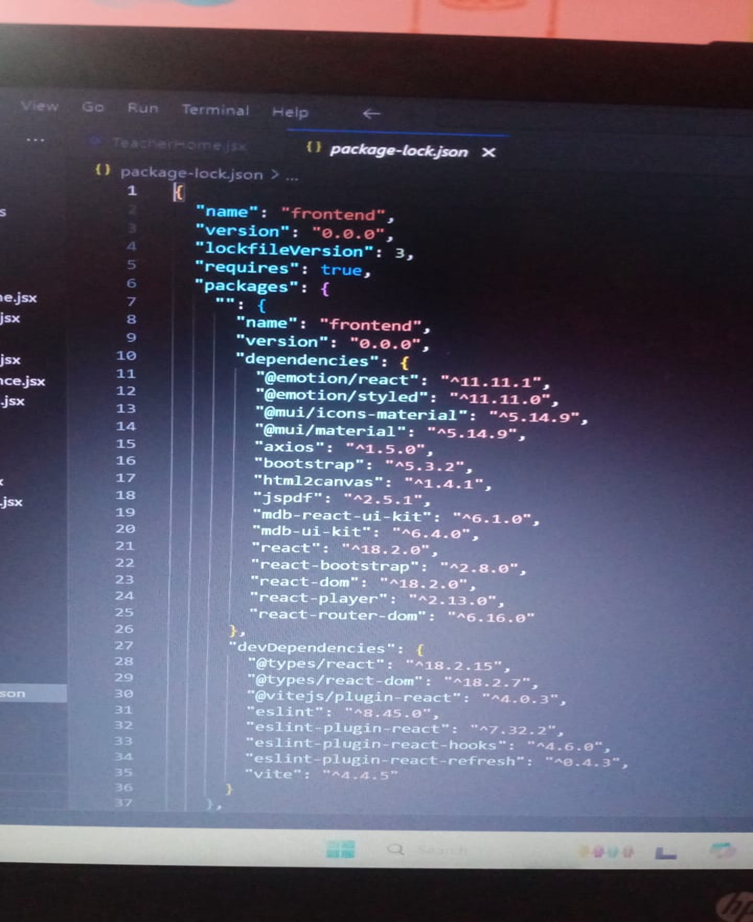
1. Import mongoose.
2. Add database connection from config.js file present in the config folder
3. Create a model folder to store all the DB schemas.

ref: [database.mp4](https://drive.google.com/file/d/17xQxIkMbVd-FORGJnEyxwwEJyl7WKbvY/view?usp=sharing)

**Milestone 4- Frontend Development**

* **Installation of required tools:**
* For frontend, we use:

1. React
2. Bootstrap
3. Material UI
4. Axios
5. Antd
6. mdb-react-ui-kit

****

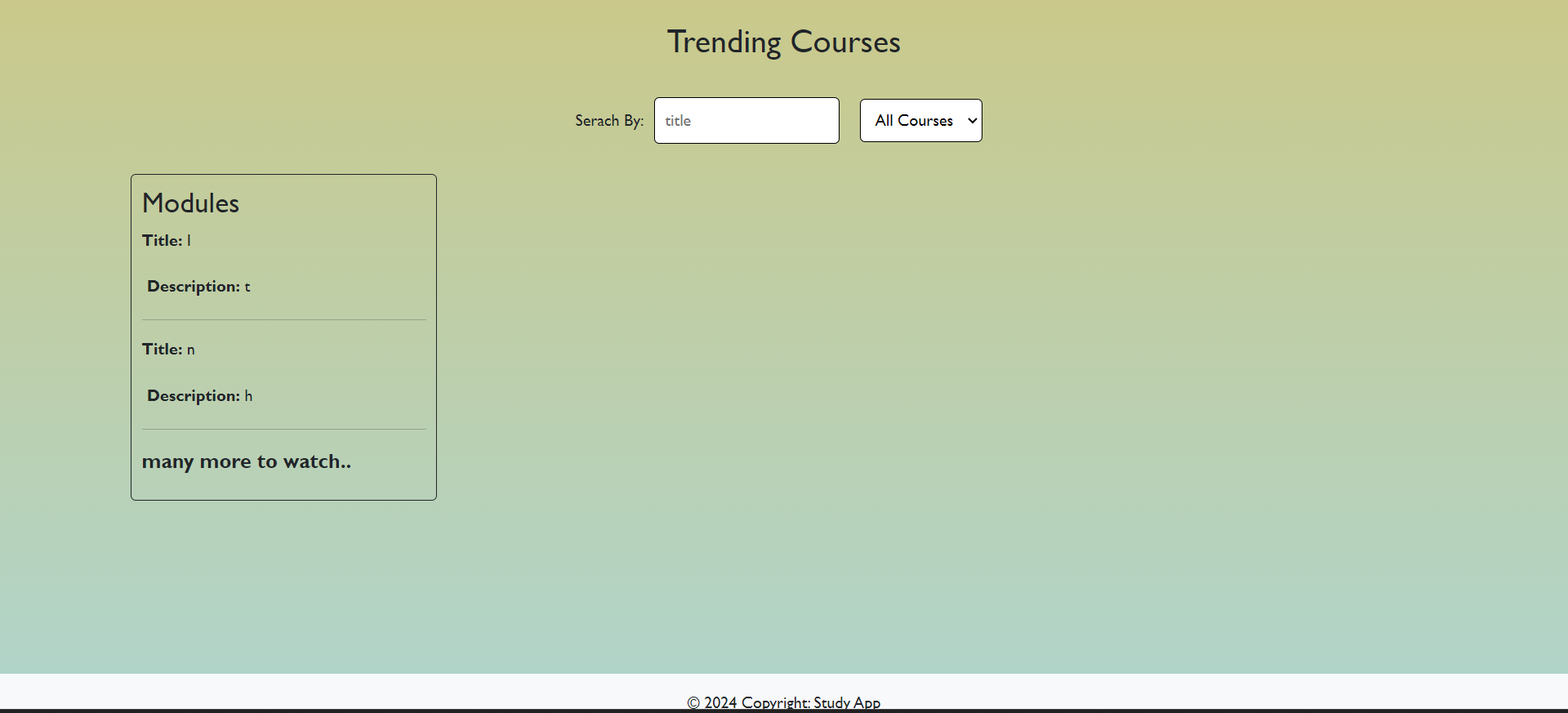
**Ref:** [](FROENT%20LINK)

**Milestone 5: Project Implementation:**

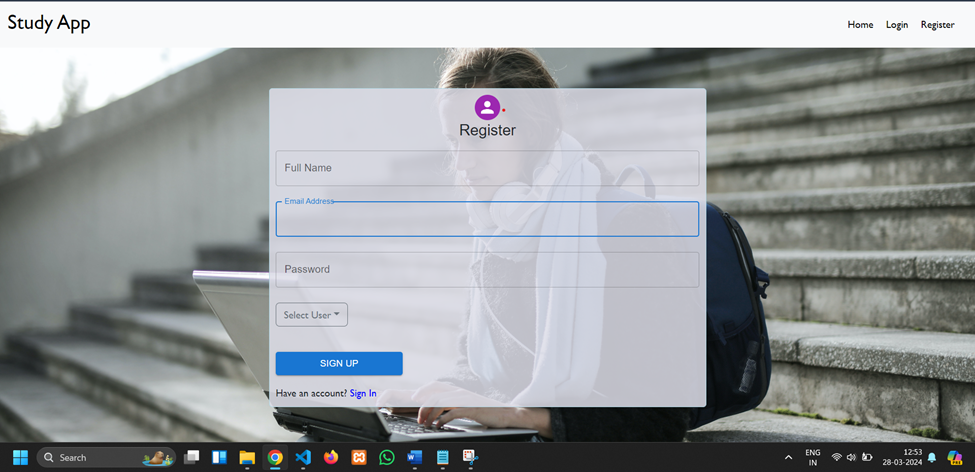
On completing the development part, we then ran the application one last time to verify all the functionalities and look for any bugs in it. The user interface of the application looks a bit like the one’s provided below.

**LANDING PAGE:**

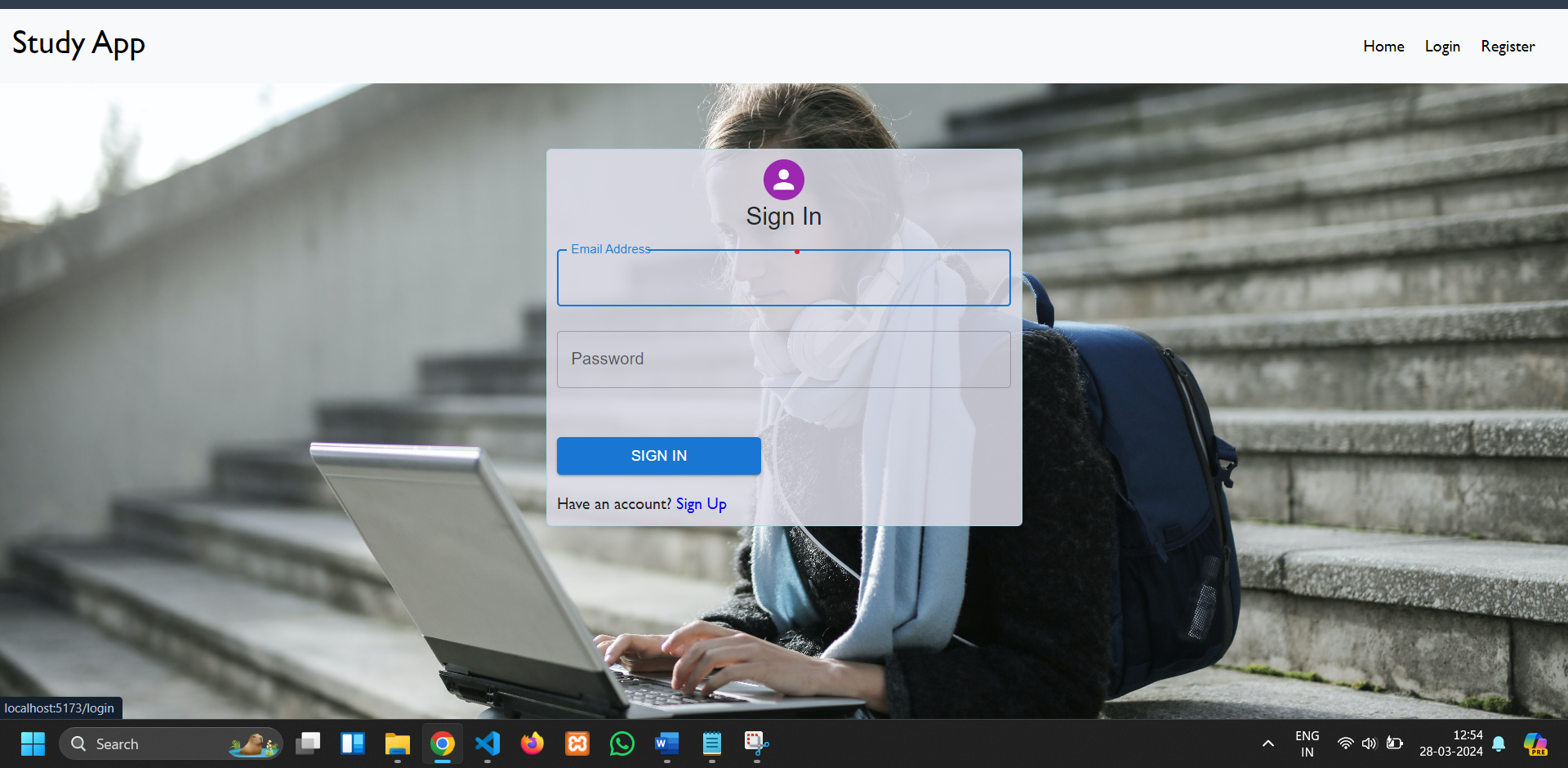




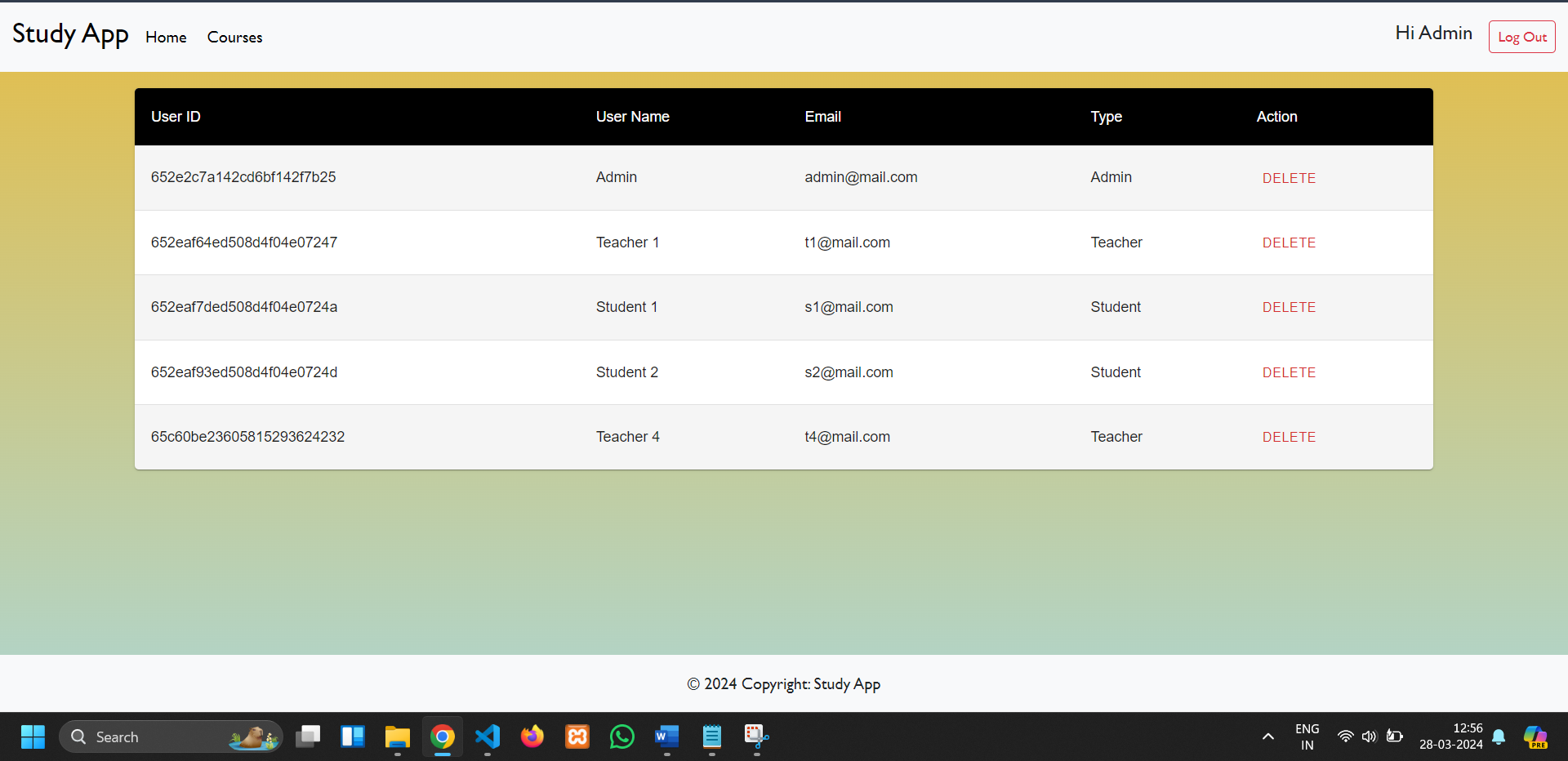
* **REGISTER PAGE:**



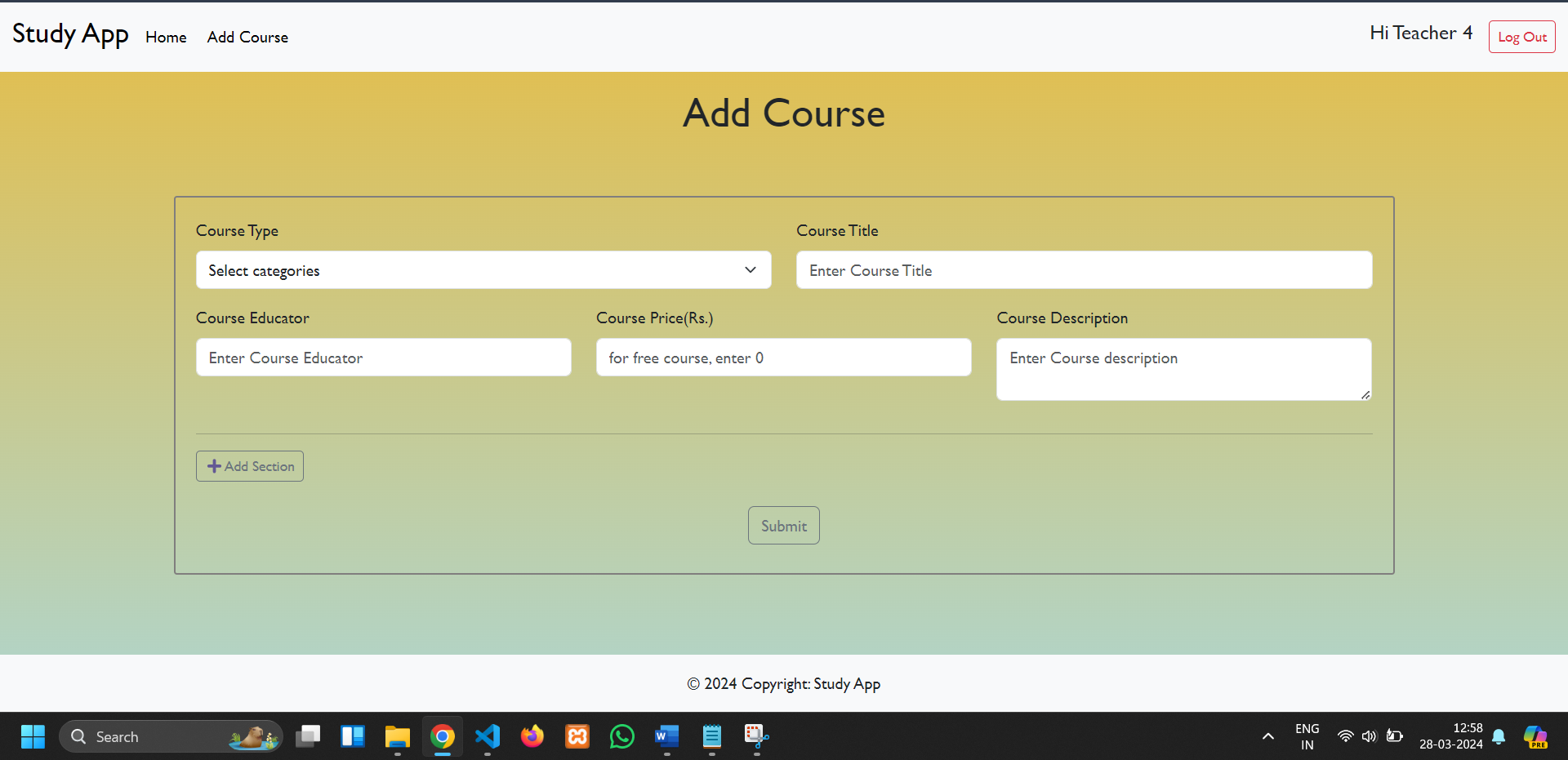
**Login page:**



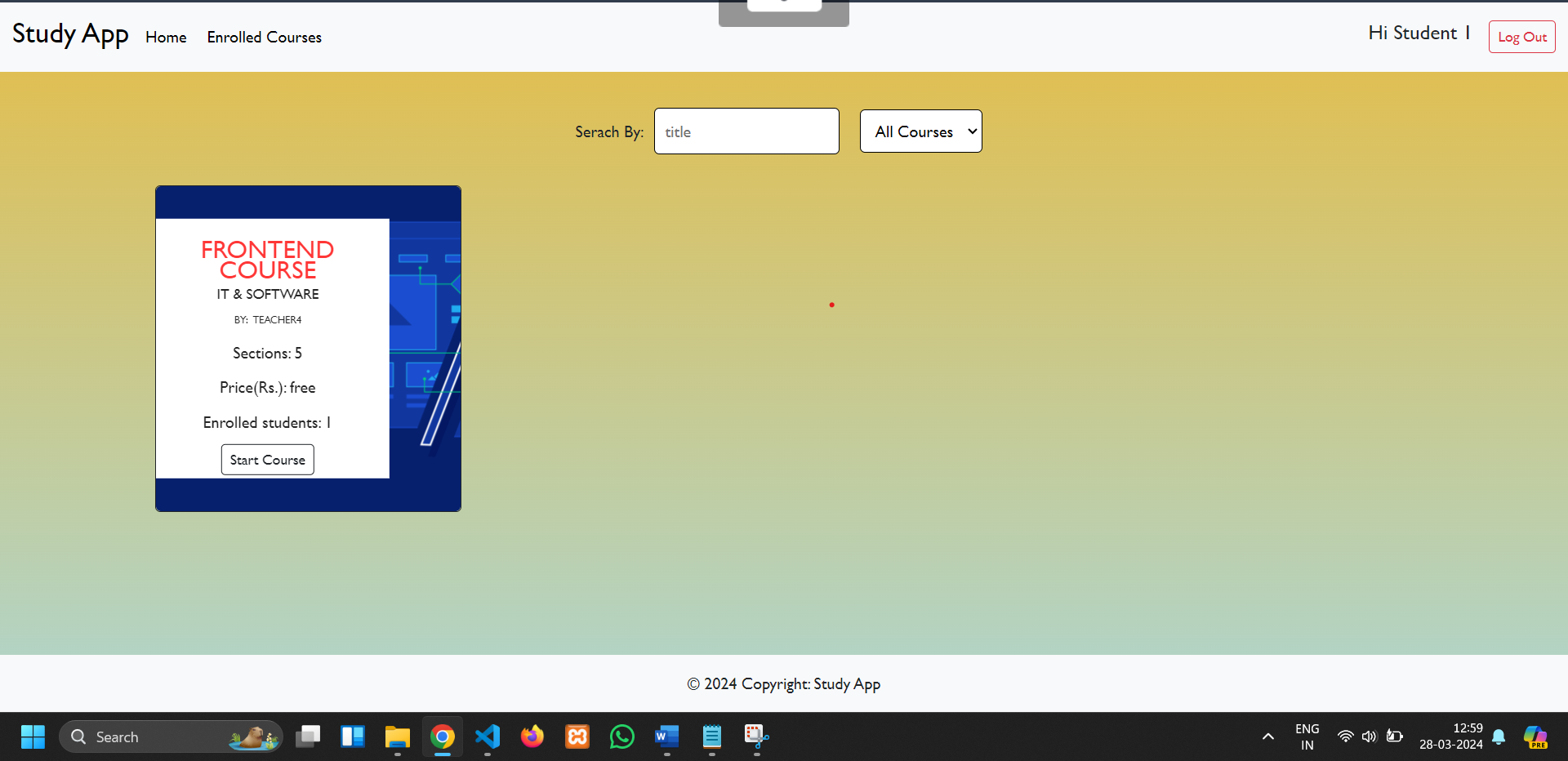
**Admin Dashboard:**

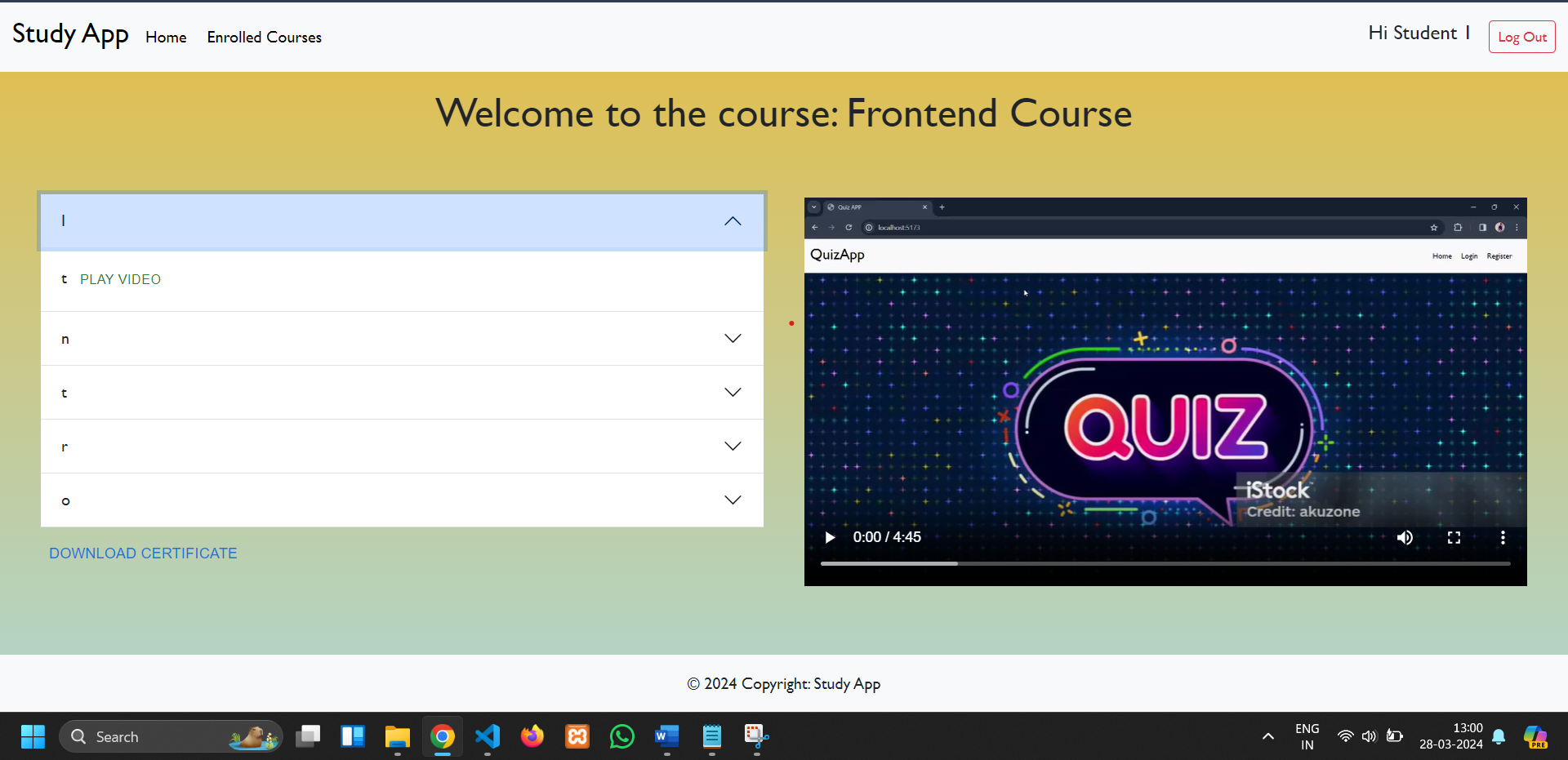


**Teacher Dashboard:**



**Student Dashboard:**





Note: For the code drive, click on link, and for the demo link, click on

[](DEMO)