**FULL STACK DEVELOPMENT**

**WITH MERN STACK**

1. **INTRODUCTION:**

* **PROJECT TITLE: “**ONLINE LEARNING PLATFORM USING MERN”
* **TEAM MEMBERS:**

1. Jayasriraman V S (Team Lead) - 310521104035
2. Koushik R - 310521104045
3. Aravind J - - 310521104012
4. **PROJECT** **OVERVIEW:**

* **PURPOSE:**

The Online Learning Platform using MERN is to provide a quality education to students. The students can explore various courses and learn multiple courses through interactive classes. In this online learning platform students can test their skills and progress by accessing the IDE, which helps the students to determine their skill levels and upon finishing the course, students will receive a certificate of completion. This online learning platform also benefits the instructors (Teachers) by becoming well-versed on their subjects. The instructors can also study courses and guide the students, so they can be knowledgeable not just in a single course, but in a variety of courses across different subjects.

* **FEATURES:**

1. ***User Authentication and Authorization:***  
   **Sign-Up / Login System**: Users (students, instructors, administrators) can register, log in, and manage their accounts.  
   **Role-Based Access Control:** Different types of users (instructors and students) have different levels of access and functionality.
2. ***Course Management:*  
   Course Creation and Editing:** Instructors can create and manage courses, including adding videos, documents and other resources. **Course Categories and Tags:** Organize courses by categories (e.g., Programming, Business, Personal Development) and tags to make it easier for students to find relevant courses.  
   **Course Completion:** Students will receive a Course Completion Certificate upon finishing the course.
3. ***Video Hosting and Streaming:*  
   Course Videos:** Integration with video streaming services or hosting for on-demand lessons, ensuring smooth playback for students.  
   **Adaptive Streaming**: Adjust video quality based on internet speed for optimal performance.
4. ***Payments and Subscriptions:*  
   Course Purchases:** Students can purchase individual courses or subscribe to a bundle of courses.  
   **Secure Payment Integration:** Integration with payment gateways (e.g., Stripe, PayPal) to process payments securely.
5. ***Student* *Dashboard:*  
   Personalized Dashboard:** A student’s homepage showing enrolled courses and recent activity.  
   **Certificates:** After course completion, students can download or print certificates.
6. ***Instructor* *Dashboard:*  
   Course Creation:** Instructors can add new courses according to the field of expertise.
7. ***Backend* *Features:***  
   **Database Management**: Use MongoDB to store user data, course content and activity logs.

* **Course Suggestion:  
   AI:** This web application has an artificial intelligence which helps the students to enrol in their desired course.
* **IDE:  
   Online Compiler:** We provide the students with an online compiler (i.e.) IDE which helps the students to test their code and skill levels on respective programming languages. It is also open to all users.
* **Resume:  
   Resume Builder:** The Online Learning Platform also contains a resume building page which helps the students to build their resume and instructors to update their old resume. It is also open to all users.

1. **ARCHITECTURE:**

**Frontend:**

**Framework**: React.js

* + **Components:** The frontend is designed using reusable React components that represent different parts of the application, such as user dashboards, project forms, chat interfaces, and notification panels.
  + **Routing:** React Router is utilized for client-side routing, allowing smooth navigation between pages such as the client dashboard, freelancer project listings, and admin panel.
  + **State Management:** The application leverages React's use State and use Context hooks for local and global state management, while more complex data handling is managed using third-party libraries like Redux (if applicable).
  + **Styling:** The UI is styled using CSS-in-JS libraries (e.g., styled-components) or traditional CSS for custom and responsive design, providing a seamless experience across devices.
  + **Security:** Prevents unauthorized access to specific pages with protected routes, verified through token-based authentication.

**Backend:**

**Framework:** Node.js with Express.js

* **Structure:** The backend is organized into controllers, routes, and middleware:
* **Controllers:** Contain the business logic for handling HTTP requests, such as creating projects, handling proposals, and managing user interactions.
* **Routes:** Define the endpoints and link them to the appropriate controller functions.
* **Middleware:** Custom middleware ensures secure user sessions, validates JSON Web Tokens (JWT), and processes errors.

○ **Services:** Service modules separate complex operations, such as payment processing or notification handling, from the main controller logic.

○ **Real-Time Communication:** Integrated using libraries like Socket.io for the chat feature, enabling real-time messaging between clients and freelancers.

**Database:**

We use MongoDB as the database for scalable and efficient data storage. MongoDB manages information on users, courses and certificates, allowing quick retrieval and easy data management. Its document-based structure is ideal for handling the variety of data types in an educational platform.

1. **SETUP INSTRUCTIONS:**

**PREREQUISITES AND INSTALLATION:**

**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/

**Front-end Framework:**

Utilize Reactjs to build the user-facing part of the application, including entering booking room, status of the booking, and user interfaces for the admin dashboard.

For making better UI we have also used some libraries like material UI and bootstrap.

**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

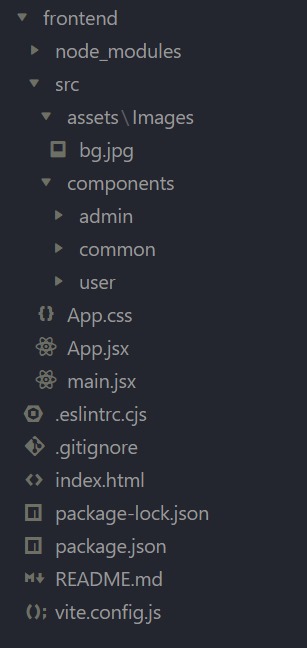
The OLP (Online Learning Platform) app will be accessible at http://localhost:5172

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.

1. **FOLDER STRUCTURE:**

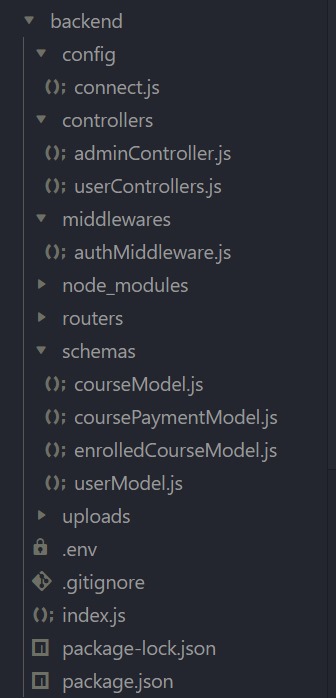
**Frontend:**

The frontend folder structure is organized to ensure modularity and maintainability. Key folders include components for reusable UI elements, pages for main application views, styles for global styling, assets for static resources (like images and icons), context for global state management, hooks for custom reusable logic, utils for helper functions, and services for API integrations. This structure promotes clean, scalable code, making it easy to develop, extend, and maintain the application’s user interface and functionality.



**Server:**

 The backend folder structure is organized to support scalability and efficient code management. Key folders include `controllers` for handling request logic, `models` for database schemas, `routes` for defining API endpoints, `middleware` for reusable request handlers (such as authentication), `services` for core business logic, `utils` for helper functions, and `config` for environment-specific settings. This structure ensures clear separation of responsibilities, making the codebase easier to understand, extend, and maintain as the application grows.



1. **AUTHENTICATION:**
   1. **User Registration and Login:**

When a user registers, their password is securely hashed using bcrypt before being saved to the database, ensuring password security. Upon login, the user provides their email and password, which is then validated. If correct JWT token is generated and signed with a secret key. This token contains an encrypted payload with the user's ID and has a set expiration time.

* 1. **Token-Based Authentication:**

Once logged in, the generated JWT is returned to the client. This token is included in the Authorization header of subsequent requests to access protected resources.

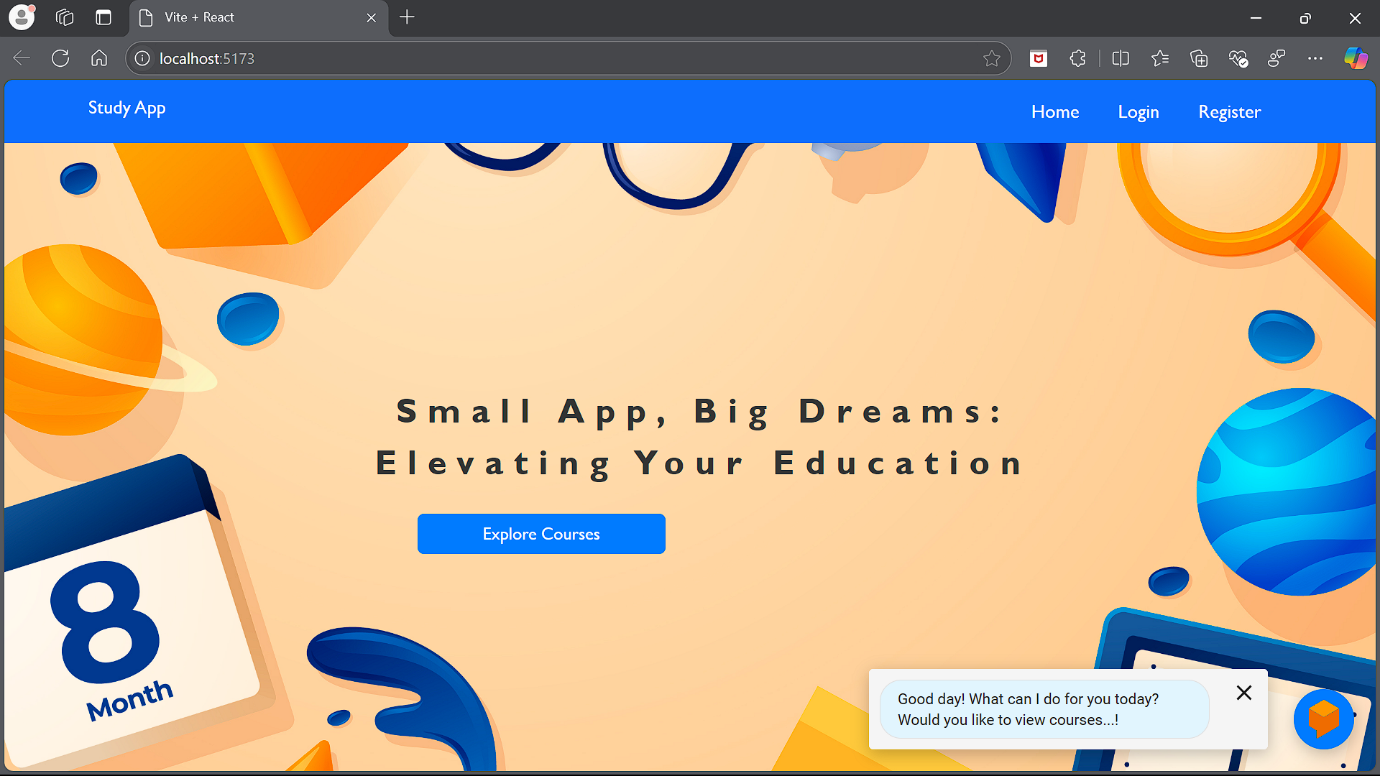
* 1. **Authorization:**

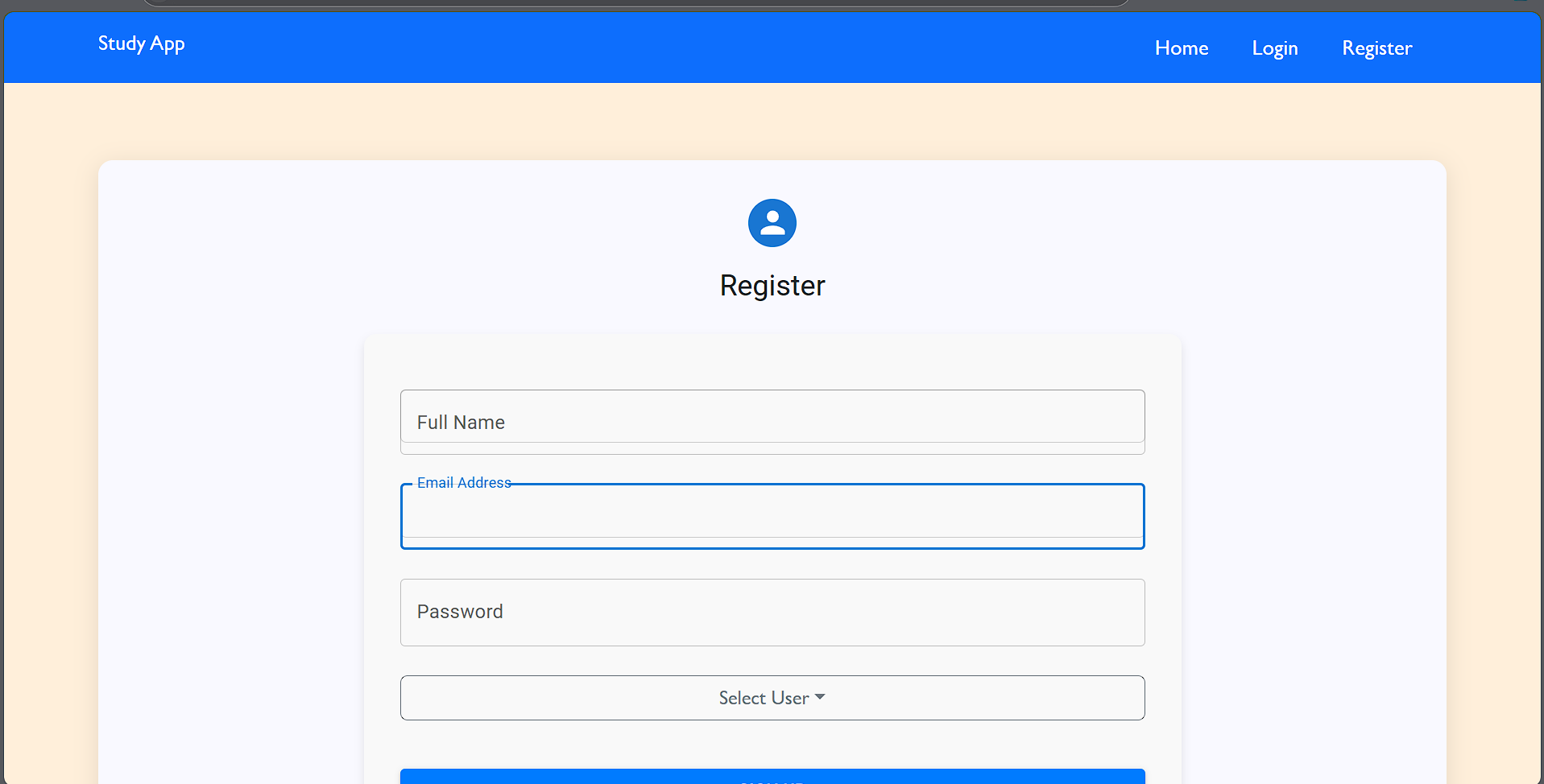
Authorization ensures that only users with the correct permissions can access specific resources. Based on the token’s payload, different user roles or permissions are checked before allowing access to resources like course management or enrolment.

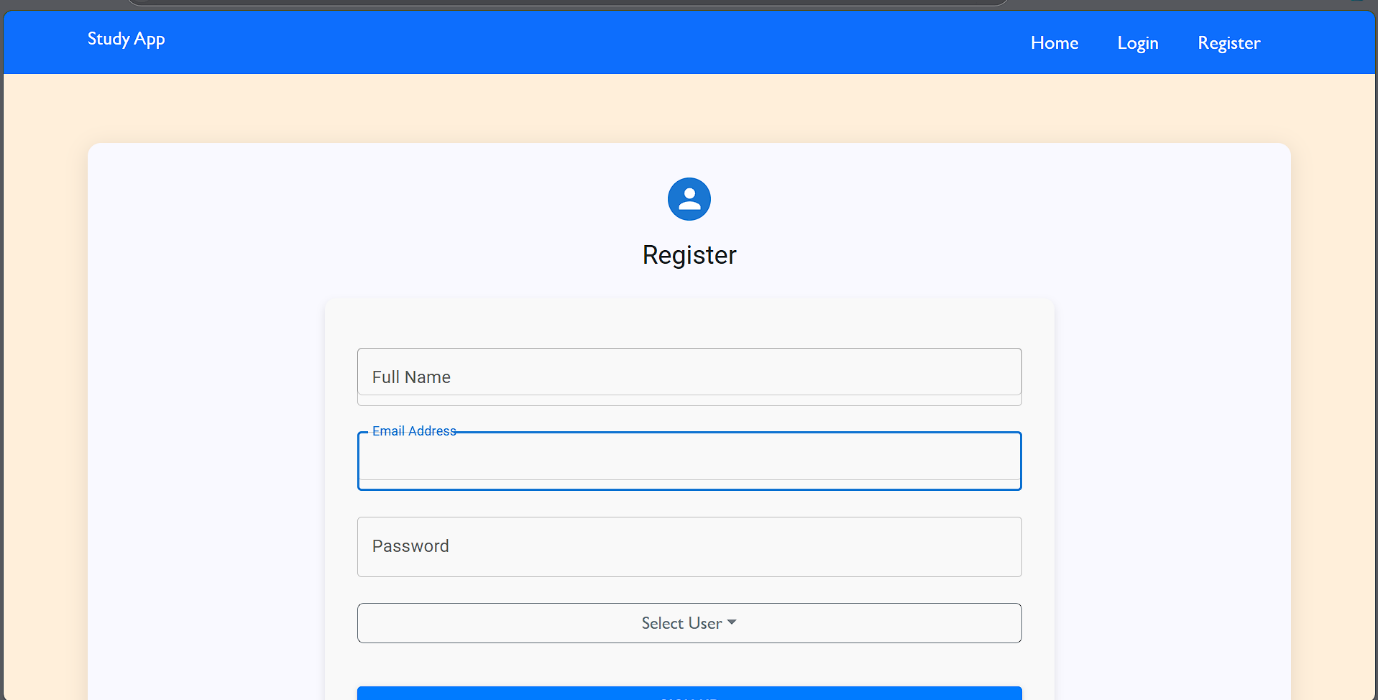
* 1. **Token Expiration and Security:**

Tokens are set to expire after a certain period to enhance security. If a token is expired, the user must log in again to receive a new token, ensuring that only active sessions can access sensitive data.

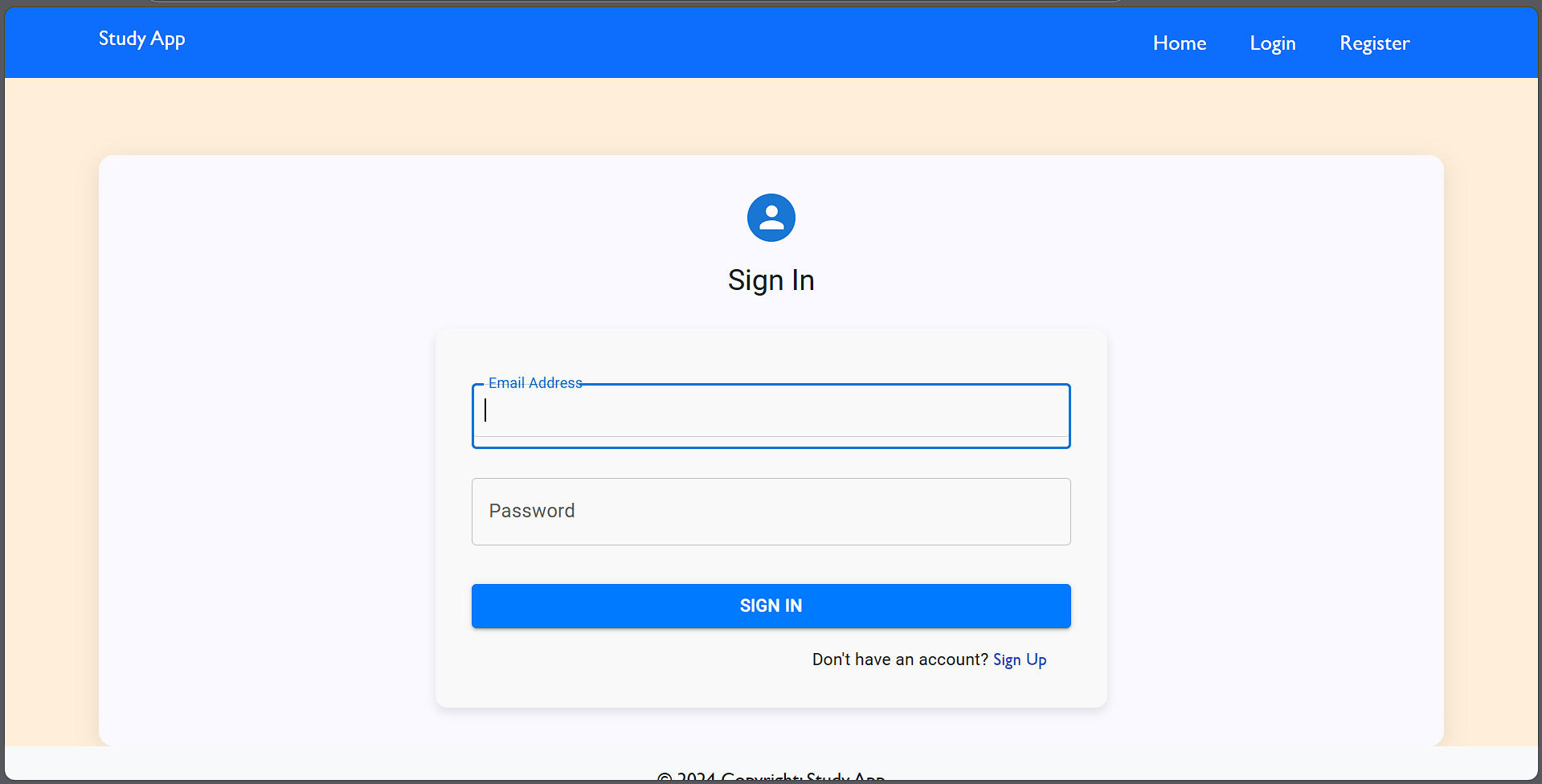
1. **USER INTERFACE:**

* **Landing Page:**
* **Register Page:**

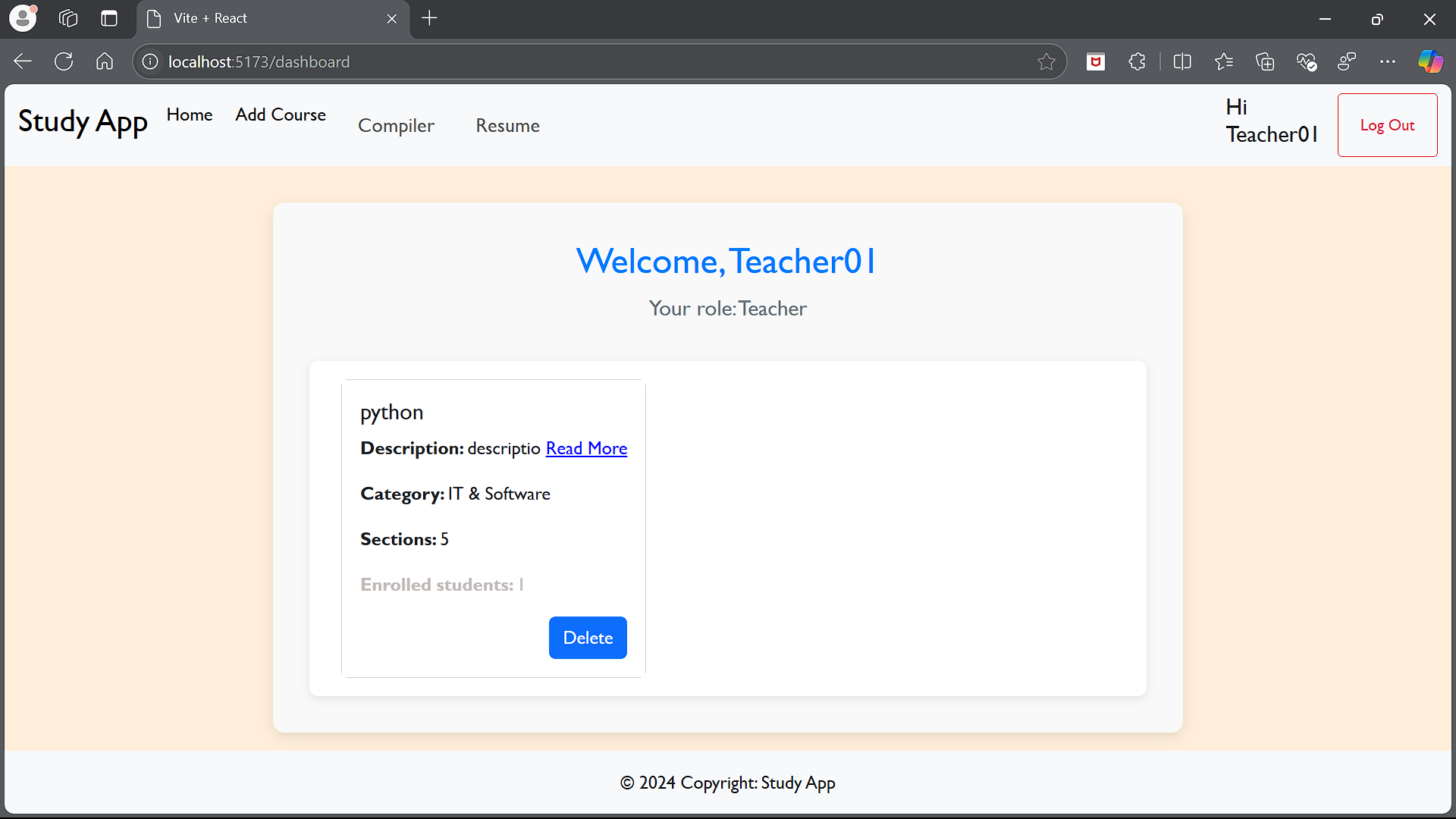




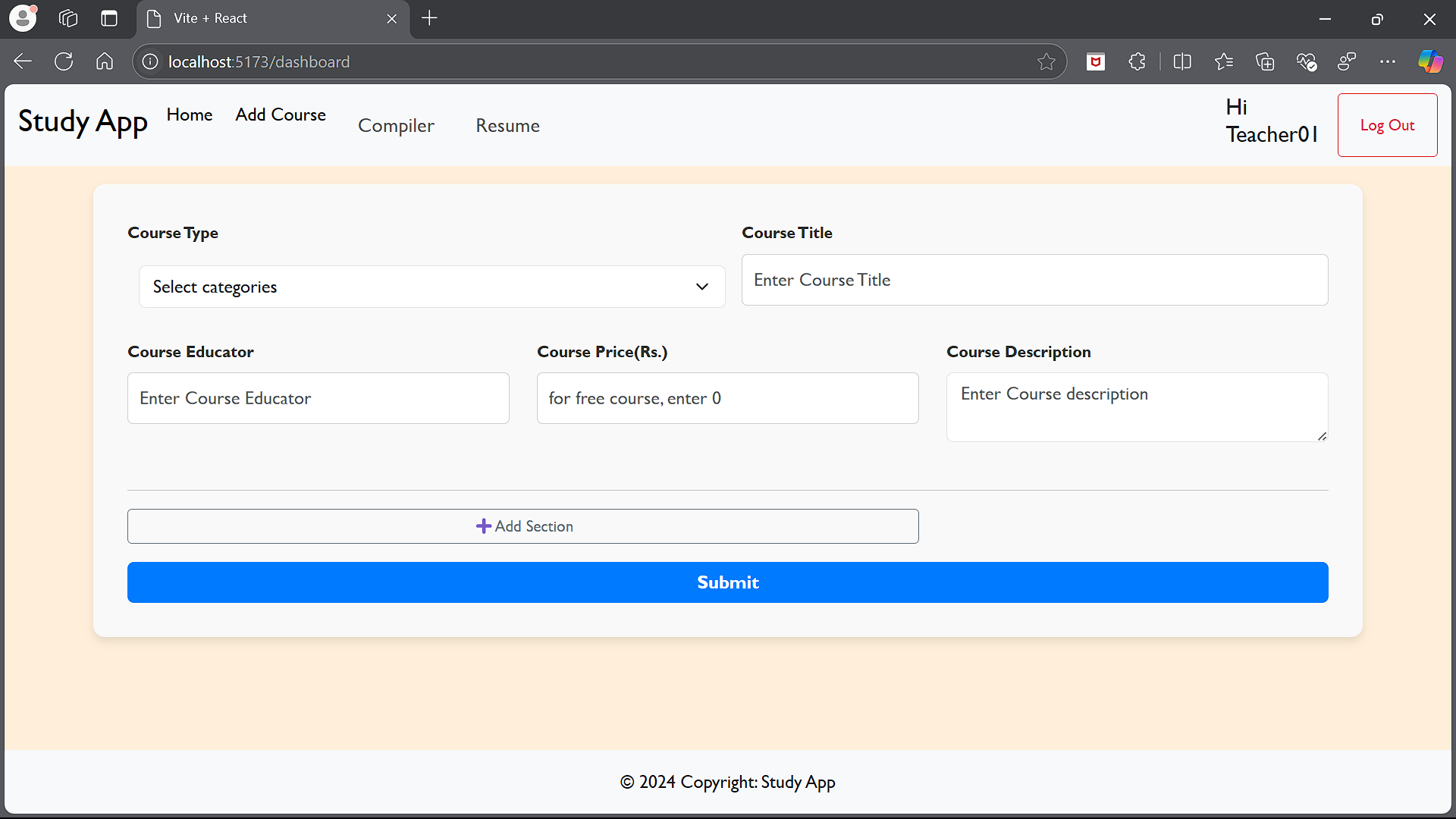
* **Login Page:**



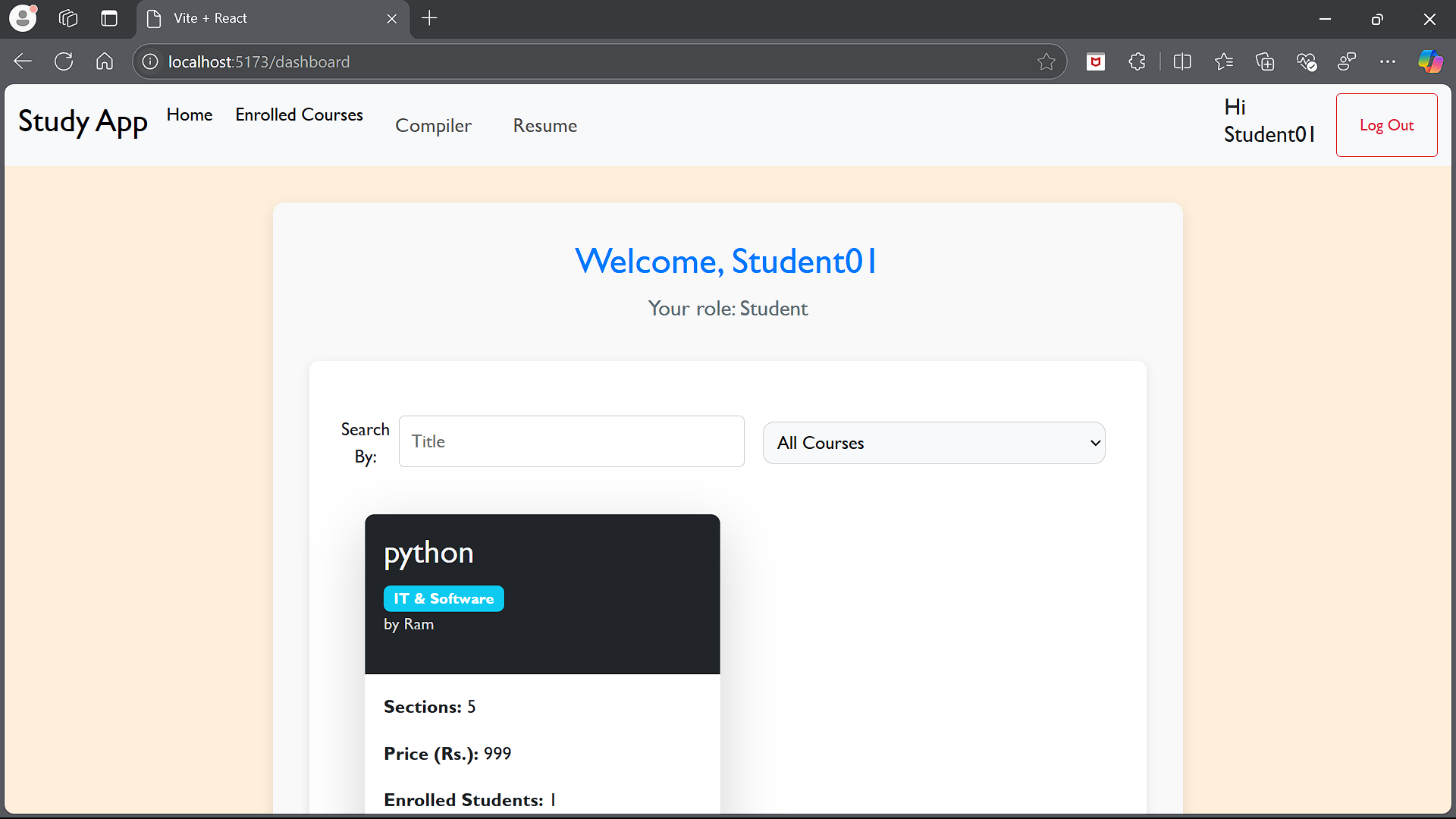
* **Teacher Dashboard:**



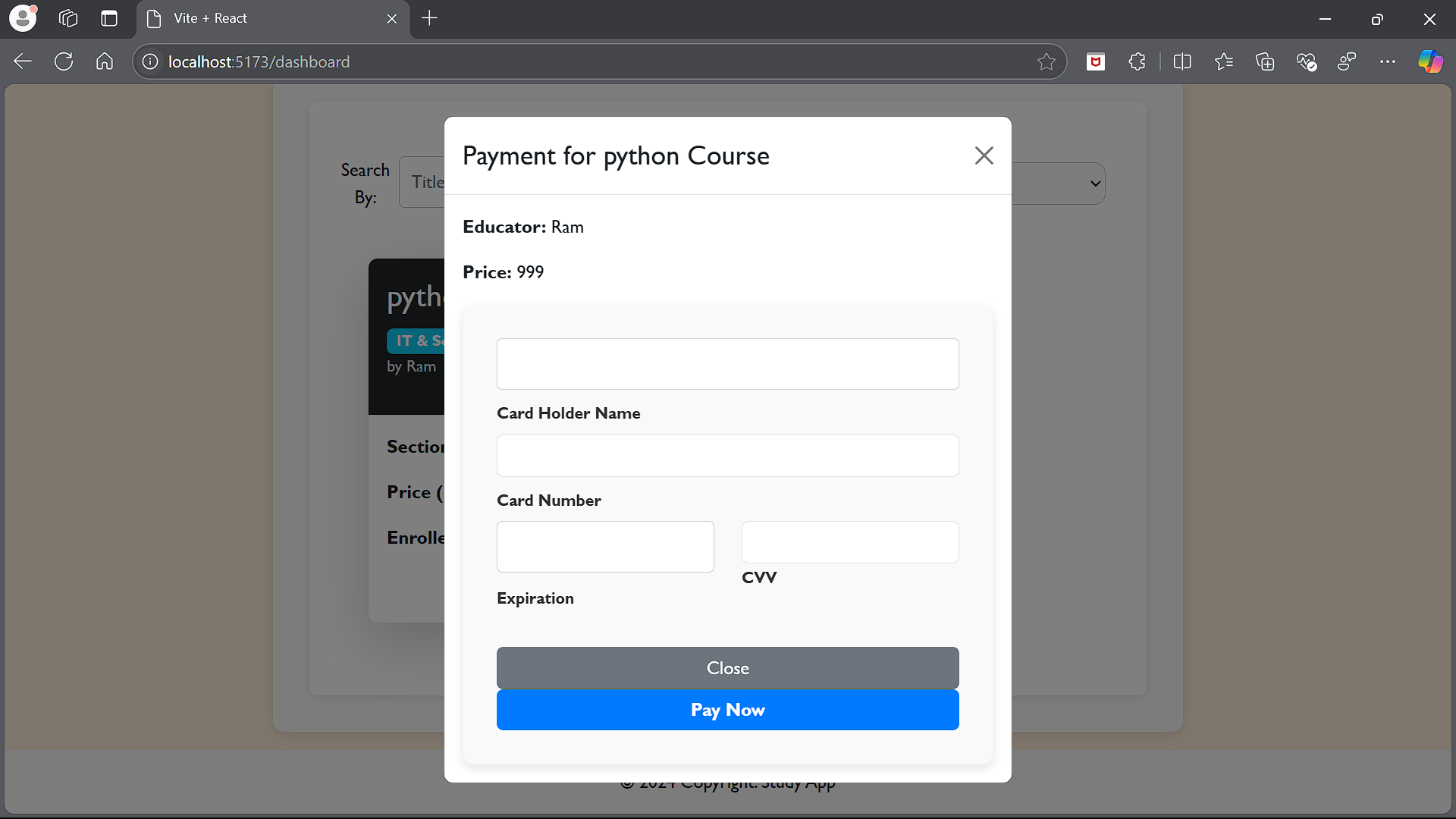
* **Add Course:**



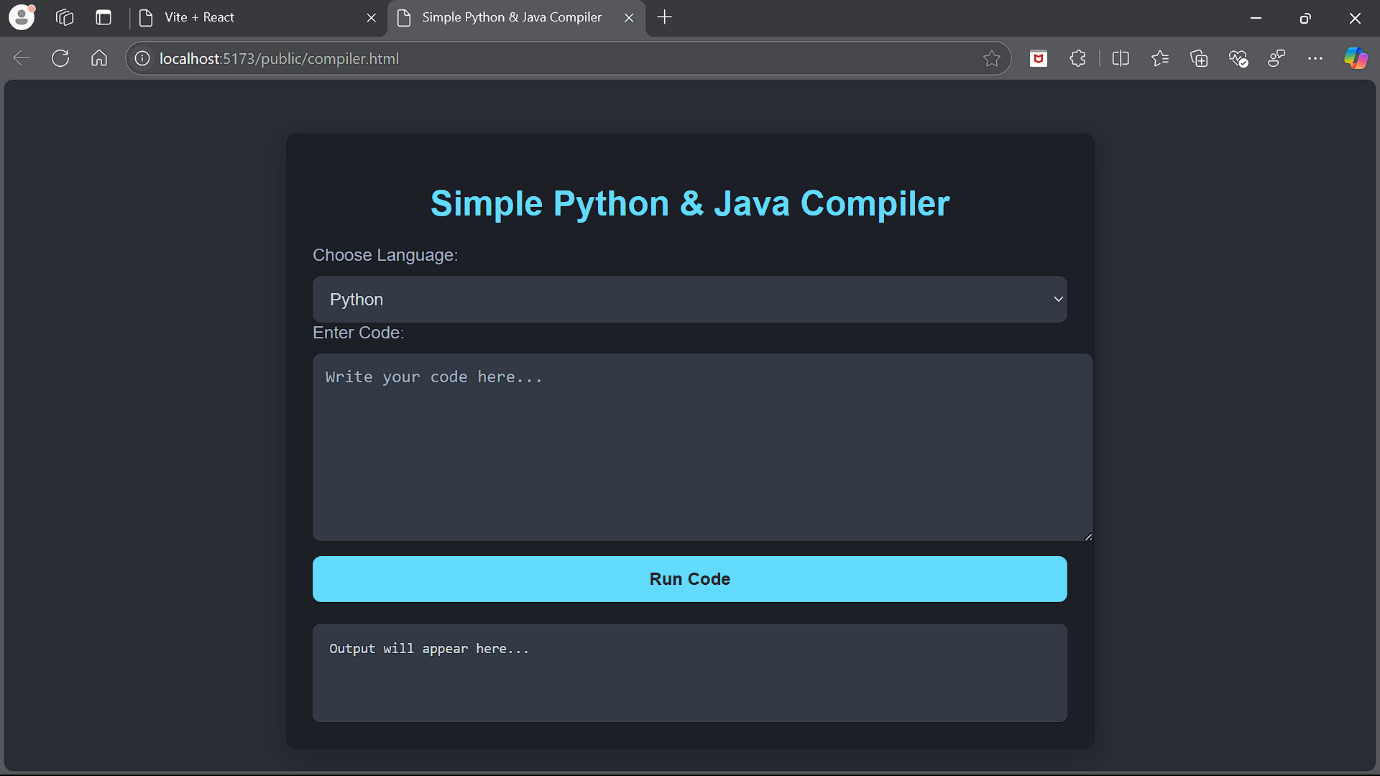
* **Student Dashboard:**



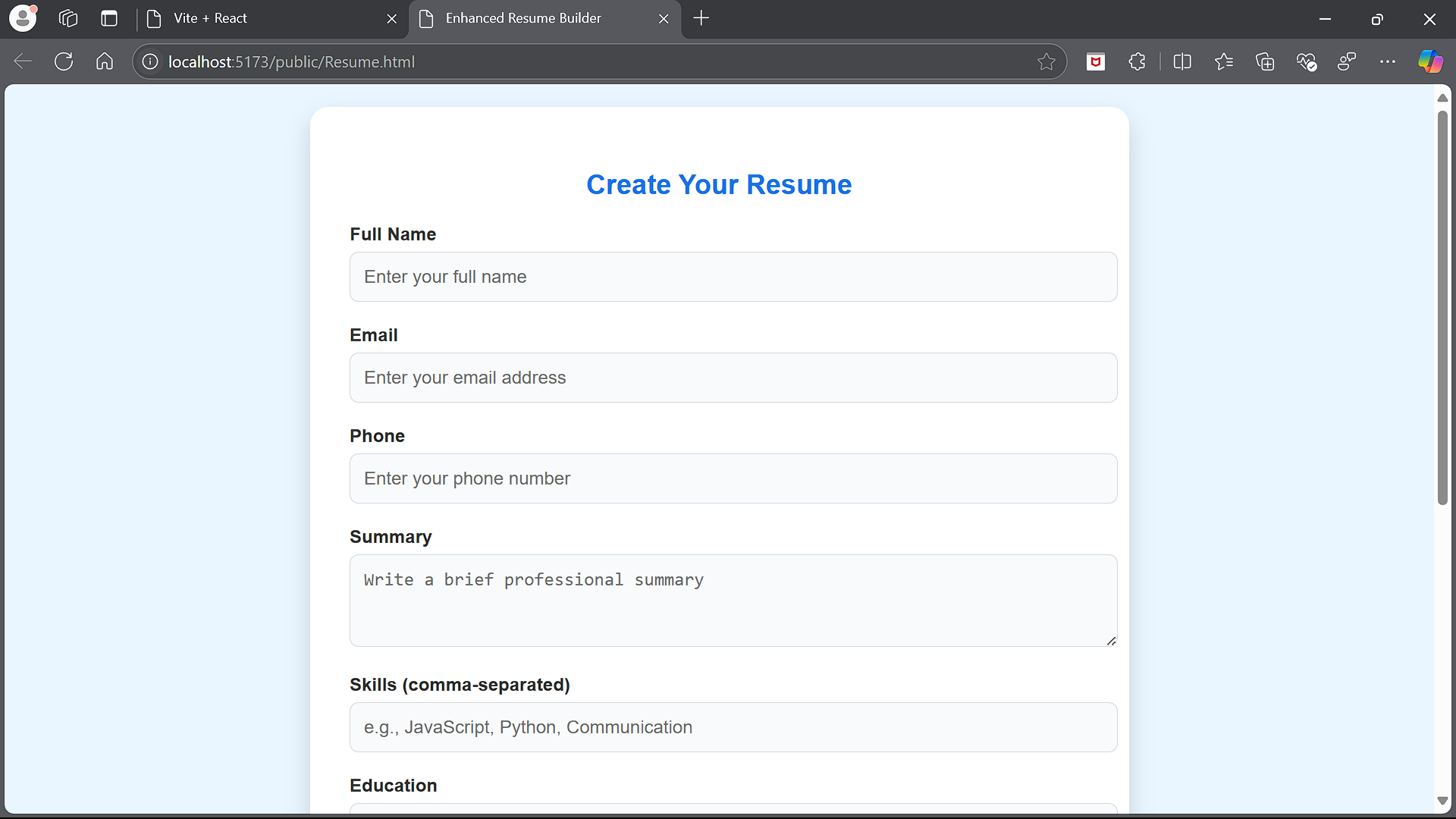
* **Course Enrolment:**



* **Online Compiler:**

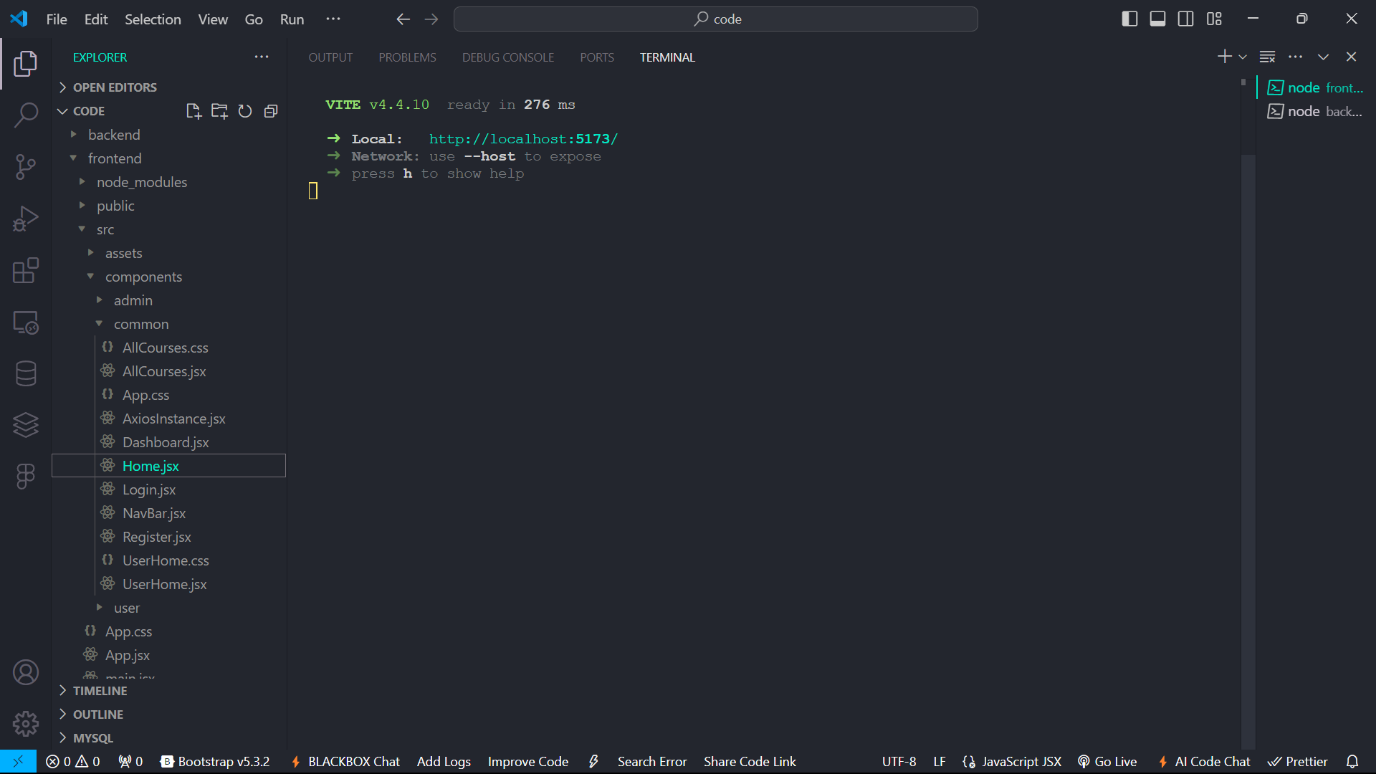


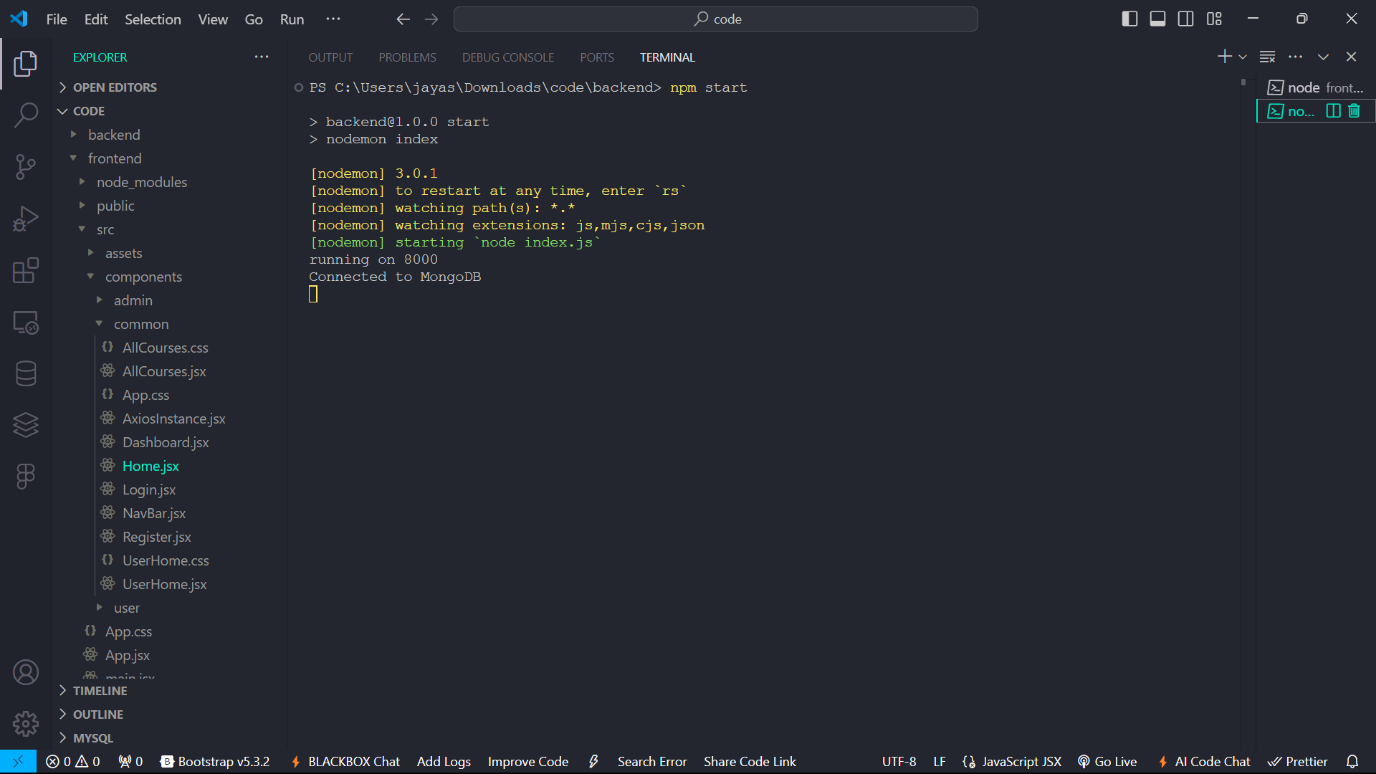
* **Online Resume:**

****

**TESTING:**

* **Frontend Testing:**



* **Frontend Testing:**
* ****

1. **SCREENSHOTS & DEMO:**

Drive Link: <https://drive.google.com/file/d/1wRBdRgKPI-39NIJXQDV6isE5NjFvpRF5/view?usp=sharing>

1. **KNOWN ISSUES:**
2. **FURTHER ENHANCEMENTS:**