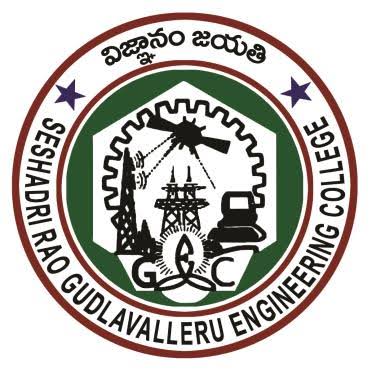
**Internship report**

****

**Seshadri Rao Gudlavalleru Engineering college , Gudlavalleru**

Submitting to : SmartInternz

Submitted by :

TeamID:- LTVIP2025TMID55604

Team :-

Potluri Pujitha

Polukonda Jenith Rahul

Polimetla Lokesh

Podugu Bala Naga Abhiram

**Online Learning Platform using MERN(Learn Hub)**

**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:

**User-Friendly Interface:** Online learning platforms typically have an intuitive and user-friendly interface that makes it easy for learners, regardless of their technical proficiency, to navigate and access the content.

**Course Management:** Instructors or course creators can upload, organize, and manage course materials. Learners can enroll in courses and track their progress.

**Interactivity:** Many platforms include interactive elements like discussion forums, chat rooms, and live webinars, which foster communication and collaboration among learners and instructors.

**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 :**

### Purpose of the Project – LearnHub

The purpose of developing LearnHub is to create a modern, user-friendly online learning platform that bridges the gap between educators and learners. In today’s digital age, access to quality education should not be limited by geography, time, or financial resources. LearnHub empowers:

* Teachers to share their expertise by creating and managing courses,
* Students to explore and enroll in diverse subjects at their own pace, and
* Admins to ensure the smooth functioning of the platform with effective user and content management.

The project also serves as a practical implementation of MERN stack development (MongoDB, Express, React, Node.js), showcasing real-world skills like full-stack architecture, secure authentication, RESTful APIs, responsive UI design, and state management.

Ultimately, LearnHub demonstrates how technology can be used to democratize education and build scalable systems for collaborative learning.

### **Key Features of LearnHub :**

### **For Teachers:**

* Course Creation: Teachers can create and publish detailed courses including title, category, description, price, and sections.
* Section Management: Add, edit, or delete sections/lectures within each course.
* Course Deletion: Remove courses that have no active enrollments or are no longer needed.

#### **For Students:**

* Course Enrollment: Browse and enroll in both free and paid courses.
* Resume Learning: Continue courses from where they left off.
* Certificate Download: Automatically generate and download a certificate upon course completion.
* Search & Filter: Search courses by name, category, or educator for quick access.
* Payment Integration (Simulated): Students must “purchase” paid courses before starting them.

#### **For Admin:**

* User Management: View and monitor all registered users (students and teachers).
* Course Oversight: Access and manage all available courses.
* Enrollment Records: Track which students are enrolled in which courses.

#### **Platform-Wide Features:**

* Role-Based Authentication: Secure login system with different privileges for admin, teacher, and student roles.
* Responsive User Interface: Mobile-friendly and responsive design using React, Material UI, and Bootstrap.
* RESTful APIs: Clean backend architecture for handling authentication, course management, and enrollment.
* Database Connectivity: Efficient use of MongoDB and Mongoose for storing user and course data.

**Scenario-based Case Study:**

**Scenario**: Learning a New Skill

**User Registration:** Sarah, a student interested in learning web development, visits the Online Learning Platform and creates an account. She provides her email and chooses a password.

**Browsing Courses:** Upon logging in, Sarah is greeted with a user-friendly interface displaying various courses categorized by topic, difficulty level, and popularity.

She navigates through the course catalog, filtering courses by name and category until she finds a "Web Development Fundamentals" course that interests her.

**Enrolling in a Course:** Sarah clicks on the course and reads the course description, instructor details, and syllabus. Impressed, she decided to enroll in the course.

After enrolling, Sarah can access the course materials, including video lectures, reading materials, and assignments.

**Learning Progress:** Sarah starts the course and proceeds through the modules at her own pace. The platform remembers her progress, allowing her to pick up where she left off if she needs to take a break.

**Interaction and Support:** Throughout the course, Sarah engages with interactive elements such as discussion forums and live webinars where she can ask questions and interact with the instructor and other learners.

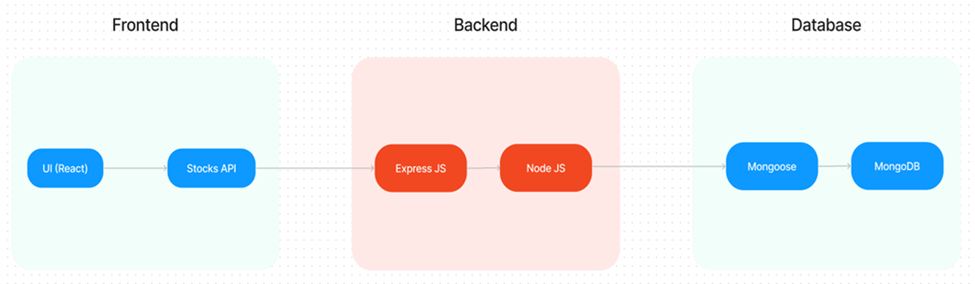
**Course Completion and Certification:** After completing all the modules and assignments, Sarah takes the final exam. Upon passing, she receives a digital certificate of completion, which she can download and add to her portfolio.

**Paid Courses:** Sarah discovers an advanced web development course that requires payment. She purchases the course using the platform's payment system and gains access to premium content.

**Teacher's Role:** Meanwhile, John, an experienced web developer, serves as a teacher on the platform. He creates and uploads new courses on advanced web development topics, adds sections to existing courses, and monitors course enrollments.

**Admin Oversight:** The admin oversees the entire platform, monitoring user activity, managing course listings, and ensuring smooth operation. They keep track of enrolled students, handle any issues that arise, and maintain the integrity of the platform.

**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.

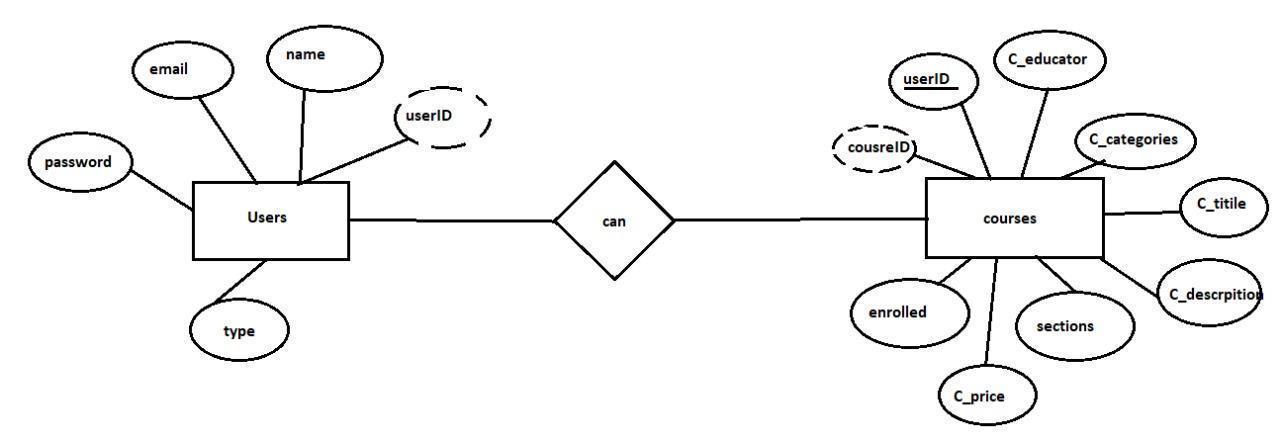
The front end utilizes the bootstrap and material UI library to establish a real-time and better UI experience for any user.

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 AND INSTALLATION :**

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/)

**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

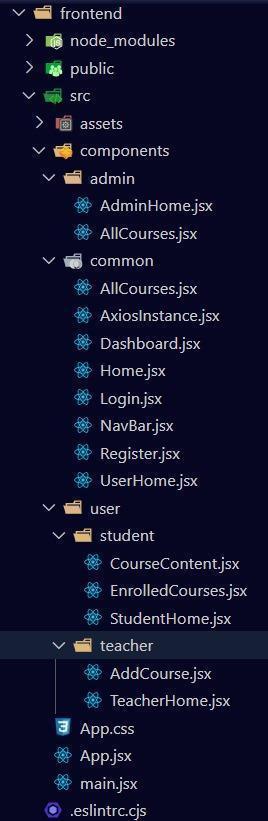
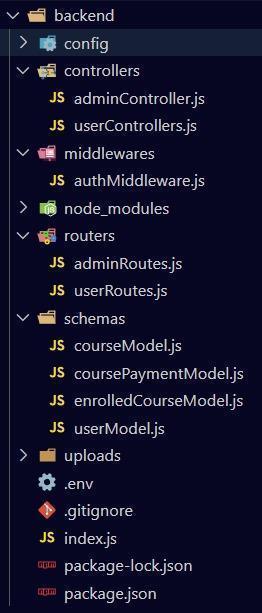
• The OLPapp will be accessible at [http://localhost:5172](http://localhost:5172/)

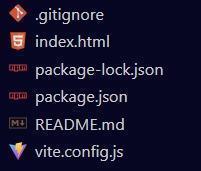
You have successfully installed and set up the Online learning app on your local machine. You can now proceed with further customization, development, and testing as needed.

**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.

**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.mp4**](https://drive.google.com/file/d/1SGqtbYtumm5JJOzldfDXhRrgSJ4AQ62L/view?usp=sharing)

**Milestone 3- Database**

* **Configure MongoDB:**

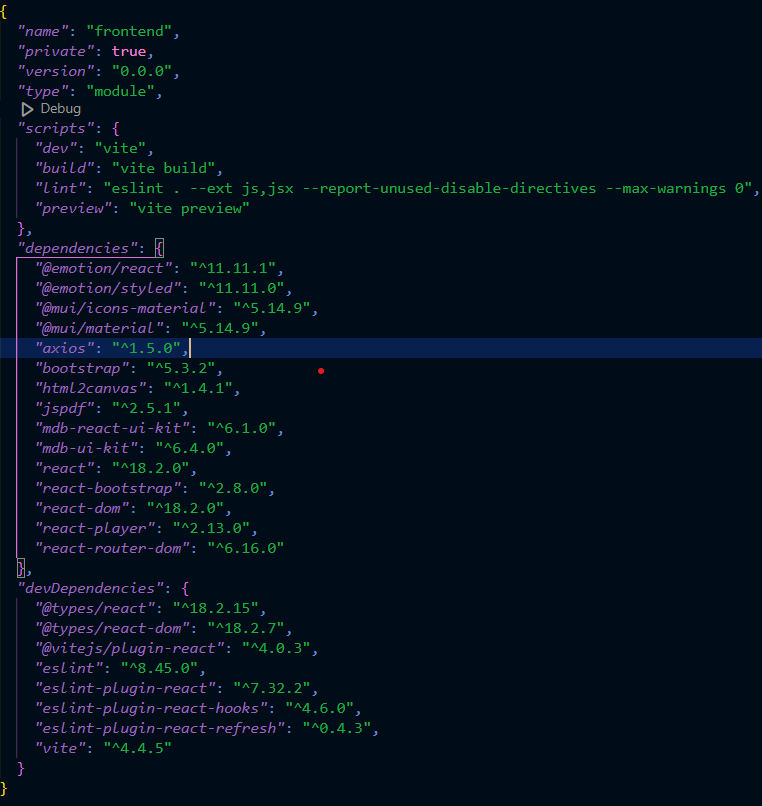
1. Import mongoose.
2. Add database connection from config.js file present in the config folder
3. Create a model folder to store all the DB schemas.

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
7. react-bootstrap



**ref:** [**frontend.mp4**](https://drive.google.com/file/d/11l3d59ibUNcoKrxigtGRH9ChCqzcSnMc/view?usp=sharing)

### **Sample Response (Course Fetch)**

**GET /api/courses**

**[**

**{**

**"\_id": "62d...",**

**"C\_title": "React for Beginners",**

**"C\_description": "Introductory course on React",**

**"C\_price": 0,**

**"C\_categories": "Frontend",**

**"C\_educator": "AbhiRam"**

**}**

**]**

**Authentication:** LearnHub uses JWT (JSON Web Token) for secure user authentication and role-based authorization.

Process:

* Register (/api/register): Users sign up with name, email, password, and role (student/teacher).
* Login (/api/login): On success, a JWT token is issued.
* Token Use: Token must be sent in the Authorization header for protected routes
* Authorization: Bearer <JWT\_TOKEN>

#### **Role-Based Access:**

1. Student: Enroll and access courses
2. Teacher: Create and manage courses
3. Admin: Manage all users and courses

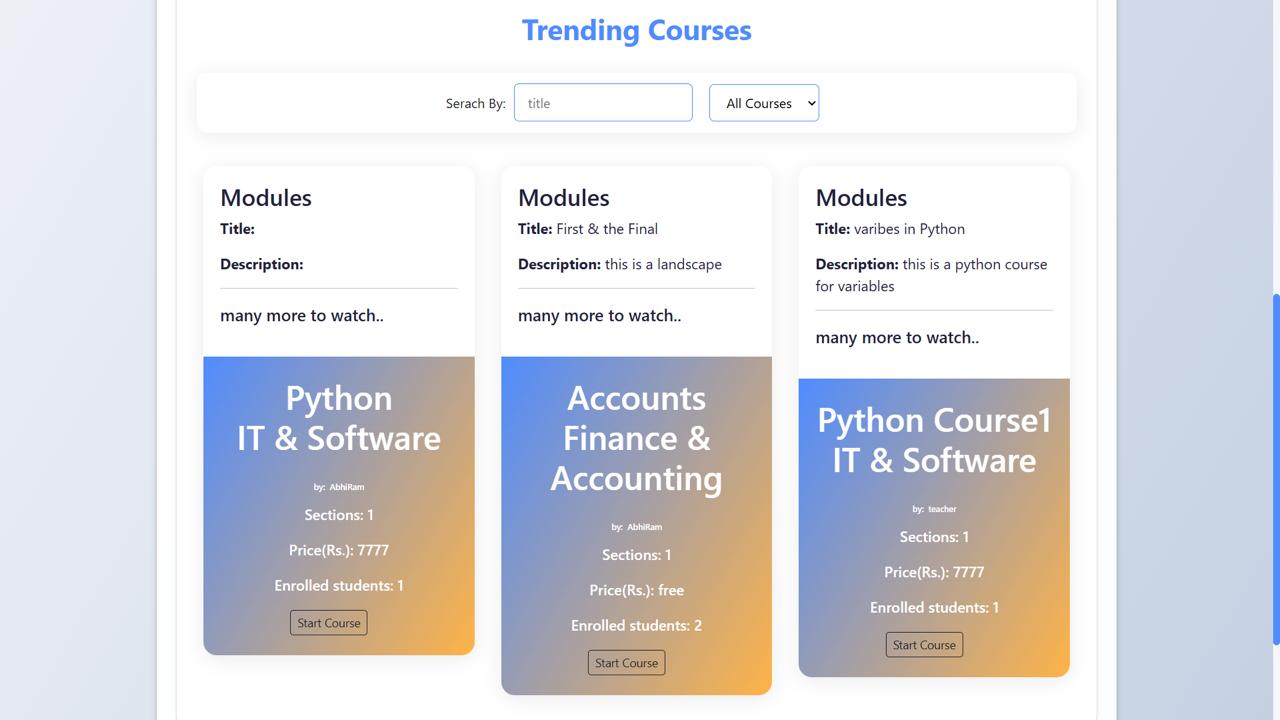
#### **Security:**

* Passwords are hashed using bcrypt
* JWT includes user ID and role, with expiry
* Middleware verifies token and controls access based on role

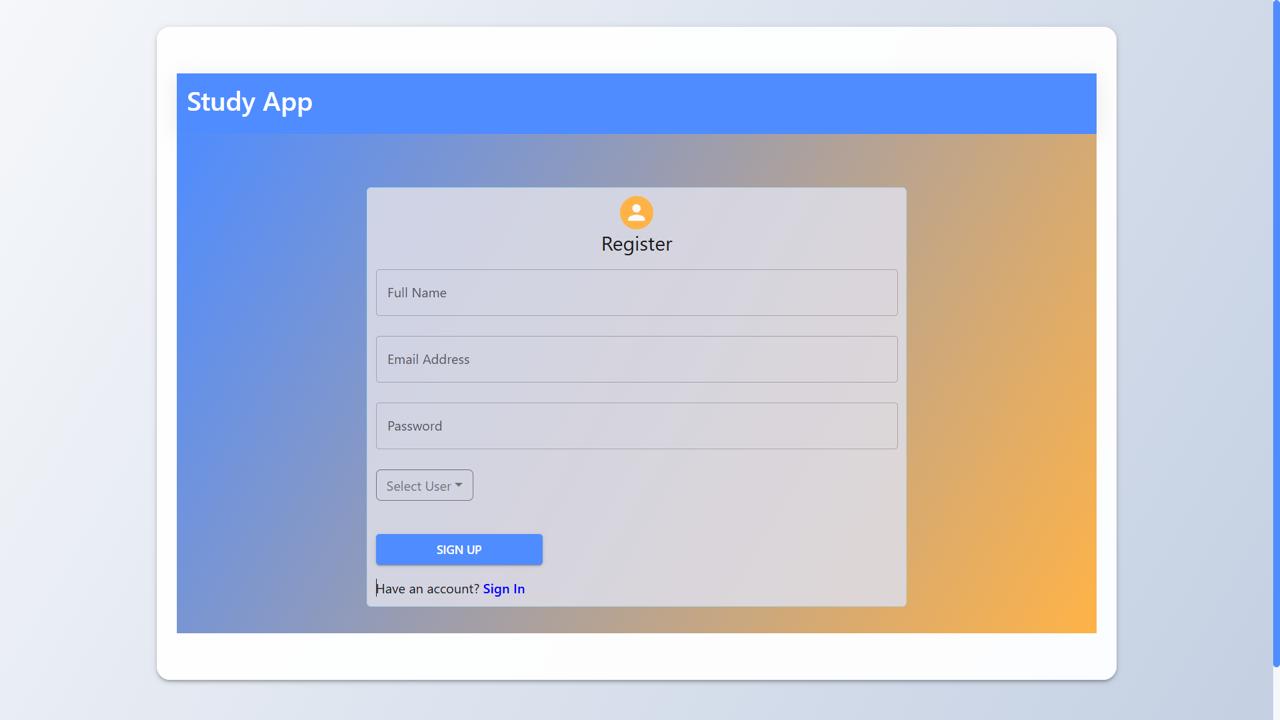
**Milestone 5: Project Implementation:**

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.

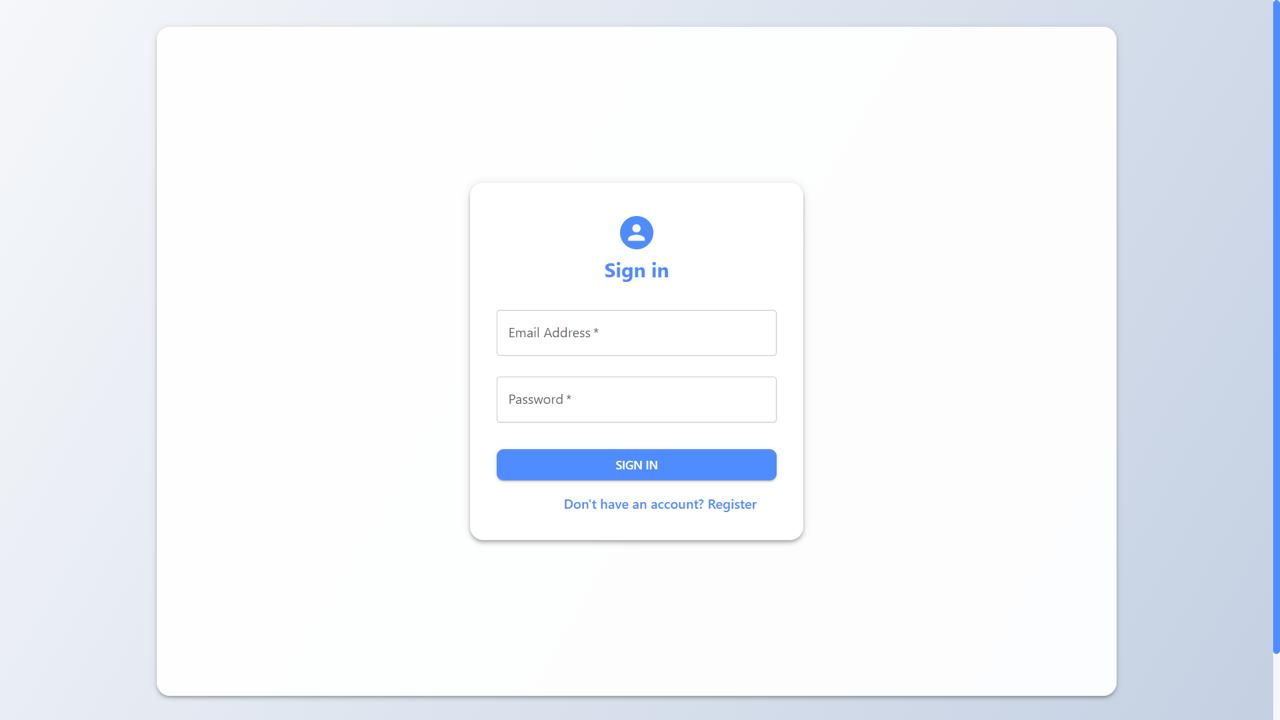
**Landing page:**



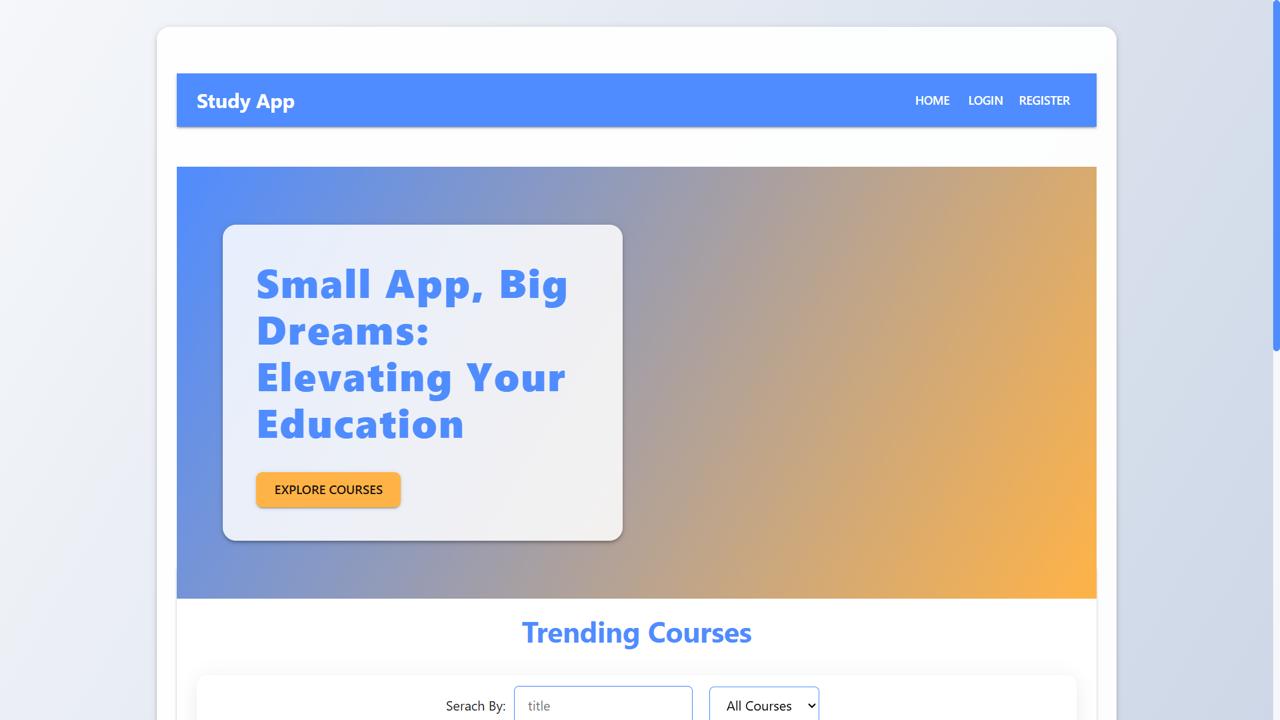
**Register page:**



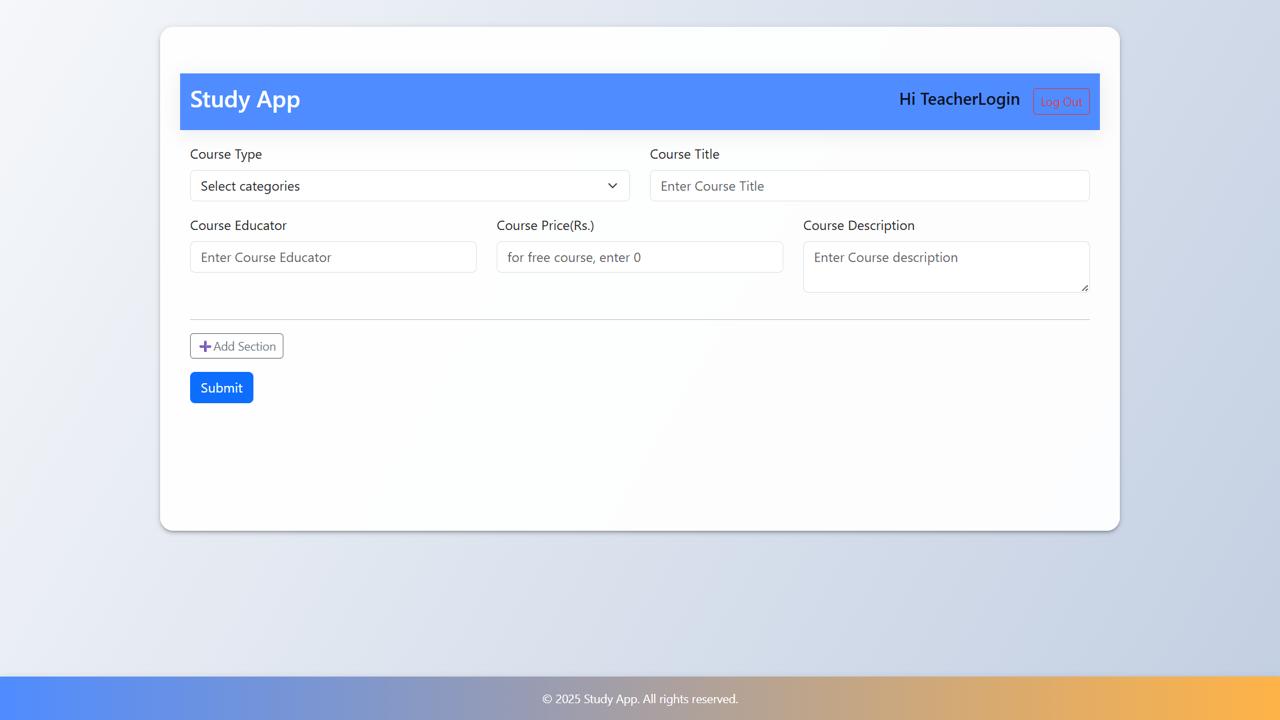
**Login page:**



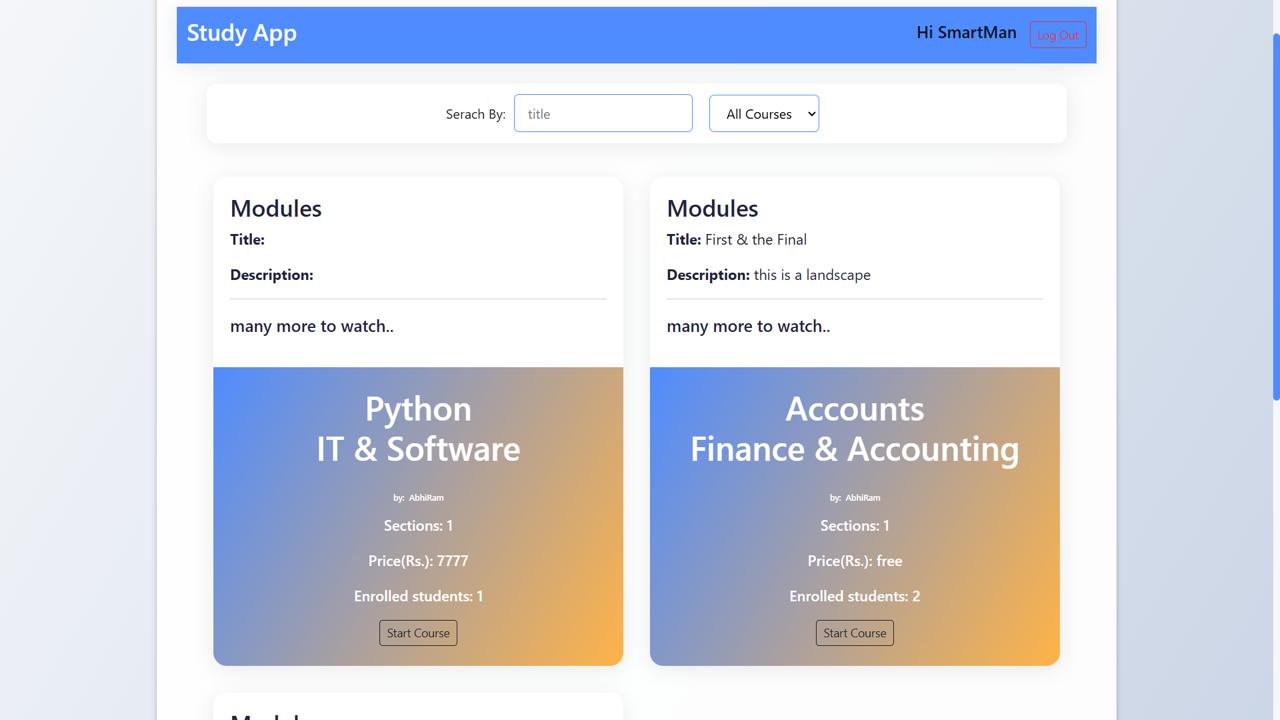
**Admin Dashboard:**

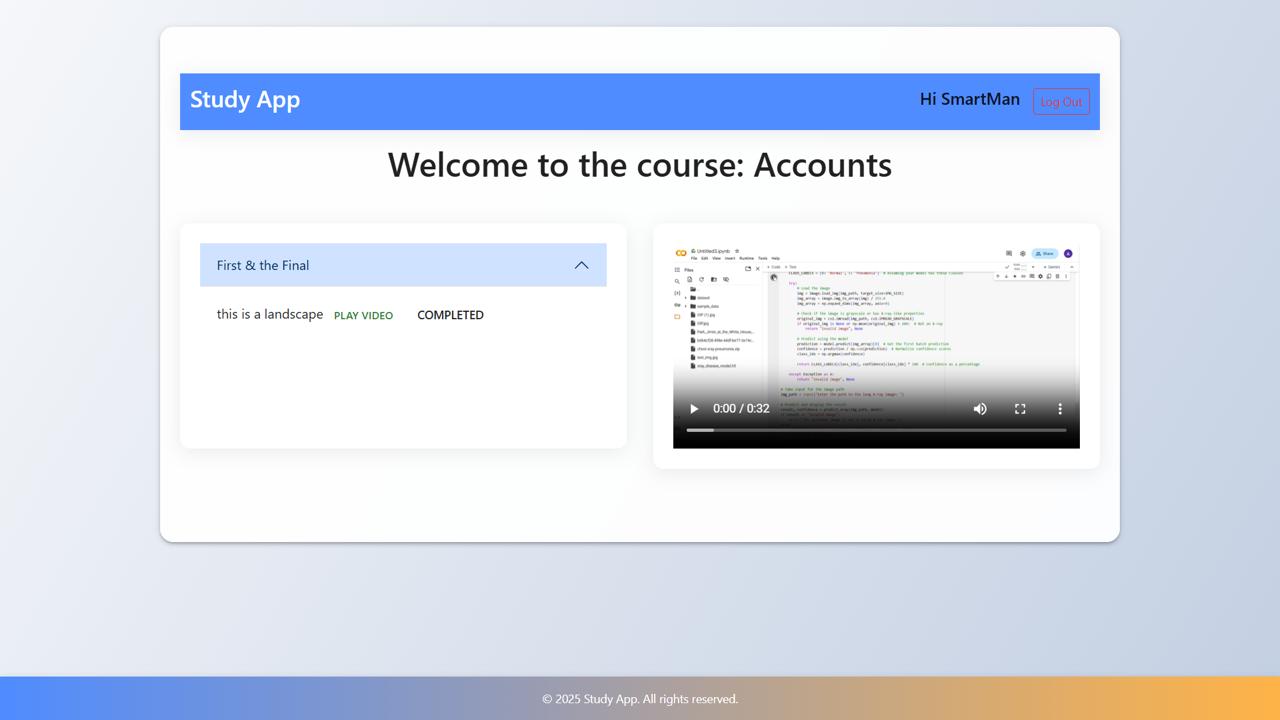


**Teacher Dashboard:**



**Student Dashboard:**





**Note: For the code drive, click on** [**link**](https://drive.google.com/drive/folders/1oWhqVvhFVO7axrB9HVdpNfboo1xlzblZ?usp=drive_link)**, and for the demo link, click on** [**project-implementation.mp**](https://drive.google.com/file/d/1XHBCeOhdYNv2GA6frBoc2AF58d-z9I3P/view?usp=sharing)

**API documentation :**

**Authentication :**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Method** | | |  | | --- | | **Endpoint** | | |  | | --- | | **Description** | | |  | | --- | | **Payload** | | |  | | --- | | **Access** | |
| |  | | --- | | **POST**  **POST** | | |  |  |  | | --- | --- | --- | | |  | | --- | | **/api/register** | | **/api/login** | | | |  | | --- | | **Register new user** | | **Login user** | | |  | | --- | | **{ name, email, password }** | | **{ email, password }** | | |  | | --- | | **Public** | | **Public** | |

**Student Routes :**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Method** | | |  | | --- | | **Endpoint** | | |  | | --- | | **Description** | | |  | | --- | | **Payload / Params** | | |  | | --- | | **Access** | |
| |  | | --- | | **GET** | | **GET**   |  |  | | --- | --- | | **PUT**   |  | | --- | | **GET** | | | | |  | | --- | | **/api/courses** | | **/api/courses/:id** |  |  | | --- | | **/api/enroll/:id** |   **/api/mycourses** | |  |  | | --- | --- | | **Get all available courses**   |  | | --- | | **Get course detail** | |   **Enroll in a course**  **Get enrolled courses** | |  | | --- | | **N/A** | | **Course ID in URL** |  |  | | --- | | **Course ID** |  |  | | --- | | **JWT token required** | | |  | | --- | | **Public** | | **Public** |  |  | | --- | | **Student** |  |  | | --- | | **Student** | |

**Teacher routes :**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Method** | | |  | | --- | | **Endpoint** | | |  | | --- | | **Description** | | |  | | --- | | **Payload** | | |  | | --- | | **Access** | |
| **POST**   |  | | --- | | **PUT** |  |  | | --- | | **DELETE** |  |  |  | | --- | --- | | |  | | --- | | **PUT** | | | |  | | --- | | **/api/courses** | | **/api/courses/:id** |  |  | | --- | | **/api/courses/:id** |  |  | | --- | | **/api/courses/:id/section** | | |  | | --- | | **Create a new cou-**  **-rse** | | **Update a course**   |  | | --- | | **Delete a course** | |  |  | | --- | | **Add section to course** | | |  | | --- | | **{ title, category, desc... }** | | **Fields to update** | | **Course ID** |  |  | | --- | | **{ title, videoURL, desc }** | | |  |  | | --- | --- | | **Teacher** | | | **Teacher**   |  | | --- | | **Teacher** | |  |  | | --- | | **Teacher** | |

**Admin routes :**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Method** | | |  | | --- | | **Endpoint** | | |  | | --- | | **Description** | | |  | | --- | | **Access** | |
| |  |  |  |  | | --- | --- | --- | --- | | **GET**   |  |  |  | | --- | --- | --- | | **GET**   |  |  | | --- | --- | | **DELETE**   |  | | --- | | **PUT** | | | | | |  | | --- | | **/api/users** | | **/api/enrollments** | | **/api/users/:id**   |  | | --- | | **/api/courses/:id** | | | |  | | --- | | **Get all users** | | **View all enrollments** | | **Delete user by ID**   |  | | --- | | **Edit any course details** | | | |  | | --- | | **Admin** | | **Admin** | | **Admin**   |  | | --- | | **Admin** | | |

* **Manual Testing:**
  + Login and registration flow for all roles (student, teacher, admin)
  + Course creation, editing, and enrollment process
  + JWT-based protected route access
  + User interface interactions (filtering, searching, video playback)
* **Error Handling:**
  + Invalid input detection (e.g., missing fields, invalid emails)
  + Token expiry and unauthorized access tests
  + Enrollment edge cases (duplicate enroll, paid access control)
* **Observations:**
  + APIs return proper status codes and messages
  + Role restrictions are enforced correctly
  + No major bugs in critical workflows.

**Known issues** :

Despite successful development and testing, a few limitations and bugs have been identified in the current version of LearnHub:

1. Video Progress Tracking
   * Users cannot resume video playback from their last watched position. Progress tracking is not yet implemented.
2. Simulated Payment System
   * Paid course purchases are handled through a mock flow; no real payment gateway (e.g., Stripe, Razorpay) is integrated yet.
3. No Pagination on Course List
   * The course listing page loads all courses at once. This may affect performance if the number of courses grows large.
4. Basic Certificate System
   * Course completion certificates are basic and not customizable or verifiable by third parties.
5. Limited Error Messages
   * Some backend errors are not clearly displayed to users (e.g., token expiry, server failures)

**Future enhancements :**

To improve user experience, scalability, and functionality, the following enhancements are planned for future versions of LearnHub:

1. Real Payment Gateway Integration
   * Implement Stripe or Razorpay to securely process actual payments for paid courses.
2. Video Progress Tracking
   * Allow students to resume courses from where they left off, with per-section progress indicators.
3. Quizzes and Assessments
   * Add quiz modules after each section to evaluate understanding and award completion based on performance.
4. Chat and Discussion Forums
   * Enable students and teachers to engage through course-specific Q&A forums or live chat.
5. Admin Analytics Dashboard
   * Provide charts and metrics for tracking course popularity, user growth, revenue, and engagement.
6. Pagination and Search Optimization
   * Implement pagination, lazy loading, and advanced search filters for better scalability.
7. Certificate Verification System
   * Add unique certificate IDs and a public verification portal for authenticity checks