

# **Learn Hub – Seamless Appointment Booking for Health**

## **Team Members:**

TEAM ID : LTVIP2025TMID59525

MEMBER	RESPONSIBILITY
1. Kalisetti Prem Kumar	Testing & Documentation
2. Guna Gowri Nandini	Database
3. Gonuguntla Purnesh	Frontend , Backend
4. Gurajala Ameer	Resource Gathering

## **1. Introduction**

**Project Title : LearnHub: Your Center for Skill Enhancement**

## **2. Project Overview**

### **Purpose:**

**Learn Hub** is a full-stack online learning platform (OLP) that enables users to explore, enroll in, and track learning modules and skill development courses. The platform bridges the gap between learners and educators by offering a centralized hub for educational content, progress monitoring, and resource sharing.

### **Features:**

- User Registration & Login (Students & Instructors)
- Browse courses by category, difficulty, or instructor
- Enroll in and track course progress
- Video content support with optional file uploads (PDF, PPT, etc.)
- Instructor dashboard for uploading and managing courses
- Admin panel for monitoring and approving instructor activities
- JWT-based authentication with role-based access control
- Notifications for enrollments, deadlines, and new course uploads

### **3. Architecture**

#### **Frontend:**

Built with React.js, following a modular, scalable component-based structure:

- Component Structure: Functional components using React Hooks (useState, useEffect)
- Routing: React Router v6 for route handling
- State Management: Context API for authentication and shared state
- UI Library: Bootstrap, Material UI, MDBReact for responsive design
- HTTP Requests: Axios for communication with backend APIs

#### **Backend:**

Developed using Node.js and Express.js:

- API Design: RESTful API routes with clear modularity
- Middleware: cors, dotenv, morgan, multer, bcryptjs, jsonwebtoken
- Authentication: JWT-based token system for protected routes
- Structure: Routes, Controllers, Models, and Middleware organized for scalability

#### **Database:**

MongoDB with Mongoose ORM is used to store user data, course details, progress tracking, and uploaded content metadata.

### **4. Setup Instructions**

#### **Prerequisites:**

- Node.js (v16.x or higher)
- npm or yarn
- MongoDB Atlas or local MongoDB instance
- Git

#### **Installation Steps:**

- **Clone the repository**

```
git clone https://github.com/Purnesh4215/LEARNING_HUB  
cd LEARNING_HUB
```

- **Setup environment variables**

```
MONGO_DB = mongodb://localhost:27017/
```

```
JWT_KEY = my-secret-key-123-456-$
```

```
PORT = 8000
```

- **Install Dependencies**

**Backend**

```
cd backend
```

```
npm install
```

**Frontend**

```
cd frontend
```

```
npm install
```

## 5.Folder Structure

### BACKEND:

```
/backend
├── config/
│   └── db.js
├── controllers/
│   ├── adminController.js
│   └── userControllers.js
├── middlewares/
│   └── authMiddleware.js
├── routers/
│   ├── adminRoutes.js
│   └── userRoutes.js
├── schemas/
│   ├── courseModel.js
│   ├── coursePaymentModel.js
│   ├── enrolledCourseModel.js
│   └── userModel.js
├── uploads/
├── .env
├── .gitignore
└── index.js
```

```
package.json
```

```
package-lock.json
```

## FRONTEND:

```
/frontend
├── public/
├── src/
|   ├── assets/
|   ├── components/
|   |   ├── admin/
|   |   |   └── AdminHome.jsx
|   |   ├── common/
|   |   |   ├── AllCourses.jsx
|   |   |   ├── AxiosInstance.jsx
|   |   |   ├── Dashboard.jsx
|   |   |   ├── Home.jsx
|   |   |   ├── Login.jsx
|   |   |   ├── NavBar.jsx
|   |   |   ├── Register.jsx
|   |   |   └── UserHome.jsx
|   |   └── user/
|       ├── student/
|       |   ├── CourseContent.jsx
|       |   ├── EnrolledCourses.jsx
|       |   └── StudentHome.jsx
|       └── teacher/
|           ├── AddCourse.jsx
|           └── TeacherHome.jsx
|   ├── App.css
|   ├── App.jsx
|   ├── main.jsx
|   └── eslintConfig.js
└── index.html
├── package.json
├── package-lock.json
├── vite.config.js
└── README.md
```

## 6. Running the Application

### Frontend:

```
cd frontend
```

```
npm run dev
```

Runs at: <http://localhost:5173>

### Backend:

```
cd backend
```

```
npm start
```

Runs at: <http://localhost:8000>

## 7. API Documentation

### Base URL:

<http://localhost:8000/api>

### Auth Endpoints:

- **POST** /auth/register – Register a new user (student/teacher)
- **POST** /auth/login – Login and receive JWT token

### Course Endpoints:

- **POST** /courses – Create a new course (teacher only)
- **GET** /courses – Fetch all available courses
- **GET** /courses/:id – Fetch course details
- **DELETE** /courses/:id – Delete a course

### User Endpoints:

- **GET** /users/me – Fetch logged-in user's profile
- **GET** /users/teachers – Get a list of instructors

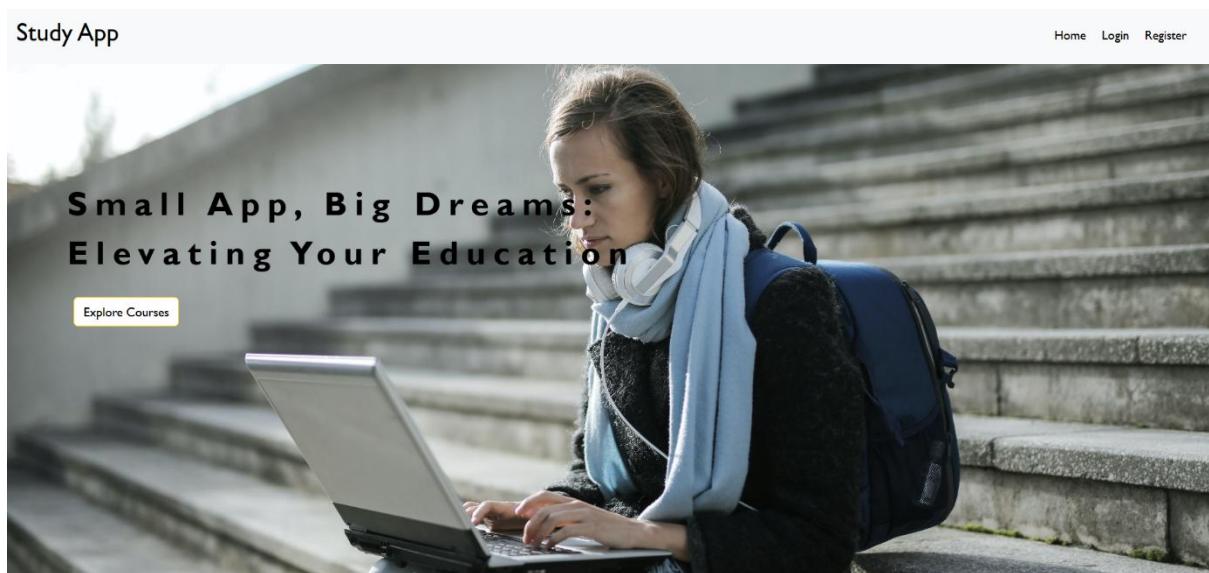
## 8. Authentication

Authentication uses **JWT (JSON Web Tokens)**.

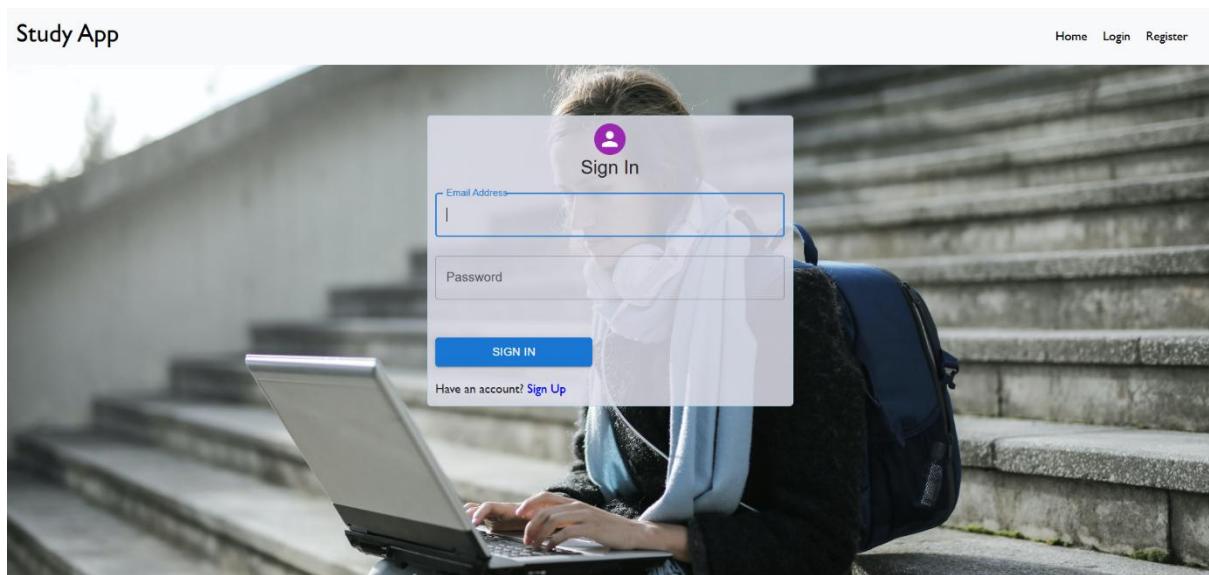
- Token is generated during login and stored in browser localStorage
- Sent in Authorization headers for all protected API requests
- Middleware (authMiddleware.js) verifies token and decodes user info
- Role-based access (student, teacher) supported via logic in backend controllers

## 9. Screenshots or Demo

**HOME PAGE :**



**LOGIN PAGE:**



## REGISTER PAGE :

Study App

Home Login Register

A woman with long dark hair, wearing a light-colored hoodie and dark pants, is sitting on a set of wide stone steps. She is looking down at her open laptop, which is resting on her lap. A blue backpack sits next to her on the steps. In the foreground, there is a semi-transparent registration form overlay. The overlay has a purple circular profile icon at the top left, followed by the word "Register". Below this are four input fields: "Full Name", "Email Address" (which contains a placeholder " "), "Password", and "Select User". At the bottom of the form is a blue "SIGN UP" button. Below the button, the text "Have an account? [Sign In](#)" is visible.

## TEACHER DASHBOARD:

Study App Home Add Course

Hi Teacher Log Out

Course Type Course Title

Select categories Enter Course Title

Course Educator Course Price(Rs.) Course Description

Enter Course Educator for free course, enter 0 Enter Course description

Section Title Section Content (Video or Image)

Enter Section Title Choose File No file chosen

Section Description

Enter Section description

+ Add Section

Submit

The dashboard has a light orange background. At the top, it shows the "Study App" logo, "Home", "Add Course", and a "Hi Teacher" greeting with a "Log Out" button. Below this, there are three main input groups: "Course Type" (with a dropdown menu "Select categories" and a text field "Enter Course Title"), "Course Educator" (with a text field "Enter Course Educator" and a note "for free course, enter 0"), and "Course Description" (with a text area "Enter Course description"). Further down is a section titled "Section Title" with a "Choose File" button and a note "No file chosen", and a text area "Enter Section description". At the bottom left is a button "+ Add Section" and a large blue "Submit" button.

## STUDENT DASHBOARD:

The screenshot shows the Student Dashboard interface. At the top, there is a navigation bar with 'Study App' and links for 'Home' and 'Enrolled Courses'. On the right, it says 'Hi Student' and has a 'Log Out' button. Below the navigation is a search bar with 'Search By: title' and a dropdown menu 'All Courses'. A large orange box contains a course card for '1234 IT & SOFTWARE'. The card includes the title '1234 IT & SOFTWARE', author 'By: 1234', sections '1', price 'Price(Rs.): 1234', and enrolled students '0'. It also features a 'Start Course' button. To the right of the card, the text 'many more to watch..' is visible.

## PAYMENT PAGE:

The screenshot shows the Payment Page for the course '1234'. The page title is 'Payment for 1234 Course'. It displays the educator '1234' and price '1234'. There are input fields for 'Cardholder's Name' (empty), 'Card Holder Name' (empty), 'Card Number' (1234 5678 9012 3457), 'MM/YYYY' (empty), 'Cvv' (empty), and three placeholder dots '...'. At the bottom are 'Close' and 'Pay Now' buttons. The background shows a dark overlay of the Student Dashboard, identical to the one in the previous screenshot.

## **10. Testing**

### **Testing Strategy**

#### **Frontend:**

- Tool: Jest with React Testing Library
- Scope:
  - Component rendering (NavBar, Login, CourseCard, etc.)
  - Form validations (login, register, course upload)
  - Auth flow (protected route redirection)

#### **Backend:**

- Tool: Mocha & Chai
- Scope:
  - API response testing (/auth, /courses, /enroll)
  - JWT-based auth validation
  - Error handling and status code verification

#### **End-to-End (E2E):**

- Tool: Cypress
- Scope:
  - Login > Browse > Enroll > View Progress flow
  - Role-based route access (student vs teacher)
  - Simulated real-user journey with course upload and enrollment

## **11. Known Issues**

Course progress updates require page reload to reflect changes immediately.

File uploads are currently limited to PDF only.

Teacher approval (if admin control is added) is manual and not automated.

Dashboard tables and some layouts are not fully responsive on smaller mobile devices.

## **12. Future Enhancements**

Integrated video sessions between instructors and students.

Course rating & reviews by enrolled students.

AI-based recommendation engine for personalized course suggestions.

Google Calendar sync for upcoming lessons or deadlines.

Multi-language support to reach global learners.

EdTech financial aid integration (scholarship & fee management).