

LearnHub

Online Learning Platform

Full Stack Development with MERN

Full Project Documentation

Technology Stack: MongoDB | Express.js | React.js | Node.js

Version 1.0 | February 2026

1. Introduction

1.1 Project Overview

LearnHub is a comprehensive full-stack Online Learning Platform (OLP) developed using the MERN technology stack — MongoDB, Express.js, React.js, and Node.js. The platform is designed to bridge the gap between educators and learners by providing a scalable, feature-rich, and user-friendly environment for course creation, management, and consumption.

The application supports three distinct user roles: Teachers (educators), Students (learners), and Administrators. Each role has a tailored set of permissions and dashboards that allow for an efficient workflow from content creation to course completion.

1.2 Project Identity

Project Title	LearnHub - Online Learning Platform
Technology Stack	MongoDB, Express.js, React.js, Node.js (MERN)
Frontend Build Tool	Vite (ViteJS)
Authentication	JWT (JSON Web Tokens) with Role-Based Access Control
Development Type	Full-Stack Web Application
Default Port (Frontend)	http://localhost:5172

1.3 Purpose & Goals

The primary goal of LearnHub is to provide a scalable, performant, and accessible digital learning ecosystem. Specific objectives include:

- Enabling teachers to create, manage, and publish courses with structured sections and media content.
- Allowing students to browse, search, filter, enroll in, and complete courses at their own pace.
- Providing administrators with centralized control over all platform users, courses, and enrollments.
- Implementing a secure, token-based authentication system with role-based authorization.
- Supporting both free and paid course models with an integrated payment flow.
- Issuing digital certificates to students upon successful course completion.

1.4 Key Features

LearnHub offers a rich set of features across all user roles:

- Role-based access for Teachers, Students, and Admins.
- Course creation with multimedia sections and categorization.
- Student enrollment and course progress tracking.
- Certificate of Completion generation for finished courses.
- Course filtering and searching by name, category, and other criteria.
- Paid course support with a payment gateway workflow.
- Admin oversight of all users, courses, and enrollment records.
- Secure JWT authentication with bcrypt password hashing.
- Responsive user interface built with React, Bootstrap, and Material UI.

2. System Architecture

2.1 Architecture Overview

LearnHub follows a client-server architecture where the React.js frontend communicates with the Node.js/Express.js backend via RESTful API calls. The backend interfaces with a MongoDB database for persistent data storage. Authentication is handled through JSON Web Tokens (JWT), which are issued upon login and verified on every protected API request.

2.2 Technology Stack Details

Frontend

- React.js — Component-based UI library for building dynamic, interactive user interfaces.
- Vite — Next-generation frontend build tool providing fast Hot Module Replacement (HMR) and optimized production builds.
- Bootstrap & Material UI (MUI) — CSS/component libraries for responsive layouts and pre-built UI components.
- Axios — Promise-based HTTP client used to communicate with the backend REST API.
- Ant Design (Antd) & mdb-react-ui-kit — Additional UI component libraries for enhanced design elements.

Backend

- Node.js — JavaScript runtime for executing server-side logic.
- Express.js — Minimalist web framework for Node.js, handling routing, middleware, and API structure.
- Mongoose — ODM (Object-Document Mapping) library for MongoDB, providing schema definitions and query helpers.
- bcryptjs — Library for hashing and comparing passwords securely.
- jsonwebtoken — Library for generating and verifying JWT tokens.
- Multer — Middleware for handling multipart/form-data, used for file/image uploads.
- dotenv — Loads environment variables from a .env file.
- cors — Middleware enabling Cross-Origin Resource Sharing between frontend and backend.
- Nodemon — Development utility that automatically restarts the server on file changes.

Database

- MongoDB — NoSQL document database storing data in flexible, JSON-like BSON format.
- Collections: Users, Courses, and Enrollments.
- Mongoose schemas enforce structure and validation on top of MongoDB's flexible storage.

2.3 Data Flow

The data flow within LearnHub follows a standard RESTful pattern:

1. The user interacts with the React frontend (browser/mobile).
2. The frontend sends HTTP requests (GET, POST, PUT, DELETE) to Express API endpoints.

3. The Express middleware validates the JWT token in the Authorization header.
4. If authorized, the corresponding controller processes the request and queries/updates MongoDB via Mongoose.
5. The database returns the result, which the controller formats as a JSON response.
6. The frontend receives the response and updates the UI state accordingly.

3. Database Design

3.1 MongoDB Collections

The LearnHub application uses two primary MongoDB collections: Users and Courses. MongoDB auto-generates a unique ObjectId (`_id`) for each document, which serves as the primary identifier.

3.2 Users Collection

The Users collection stores all platform users regardless of their role (Student, Teacher, or Admin).

Field	Type	Description
<code>_id</code>	<i>ObjectId</i>	Auto-generated unique identifier (Primary Key)
<code>name</code>	<i>String</i>	Full name of the user
<code>email</code>	<i>String</i>	User's email address (unique, used for login)
<code>password</code>	<i>String</i>	Bcrypt-hashed password string
<code>type</code>	<i>String</i>	Role of the user: 'student', 'teacher', or 'admin'

3.3 Courses Collection

The Courses collection stores all course data, linked to the creating teacher via the `userID` foreign key reference.

Field	Type	Description
<code>_id</code>	<i>ObjectId</i>	Auto-generated unique identifier (Primary Key)
<code>userID</code>	<i>ObjectId</i>	Reference to the teacher (User) who created the course — acts as a foreign key
<code>C_educator</code>	<i>String</i>	Name or identifier of the course educator/teacher
<code>C_categories</code>	<i>String/Array</i>	Course category tags (e.g., 'Web Development', 'Data Science')
<code>C_title</code>	<i>String</i>	Title of the course
<code>C_description</code>	<i>String</i>	Detailed description of the course content and objectives
<code>sections</code>	<i>Array</i>	Array of section objects, each containing title, content, and media URLs
<code>C_price</code>	<i>Number</i>	Course price. Set to 0 for free courses
<code>enrolled</code>	<i>Array</i>	Array of student user IDs who are enrolled in the course
<code>Pre-requisites</code>	<i>String/Array</i>	Prerequisite knowledge or courses required before enrollment

4. Project Setup & Configuration

4.1 Prerequisites

Before setting up the LearnHub development environment, ensure the following tools and technologies are installed on your machine:

Node.js and npm

Node.js is the JavaScript runtime that powers the backend server. npm (Node Package Manager) is bundled with Node.js and is used to install all project dependencies.

- Download from: <https://nodejs.org/en/download/>
- Verify installation: Run 'node --version' and 'npm --version' in your terminal.

```
npm init
```

MongoDB

MongoDB is the NoSQL database used to persist all application data. You can install it locally or use MongoDB Atlas for a cloud-hosted solution.

- Download Community Edition: <https://www.mongodb.com/try/download/community>
- Installation Guide: <https://docs.mongodb.com/manual/installation/>
- After installation, ensure the MongoDB service is running before starting the backend.

Vite (Frontend Build Tool)

Vite is used to scaffold and develop the React frontend. It provides significantly faster development server startup and HMR (Hot Module Replacement) compared to older tools like Create React App.

```
npm create vite@latest
```

React.js

React.js is the frontend UI library. It is installed as a project dependency rather than globally.

- Official Guide: <https://reactjs.org/docs/create-a-new-react-app.html>

4.2 Installation Steps

Step 1: Clone the Repository

Clone the project repository from GitHub to your local machine:

```
git clone <repository-url>
cd learnhub
```

Step 2: Install Frontend Dependencies

```
cd frontend
npm install
```

Step 3: Install Backend Dependencies

```
cd ../backend
npm install
```

Step 4: Configure Environment Variables

Create a `.env` file in the backend folder and define the following variables:

```
MONGO_URI=mongodb://localhost:27017/learnhub
JWT_SECRET=your_jwt_secret_key_here
PORT=5000
```

Step 5: Start the Development Servers

Start the backend server:

```
cd backend
npm start
```

In a separate terminal, start the frontend development server:

```
cd frontend
npm run dev
```

The application will be accessible at `http://localhost:5172` (frontend) and the backend API will run at `http://localhost:5000`.

4.3 Backend Dependencies (package.json)

Package	Version	Purpose
express	^4.x	Web framework for building the REST API and handling HTTP requests
mongoose	^7.x	ODM for MongoDB — provides schema-based data modeling
bcryptjs	^2.x	Password hashing and comparison for secure user authentication
jsonwebtoken	^9.x	Generates and verifies JWT tokens for authentication
cors	^2.x	Enables Cross-Origin Resource Sharing for frontend-backend communication
dotenv	^16.x	Loads environment variables from the <code>.env</code> file
multer	^1.x	Handles file/image uploads via multipart/form-data
nodemon	^3.x	Auto-restarts the Node.js server on code changes (dev dependency)

4.4 Frontend Dependencies (package.json)

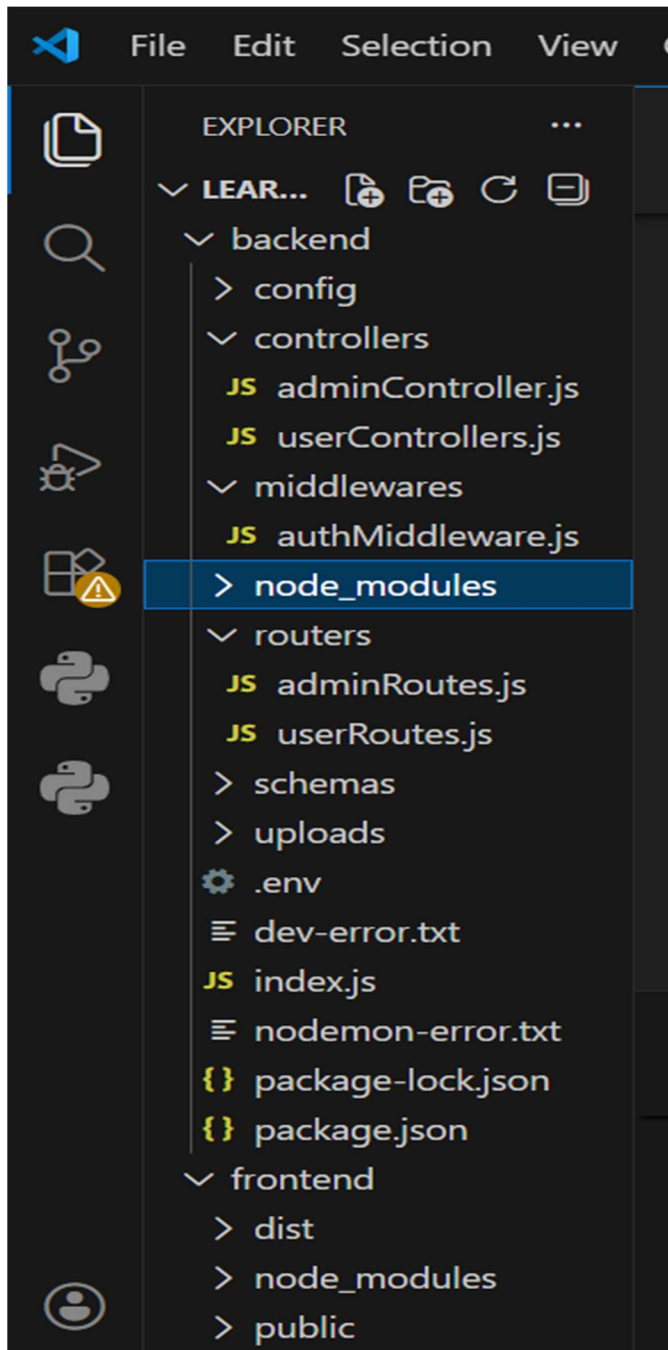
Package	Version	Purpose
react	^18.x	Core library for building the component-based user interface
react-dom	^18.x	DOM rendering for React components
axios	^1.x	HTTP client for making API requests to the backend

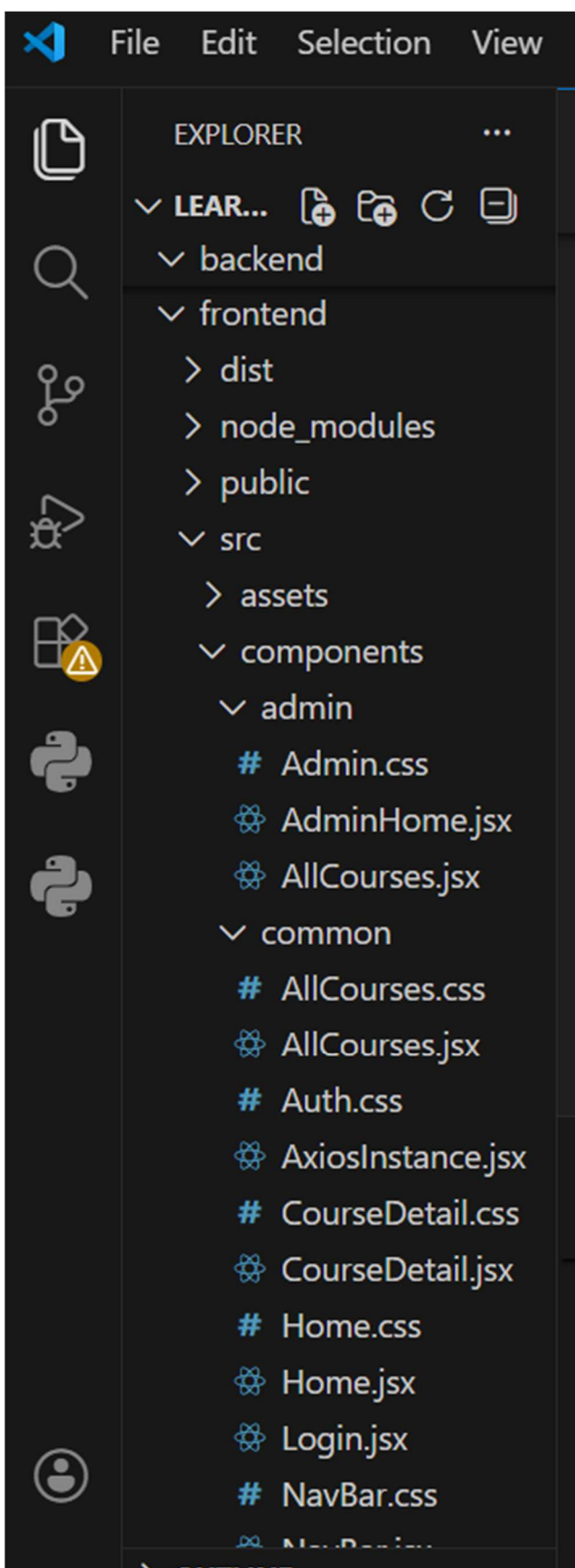
<code>bootstrap</code>	<code>^5.x</code>	CSS framework for responsive grid and utility classes
<code>@mui/material</code>	<code>^5.x</code>	Material Design component library for advanced UI elements
<code>antd</code>	<code>^5.x</code>	Ant Design component library — buttons, forms, modals, etc.
<code>mdb-react-ui-kit</code>	<code>^7.x</code>	Material Design Bootstrap components for React
<code>react-bootstrap</code>	<code>^2.x</code>	Bootstrap components rebuilt for React

5. Project Structure

5.1 Folder Architecture

The project is organized into two top-level directories: frontend and backend. This separation of concerns promotes modularity and makes it easier to manage, scale, and deploy each part independently.











File Edit Selection View



EXPLORER



✓ LEAR...    



✓ frontend



✓ src

✓ components

✓ common




✓ user




✓ student

Certificate.css


 Certificate.jsx

CourseContent...

 CourseContent...




Student.css

 StudentHome.jsx




✓ teacher


AddCourse.css

 AddCourse.jsx

ManageCourse...


 ManageCourse...

Teacher.css

 TeacherHome.jsx

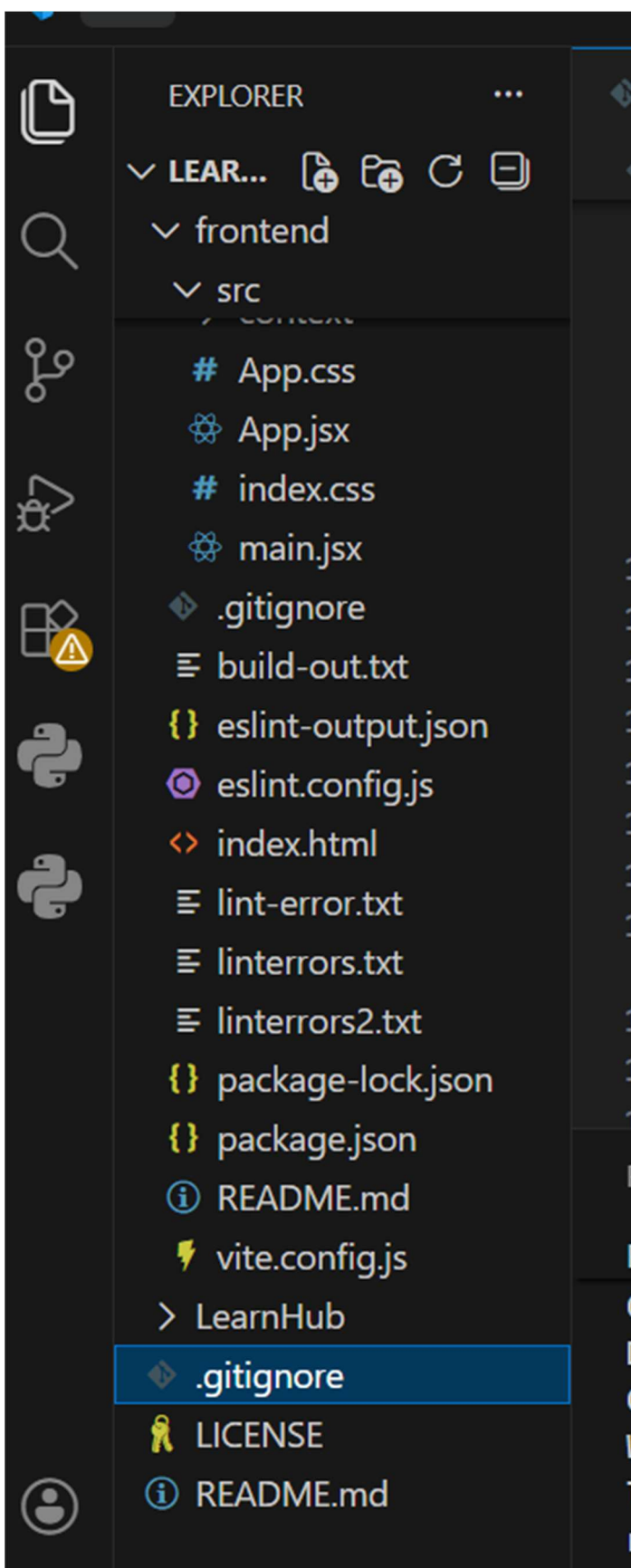
> context

App.css

 App.jsx

index.css





Frontend Folder Structure

```
frontend/  
  src/  
    components/    # Reusable UI components (Navbar, Cards, Modals)  
    pages/         # Full page components (Home, Login, Dashboard)  
    services/      # Axios API service functions  
    assets/        # Static files (images, icons)  
    App.jsx        # Root component with route definitions  
    main.jsx       # Entry point, renders App into the DOM  
  public/          # Static public assets  
  index.html       # HTML shell  
  vite.config.js   # Vite build configuration  
  package.json     # Frontend dependencies
```

Backend Folder Structure

```
backend/  
  models/          # Mongoose schema definitions (User.js, Course.js)  
  routes/          # Express route handlers (authRoutes.js, courseRoutes.js)  
  controllers/     # Business logic for each route  
  middleware/      # Auth middleware (authMiddleware.js)  
  config/          # Database connection (config.js)  
  uploads/         # Uploaded files and media  
  index.js         # Express app entry point  
  .env             # Environment variables (gitignored)  
  package.json     # Backend dependencies
```

5.2 Frontend Structure Screenshot

The following screenshot shows the actual frontend folder and file organization as seen in the IDE:

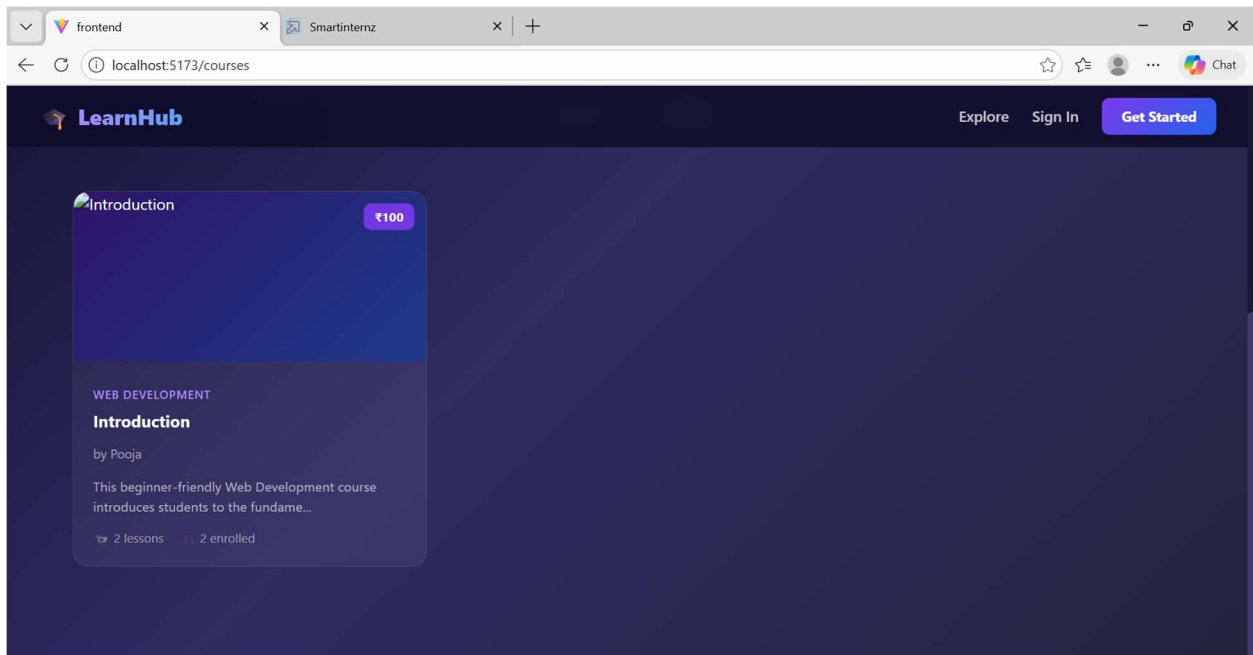


Figure 1: Frontend Project Folder Structure

5.3 Backend Structure Screenshot

The following screenshot shows the actual backend folder and file organization as seen in the IDE:

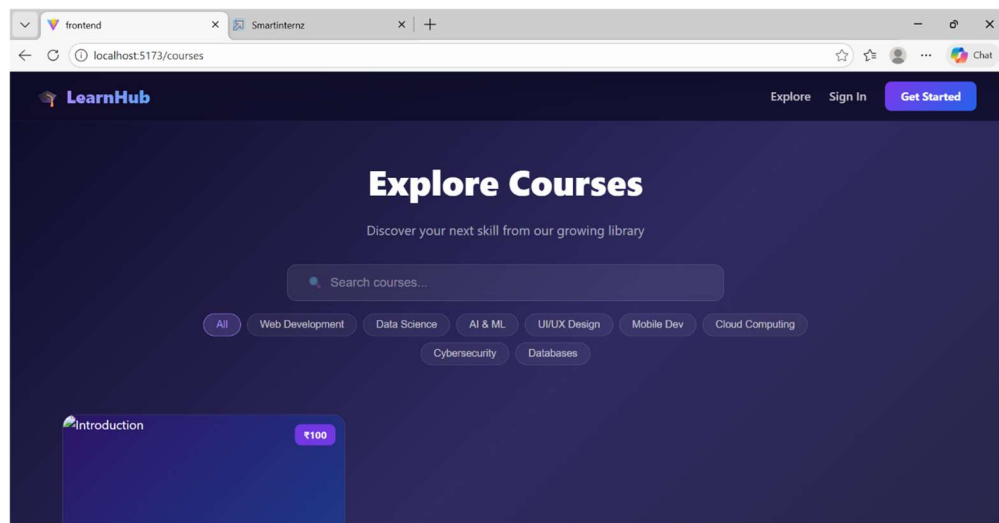


Figure 2: Backend Project Folder Structure

6. Backend Development

6.1 Express Server Setup

The backend server is initialized in the `index.js` file at the root of the backend directory. The server uses `Express.js` as the web framework and connects to the MongoDB database via `Mongoose`.

Server Configuration

The main server file performs the following setup tasks:

- 7. Imports `Express` and other required middleware packages.
- 8. Defines the server port from the `.env` file (default: 5000).
- 9. Configures `CORS` to allow requests from the React frontend.
- 10. Adds `body-parser` middleware (or `express.json()`) to parse incoming JSON request bodies.
- 11. Imports and mounts API route files.
- 12. Connects to the MongoDB database via the `config.js` module.
- 13. Starts the HTTP server and listens on the specified port.

6.2 Authentication Middleware

Authentication is implemented using JSON Web Tokens (JWT). When a user logs in, the server issues a signed JWT token which the frontend stores (typically in `localStorage` or a cookie). On subsequent requests to protected routes, the frontend sends this token in the `Authorization` header.

The `authMiddleware.js` file in the `middleware/` folder intercepts requests and validates the token before passing control to the route handler. If the token is missing or invalid, the middleware returns a 401 Unauthorized response.

Authentication Flow

- 14. User submits login credentials (email + password) to `POST /api/auth/login`.
- 15. The server retrieves the user from MongoDB by email.
- 16. `bcryptjs` compares the submitted password against the stored hash.
- 17. On match, the server signs a JWT with the user's ID and role, setting an expiry.
- 18. The token is returned to the client in the response body.
- 19. The client includes the token in the `Authorization: Bearer <token>` header for all subsequent requests.
- 20. The `authMiddleware` validates the token and attaches the decoded user data to `req.user`.

6.3 API Endpoints

Authentication APIs

Method	Endpoint	Access	Description
POST	/api/auth/register	Public	Register a new user with name, email, password, and role

POST	/api/auth/login	<i>Public</i>	Login with email and password, returns JWT token
GET	/api/auth/profile	<i>Protected</i>	Get the profile of the currently authenticated user
PUT	/api/auth/profile	<i>Protected</i>	Update user profile information

Course APIs

Method	Endpoint	Access	Description
GET	/api/courses	<i>Public</i>	Retrieve all available courses (with search/filter support)
GET	/api/courses/:id	<i>Public</i>	Get details of a specific course by ID
POST	/api/courses	<i>Teacher</i>	Create a new course (requires Teacher role JWT)
PUT	/api/courses/:id	<i>Teacher/Admin</i>	Update course details
DELETE	/api/courses/:id	<i>Teacher/Admin</i>	Delete a course (only if no students enrolled)
POST	/api/courses/:id/sections	<i>Teacher</i>	Add a new section to an existing course
DELETE	/api/courses/:id/sections/:sectionId	<i>Teacher</i>	Remove a section from a course

Enrollment APIs

Method	Endpoint	Access	Description
POST	/api/enrollments/:courseId	<i>Student</i>	Enroll the authenticated student in a course
GET	/api/enrollments/my-courses	<i>Student</i>	Get all courses the student is enrolled in
GET	/api/enrollments/all	<i>Admin</i>	View all enrollment records across all students
PUT	/api/enrollments/:courseId/progress	<i>Student</i>	Update the student's progress in a course
GET	/api/enrollments/:courseId/certificate	<i>Student</i>	Download completion certificate

7. Application Flow & User Roles

7.1 Role Overview

LearnHub implements a three-tier Role-Based Access Control (RBAC) system. Each user is assigned a role upon registration, and the backend middleware enforces access restrictions based on this role. The three roles are Teacher, Student, and Admin.

7.2 Teacher Role

Teachers are the content creators of the platform. They are responsible for building and maintaining the course catalog.

- Create new courses with full details: title, description, categories, price, prerequisites.
- Add, edit, and remove sections within each course (videos, text, quizzes).
- Delete courses that have no active student enrollments.
- View enrollment statistics for their own courses.
- Access a dedicated Teacher Dashboard showing their courses and performance.

7.3 Student Role

Students are the consumers of the platform. They discover, enroll in, and complete courses.

- Browse and search the full course catalog using filters (name, category, price, educator).
- View detailed course information including sections, prerequisites, and reviews.
- Enroll in free courses directly; purchase paid courses through the payment flow.
- Resume a course from exactly where they left off (progress tracking).
- Download a Certificate of Completion upon finishing all course sections.
- Access a personalized Student Dashboard listing all enrolled courses and progress.

7.4 Admin Role

Administrators have full oversight and management capability over the entire platform.

- View, edit, and delete any course on the platform regardless of who created it.
- Manage all registered users — view profiles, change roles, deactivate accounts.
- View comprehensive enrollment records across all students and all courses.
- Monitor platform health, flagged content, and usage statistics.
- Access the Admin Panel with a full data overview dashboard.

8. Frontend Development

8.1 React Component Structure

The React frontend is built using a modular component architecture. Pages are composed of smaller, reusable components to maximize code reuse and maintainability.

Key Pages

- Landing Page — Public homepage showcasing platform features and a course preview.
- Register Page — New user registration form with role selection (Student or Teacher).
- Login Page — Authentication form that stores the returned JWT token.
- Course List Page — Browsable and searchable list of all available courses.
- Course Detail Page — Full course view with section listing and enroll button.
- Teacher Dashboard — Course management interface for teachers.
- Student Dashboard — Enrolled course tracker with progress indicators.
- Admin Panel — User and course management interface for administrators.

Routing

React Router is used for client-side navigation. Protected routes check for the presence and validity of a JWT token before rendering. Unauthorized access attempts redirect the user to the login page.

8.2 State Management & API Integration

Axios is used throughout the frontend to communicate with the backend REST API. API call functions are typically organized in a `services/` directory to keep components clean and promote reusability. State is managed using React hooks (`useState`, `useEffect`, `useContext`).

8.3 UI Libraries

The application uses a combination of UI libraries to achieve a polished, professional look:

- Bootstrap 5 provides the responsive grid system and base utility classes.
- Material UI (MUI) supplies higher-level components such as Cards, Modals, Drawers, and DataGrids.
- Ant Design (Antd) is used for data-heavy components like Tables, Forms, and Notifications.
- MDB React UI Kit offers themed Bootstrap components with Material Design aesthetics.

9. Application Screenshots

The following screenshots illustrate the key user-facing interfaces of the LearnHub platform. All screenshots were captured from a fully functional instance of the application.

9.1 Landing Page

The landing page is the first screen users see when they visit LearnHub. It presents the platform's value proposition, featured courses, and clear call-to-action buttons for Registration and Login.

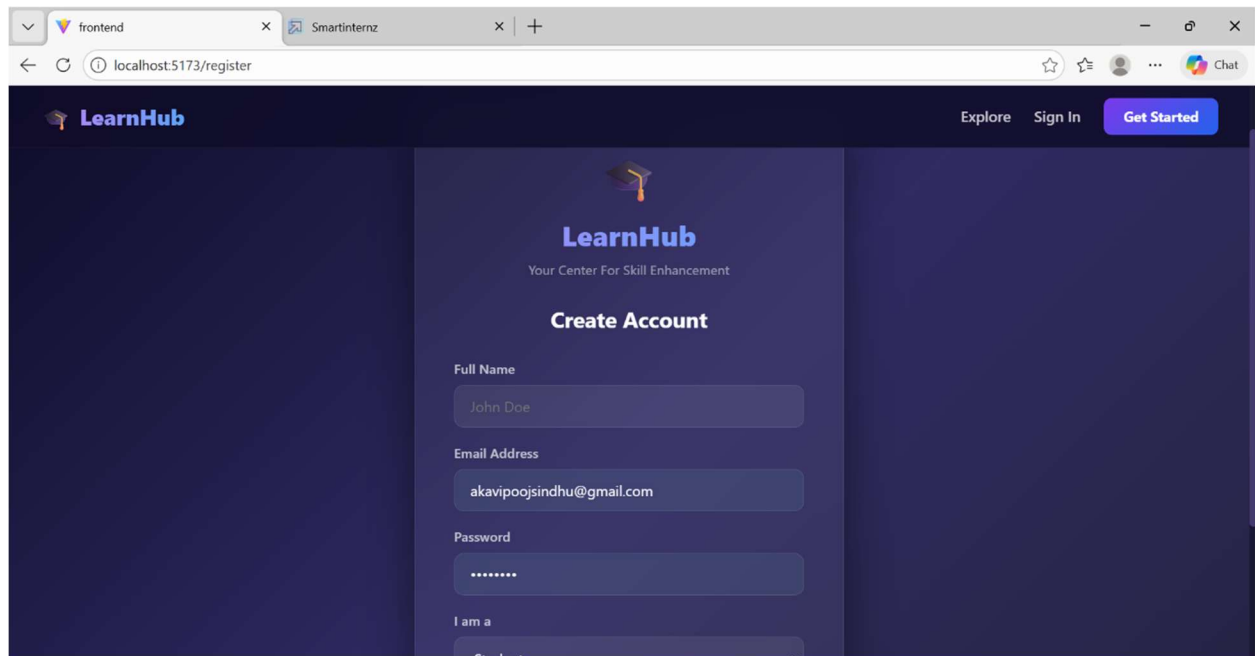


Figure 3: LearnHub Landing Page

9.2 Registration Page

The Registration page allows new users to create an account. Users select their role (Student or Teacher) during registration, which determines their access level throughout the platform.

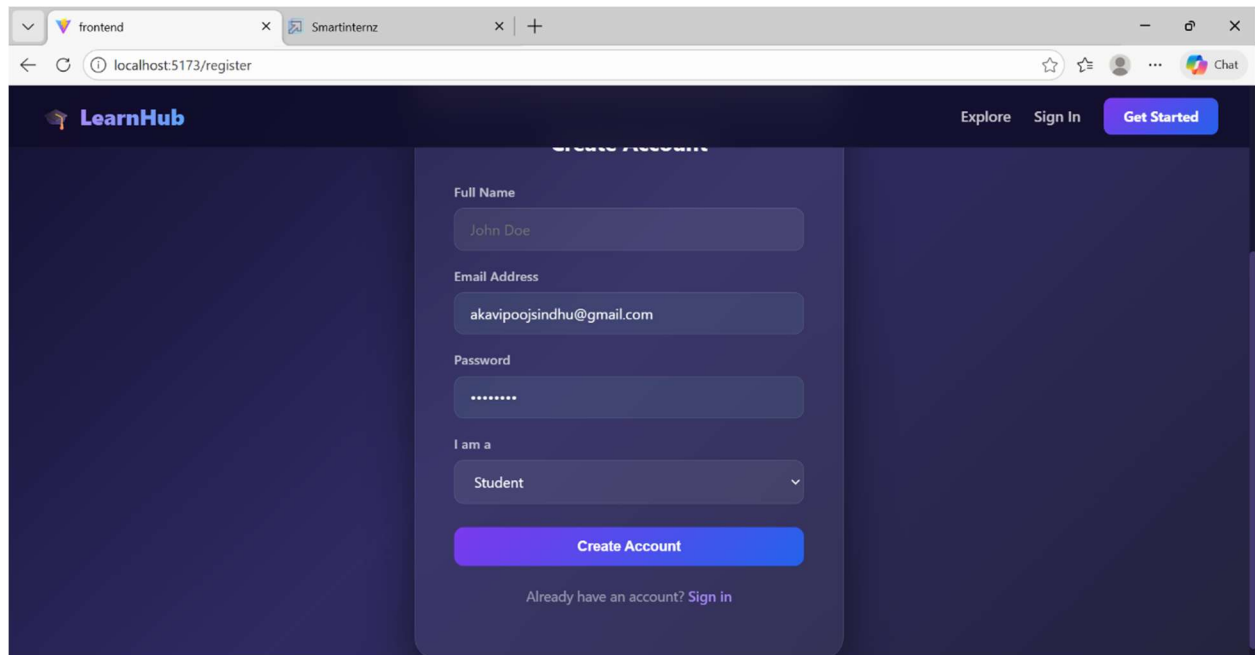


Figure 4: User Registration Page

9.3 Login Page

The Login page accepts the user's email and password credentials. Upon successful authentication, a JWT token is issued and the user is redirected to their role-specific dashboard.

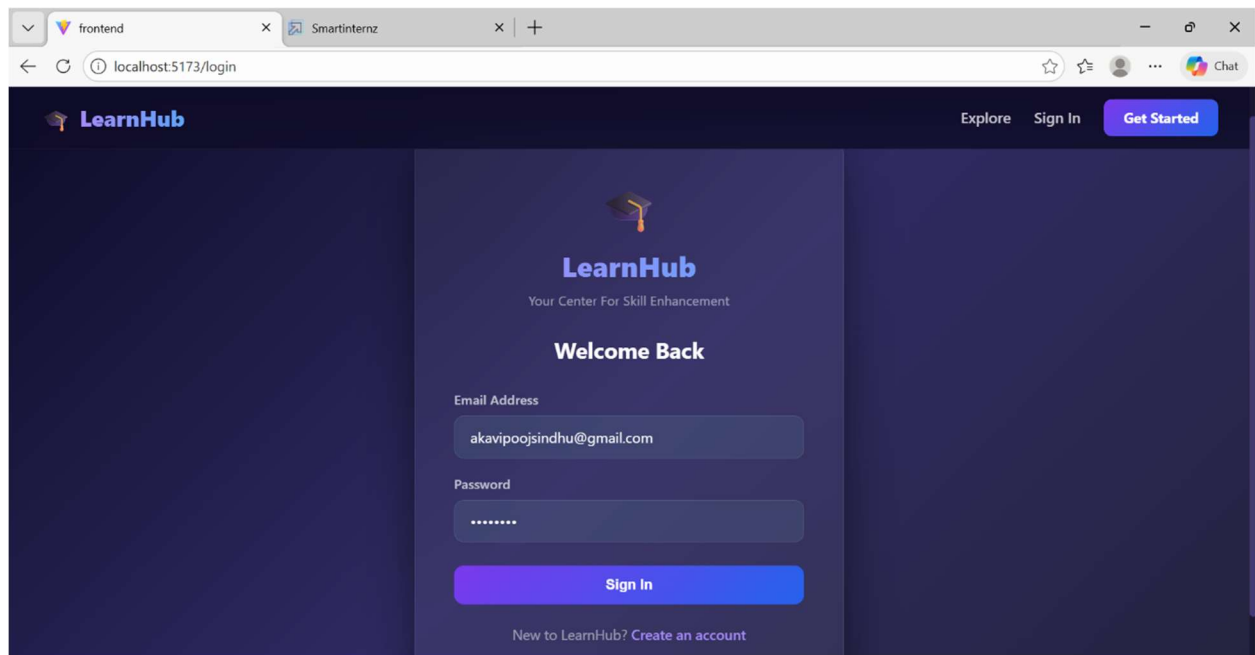


Figure 5: User Login Page

9.4 Teacher Dashboard

The Teacher Dashboard provides a centralized view for educators to manage their courses. Teachers can view course enrollment counts, add new courses, edit existing courses, and manage course sections from this screen.

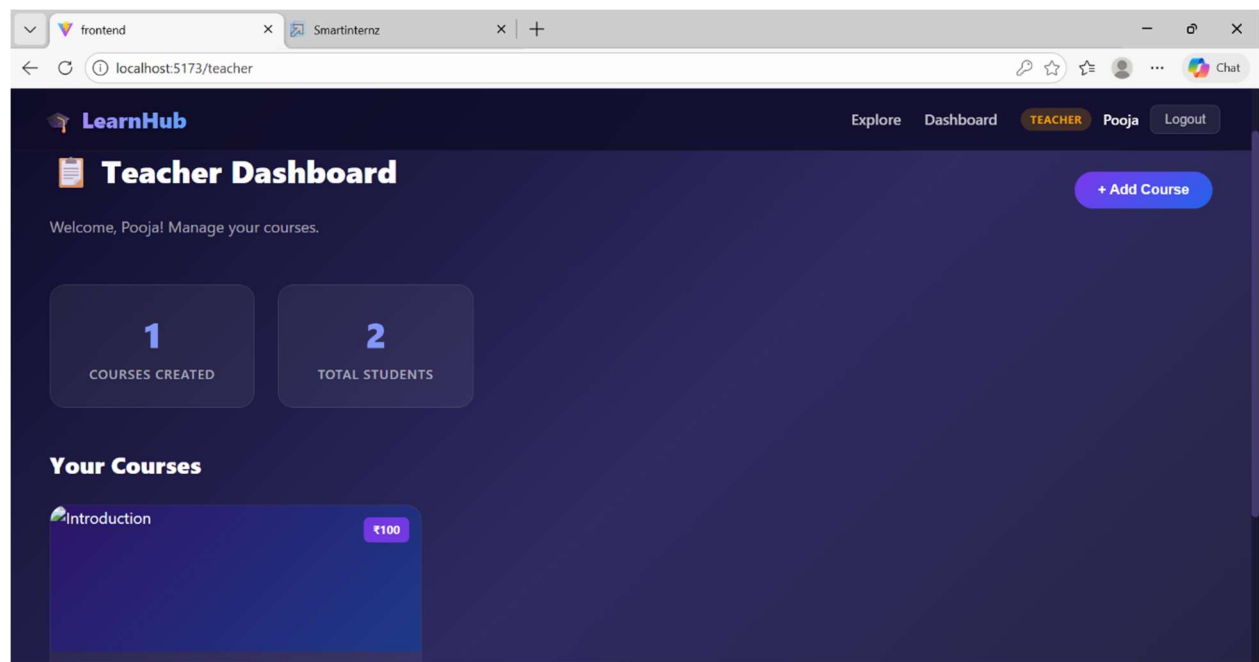


Figure 6: Teacher Dashboard

9.5 Student Dashboard

The Student Dashboard displays all courses the student is currently enrolled in, along with progress indicators for each course. Students can resume learning, view course details, or download certificates for completed courses.

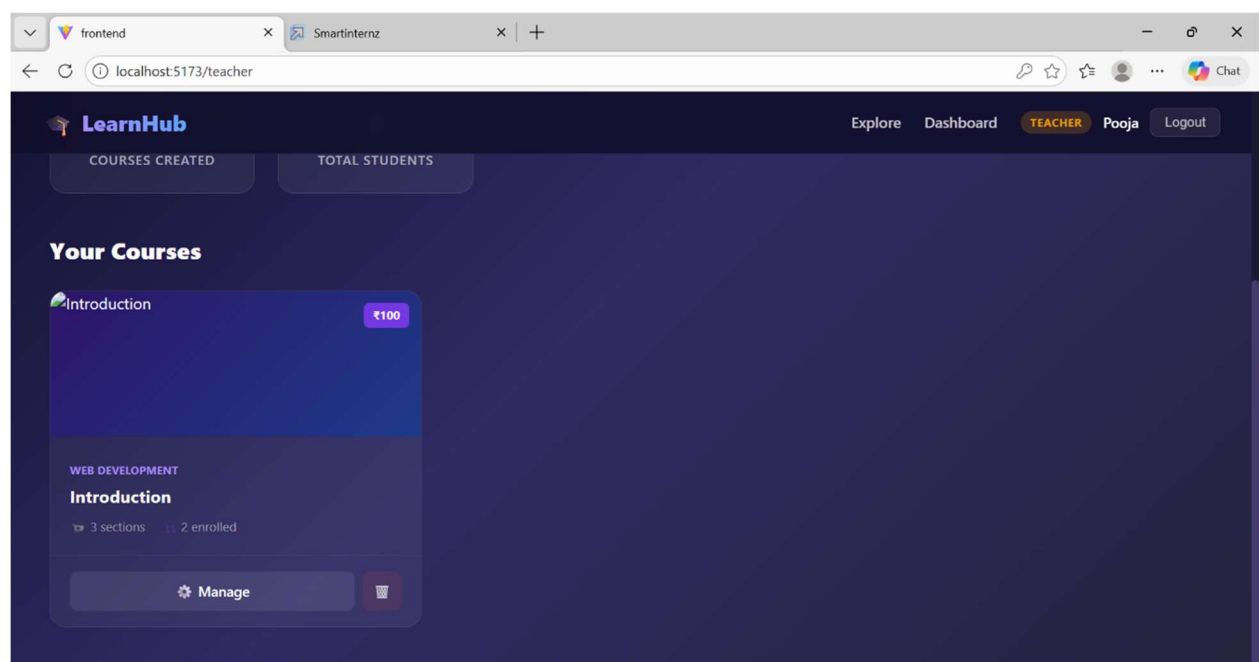


Figure 7: Student Dashboard

9.6 Additional UI Screens

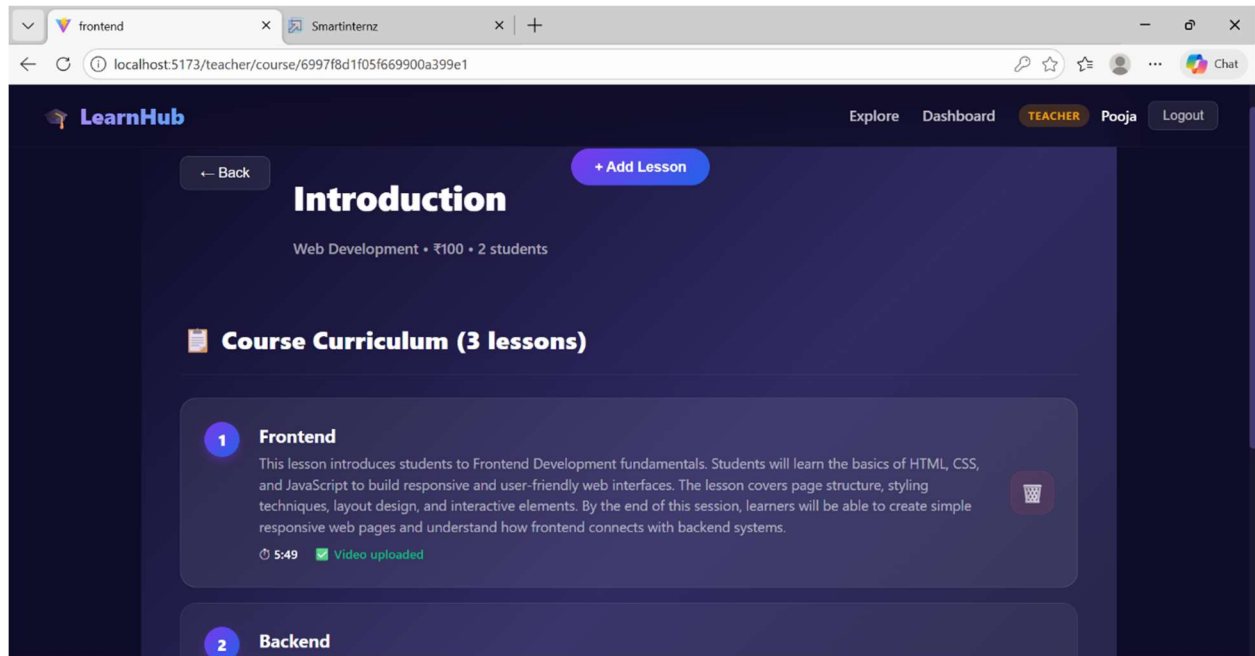


Figure 8: Course Catalog / Browse Courses

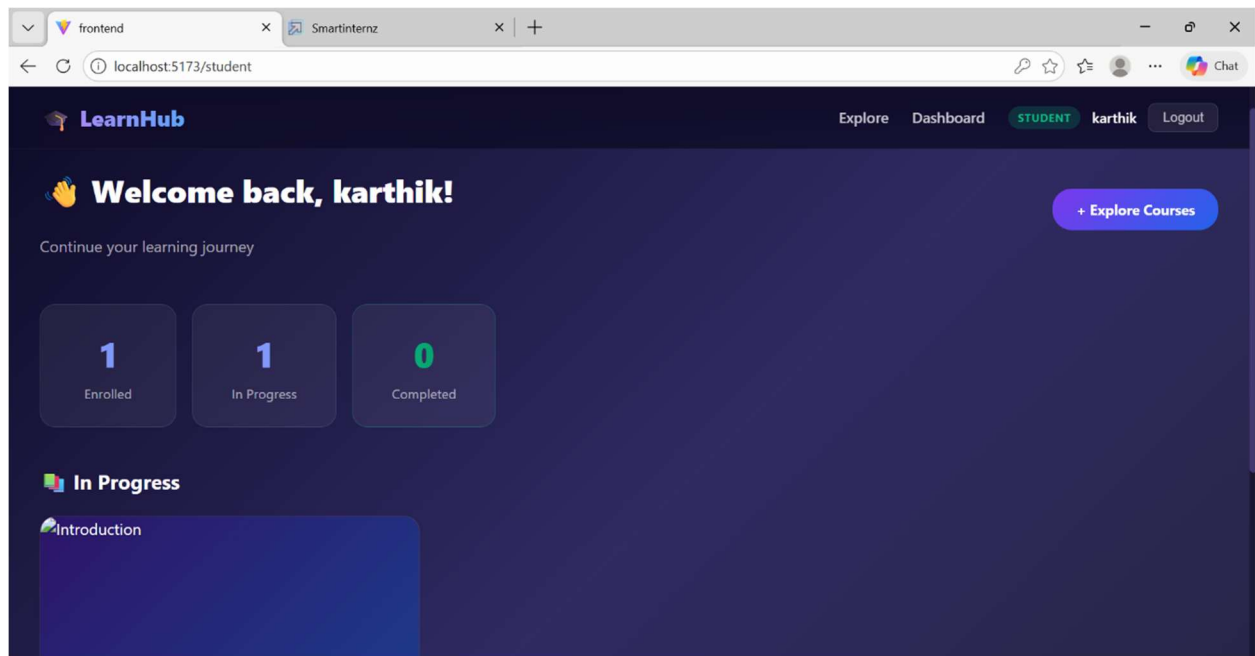


Figure 9: Course Detail Page

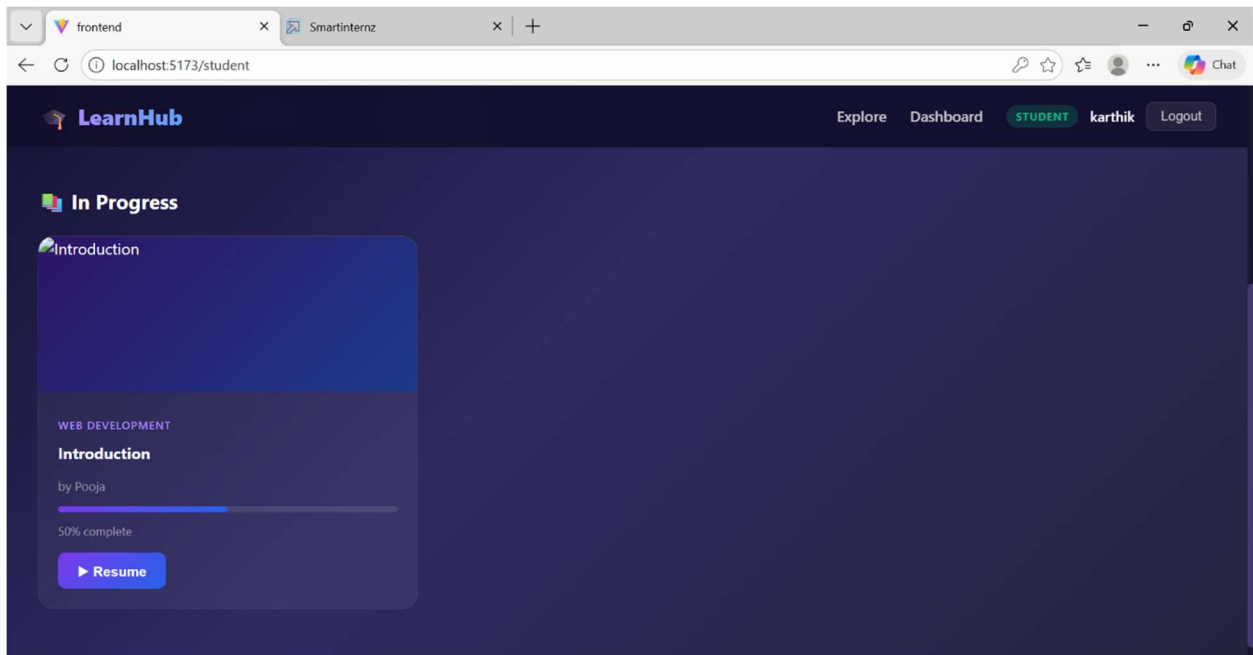


Figure 10: Enrollment / Payment Screen

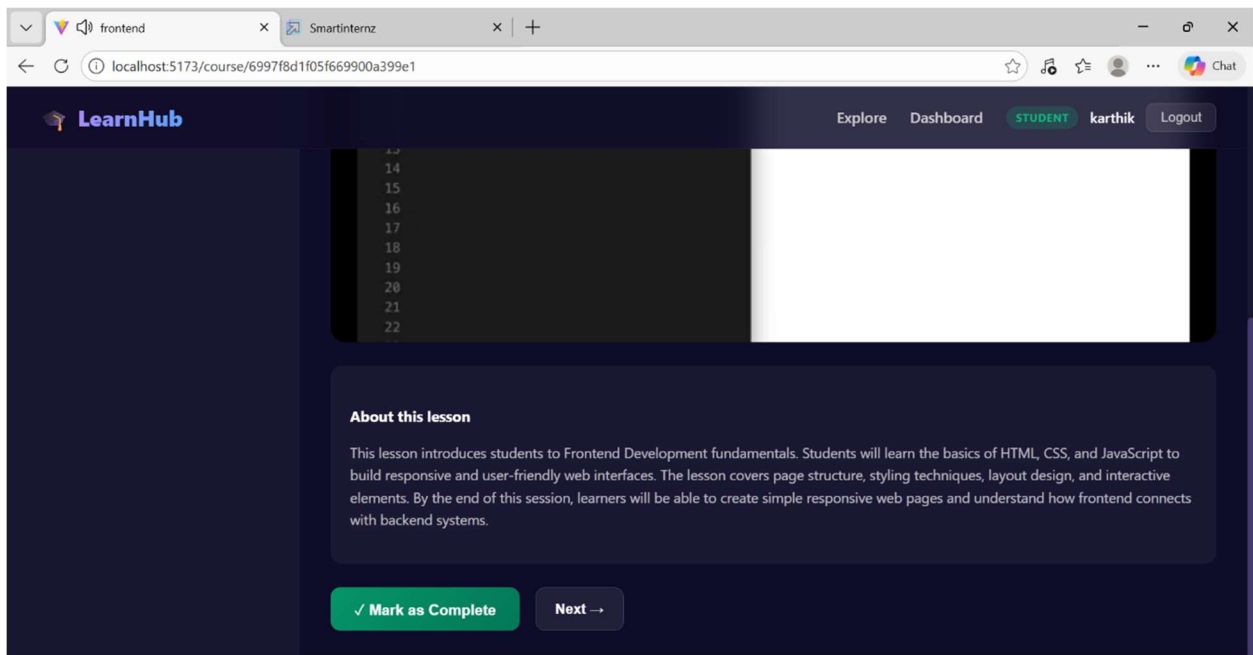


Figure 11: Admin Panel

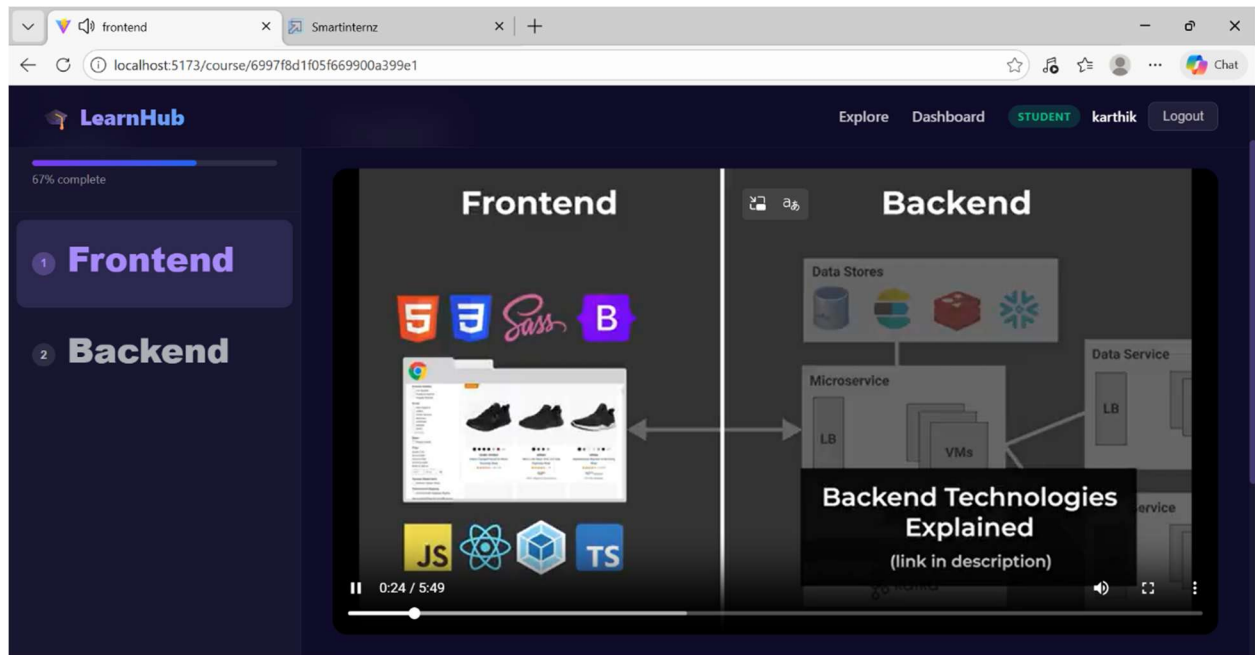


Figure 12: Certificate of Completion

10. Testing

10.1 Testing Approach

The LearnHub application underwent comprehensive manual testing across all modules and user roles. Testing was conducted both during development (unit-level) and after integration (end-to-end) to validate the complete application flow.

10.2 Test Coverage Summary

Test Case	Description	Status
User Registration	POST /api/auth/register with valid/invalid data	✓ Pass
User Login	POST /api/auth/login with correct and incorrect credentials	✓ Pass
JWT Protection	Access protected routes with/without valid token	✓ Pass
Course Creation	Teacher creates course with all required fields	✓ Pass
Course Update	Teacher edits course title, description, and price	✓ Pass
Course Deletion	Teacher deletes course with no enrolled students	✓ Pass
Student Enrollment	Student enrolls in a free and a paid course	✓ Pass
Progress Tracking	Student resumes course from last watched section	✓ Pass
Certificate Download	Student downloads certificate upon 100% completion	✓ Pass
Admin User Management	Admin views, edits, and deactivates user accounts	✓ Pass
Admin Course Management	Admin edits any course on the platform	✓ Pass
Course Search/Filter	Student filters courses by name, category, and price	✓ Pass
Role-Based Access	Students cannot access Teacher-only routes	✓ Pass
Password Hashing	Verify passwords are stored as bcrypt hashes in DB	✓ Pass

11. Known Issues & Future Enhancements

11.1 Known Issues

The following issues were identified during the final testing phase and are acknowledged for future resolution:

Issue ID	Description	Severity	Status
UI-001	Minor CSS alignment issues on mobile screens below 375px width	Low	Open
UI-002	Course card images may not scale uniformly in some Safari versions	Low	Open
BE-001	Large file uploads via Multer may timeout on slow connections	Medium	Open
BE-002	No pagination implemented on the course listing API endpoint	Medium	Planned
DB-001	No database indexing on frequently queried fields (email, category)	Medium	Planned

11.2 Future Enhancements

The following features and improvements are planned for future versions of LearnHub to enhance functionality, scalability, and user experience:

21. Payment Gateway Integration — Integrate Stripe or Razorpay to handle real transactions for paid course purchases securely and reliably.
22. Video Streaming Support — Implement adaptive bitrate video streaming (e.g., using HLS) so students can watch lecture videos directly within the platform without downloading them.
23. Certificate Generation — Build an automated PDF certificate generator using libraries such as PDFKit or jsPDF, populated with the student's name, course title, and completion date.
24. AI-Based Course Recommendations — Leverage machine learning or collaborative filtering to suggest relevant courses to students based on their enrollment history and browsing behavior.
25. Cloud Deployment — Deploy the frontend on Vercel or Netlify and the backend on AWS EC2, DigitalOcean, or Heroku, with the database hosted on MongoDB Atlas.
26. Real-Time Notifications — Add WebSocket-based notifications (using Socket.io) to alert students about new course content and teachers about new enrollments.
27. Discussion Forums — Add per-course discussion boards where students and teachers can interact, ask questions, and share resources.
28. Mobile App — Develop a React Native mobile application that reuses the existing backend API to serve both web and mobile users.
29. Database Indexing & Performance Optimization — Add compound indexes on frequently queried fields and implement server-side pagination for all list endpoints.
30. Automated Testing — Implement Jest and Supertest for backend unit and integration tests, and React Testing Library for frontend component tests.

12. Glossary

Term	Definition
MERN	MongoDB, Express.js, React.js, Node.js — a popular JavaScript full-stack framework combination.
MongoDB	A NoSQL document-oriented database that stores data in flexible, JSON-like BSON format.
Express.js	A lightweight web framework for Node.js used to build RESTful APIs.
React.js	A JavaScript library developed by Facebook for building component-based user interfaces.
Node.js	A server-side JavaScript runtime environment built on Chrome's V8 engine.
Vite	A modern frontend build tool that provides fast development server startup via native ES module support.
JWT	JSON Web Token — a compact, URL-safe token used for stateless authentication.
RBAC	Role-Based Access Control — a method of restricting access based on the user's assigned role.
REST API	Representational State Transfer API — an architectural style for designing networked applications using HTTP.
Mongoose	An ODM (Object-Document Mapper) library for MongoDB and Node.js providing schema-based solutions.
bcryptjs	A JavaScript library used to hash passwords before storing them in the database.
Axios	A promise-based HTTP client for JavaScript used to make API calls from the browser or Node.js.
ODM	Object-Document Mapper — a programming technique that maps application objects to database documents.
HMR	Hot Module Replacement — a feature allowing modules to be updated in a running app without a full page reload.
CRUD	Create, Read, Update, Delete — the four basic operations for persistent data storage.

13.Conclusion:

LearnHub successfully demonstrates the power and versatility of the MERN stack in building a full-featured, role-based Online Learning Platform. From secure JWT authentication and dynamic course management to student progress tracking and certificate generation, the application covers the complete lifecycle of an e-learning experience.

The project achieves its core objectives — enabling teachers to publish content, students to learn at their own pace, and administrators to oversee the platform — all within a clean, responsive, and scalable architecture. Through this development process, key full-stack concepts were applied in practice: RESTful API design, MongoDB schema modeling, React component architecture, middleware-based authorization, and responsive UI design using modern libraries.

While the current version is a solid foundation, the roadmap for future enhancements — including payment integration, video streaming, AI recommendations, and cloud deployment — positions LearnHub as a platform with strong potential for real-world production use.

In summary, LearnHub is not just a project — it is a practical demonstration of how modern JavaScript technologies can be combined to deliver a complete, end-to-end web application that solves a real-world problem in the education space