PROJECT DOCUMENTATION

1. INTRODUCTION

Project title:

LearnHub – Your Center for Skill Enhancement

LearnHub is a full-featured online learning platform developed using the MERN stack (MongoDB, Express.js, React, Node.js). It empowers students, educators, and administrators with seamless course delivery, enrollment, and management capabilities. Designed with a user-centric approach, LearnHub facilitates self-paced learning and teaching with dynamic dashboards for all user roles. It is ideal for institutions, independent educators, and learners seeking to manage and access quality educational content.

Team members:

- Team ID : LTVIP2025TMID22746
- Team Size : 4
- Team Leader : Shaik Amar
- Team member : Shaik Gafoor
- Team member : Saniya Tarunnam
- Team member : Shaik Dharanikota Abdul Gani

2. OVERVIEW

Purpose:

The objective of this project is to design and implement a responsive, scalable, and secure online learning environment. The system allows students to browse and enroll in both free and paid courses, provides teachers with tools to create and manage course content, and equips administrators with analytical insights such as total enrollments and revenue tracking. By enabling an interactive and automated learning system, LearnHub aims to enhance digital education access and management.

Features:

Student Role

- Register and log in to the platform
- Browse courses by filtering (free/paid)
- Enroll in courses (if free or with a price tag)
- View enrolled courses in the student dashboard
- Track course progress and access content
- View course price and description
- Receive digital certification upon course completion (extendable feature)

Teacher Role

- Register and log in as a teacher
- Access a personalized dashboard
- Add new courses with video upload, title, description, and price
- Manage their course listings and delete them as needed
- Track enrolled students count for each course

Admin Role

- Securely access the admin dashboard
- View analytics including:
- Total number of students
- Total number of teachers
- Total number of courses
- Total revenue generated from enrollments
- No approval, moderation, or payment gateway integration involved
- Only statistical display for operational oversight

3. ARCHITECTURE

LearnHub follows a modular client-server architecture with clear separation of concerns

Front end :

This project is developed using **React**, a powerful and flexible JavaScript library for building user interfaces, particularly single-page applications. For styling, it incorporates **Tailwind CSS**, a utility-first CSS framework that allows for rapid UI development with a consistent and responsive design system.

To enhance development speed and performance, the project is bundled using **Vite**, a modern build tool known for its lightning-fast cold starts, hot module replacement, and optimized production builds.

For handling API requests and asynchronous data fetching, the application uses **Axios**, a promise-based HTTP client that simplifies communication with back-end services by offering an intuitive API and built-in error handling capabilities.

Together, these technologies create a modern, responsive, and high-performance web application architecture optimized for both developer productivity and user experience.

Back end :

The backend of the application is developed using **Node.js**, a powerful JavaScript runtime built on Chrome's V8 engine that enables server-side scripting and scalable network applications.

To structure the server-side logic and handle routing efficiently, the project uses **Express.js**, a minimalist and flexible web framework for Node.js. Express facilitates the development of **RESTful APIs** by providing robust tools for managing HTTP requests, middleware, and URL routing.

Together, Node.js and Express.js enable fast, scalable, and maintainable server-side logic, making the application well-suited for real-time interactions and efficient API communication with the frontend.

Database:

The application uses **MongoDB**, a NoSQL document-oriented database, for efficient storage and retrieval of application data. MongoDB's flexible schema design makes it ideal for handling dynamic data structures and scaling horizontally.

To interact with the database, the project utilizes **Mongoose**, an Object Data Modeling (ODM) library for MongoDB and Node.js. Mongoose provides a straightforward, schema-based solution to model application data, enforce validations, and manage relationships between documents.

Layer	Technology
Frontend	React.js, Vite, Tailwind CSS, Bootstrap, Material UI
Backend	Node.js, Express.js
Database	MongoDB with Mongoose
UI Support	Bootstrap, Ant Design, Material UI
Media Upload	Multer (videos)
Auth	JWT Authentication

4. 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/

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 app will be accessible at http://localhost:5172

5. FOLDER 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

6. RUNNING THE APPLICATION

Frontend: cd frontend && npm install

Backend: cd ../backend && npm install

Create .env file in backend:

ini

Copy

Edit

PORT=5000

MONGO_URI=your_mongodb_uri

JWT_SECRET=your_jwt_secret

Run both servers:

Frontend: npm run dev

Backend: npm start

7. API DOCUMENTATION

This section outlines all RESTful API endpoints exposed by the backend of the **LearnHub** platform. Each endpoint includes the method type, endpoint path, required parameters, and example responses.

Authentication APIs

1. **Register API (POST /api/auth/register)**
   This endpoint allows a new user (student or teacher) to register on the platform. The request body must contain essential fields like name, email, password, and role. Upon successful registration, the server responds with a confirmation message and a JSON Web Token (JWT) for authentication.
2. **Login API (POST /api/auth/login)**
   This endpoint is used for user authentication. It validates the email and password combination and returns a JWT token upon successful login. This token is required to access protected routes.

Student APIs

1. **View Courses (GET /api/courses)**
   This endpoint fetches all available courses. Optional query parameters allow filtering

by course type (free or paid). It is accessible to both authenticated and unauthenticated users.

2. **Enroll in Course (POST /api/students/enroll/:courseId)**
   Allows a logged-in student to enroll in a course. The course ID is passed as a route parameter, and the student's identity is determined from the JWT token.
3. **My Enrolled Courses (GET /api/students/courses)**
   Returns a list of all courses in which the logged-in student is currently enrolled. Each course entry may include metadata like title, progress, and completion status.

Teacher APIs

1. **Add New Course (POST /api/teachers/courses)**
   This endpoint allows an authenticated teacher to create a new course. The request includes the course title, description, price, and potentially media assets. The course is then saved to the database under the teacher's profile.
2. **View My Courses (GET /api/teachers/courses)**
   Retrieves a list of all courses created by the logged-in teacher. Each course includes details such as the number of students enrolled.
3. **Delete Course (DELETE /api/teachers/courses/:courseId)**
   Enables a teacher to delete any course they have created by passing the course ID in the URL. Only the course owner can perform this operation.

Admin APIs

1. **Admin Dashboard Data (GET /api/admin/overview)**
   Provides statistical insights including the total number of students, teachers, courses, and revenue generated from paid enrollments. This endpoint is accessible only to users with administrative privileges and requires authentication.

8. AUTHENTICATION

Authentication and authorization are critical components of the **LearnHub** platform to ensure secure access to resources and role-based functionality. The application uses **JWT (JSON Web Token)**-based authentication to manage user sessions and protect sensitive API endpoints.

Authentication Flow

1. **User Registration:**
    - When a new user (student or teacher) registers, their information (name, email, hashed password, and role) is securely stored in the MongoDB database.
    - Passwords are hashed using a secure hashing algorithm (typically **bcrypt**) to prevent plain-text storage.
2. **User Login:**
    - During login, the server verifies the provided credentials.
    - If authentication is successful, a **JWT token** is generated and returned to the client.
    - This token contains encoded information such as the user's ID and role and is signed using a secret key stored in environment variables (JWT_SECRET).
    -

Token-Based Session Management

- The application does **not use server-side sessions or cookies**. Instead, it adopts **stateless authentication** using JWT tokens.
- After login, the frontend stores the token (typically in localStorage or sessionStorage).
- For each subsequent API request to protected routes, the client includes the token in the Authorization header:
- The backend middleware decodes and verifies the token to:
    - Confirm user identity
    - Check the user's role (student, teacher, or admin)
    - Grant or restrict access to certain functionalities accordingly.
    -

Authorization Logic

Once a user is authenticated, **authorization** is handled based on their assigned role:

- **Students** can only enroll in courses, view their progress, and access course content.
- **Teachers** can create, manage, and delete their own courses, and view enrolled student counts.

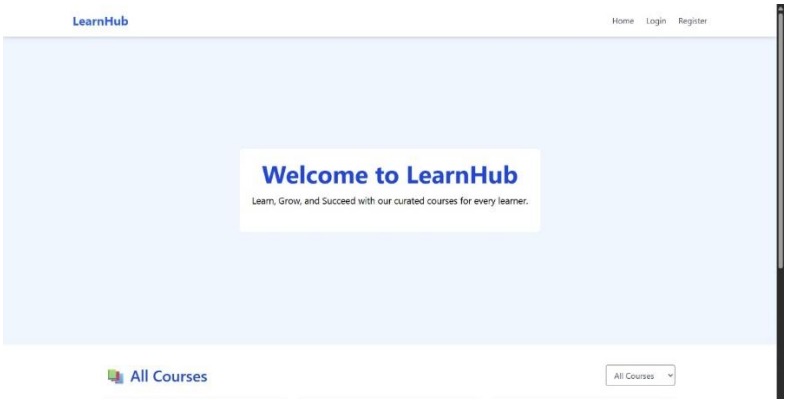
- **Admins** have elevated access to platform-wide data such as total users, total courses, and revenue analytics. Admin routes are strictly protected and only accessible to users marked as admin in the database.

Token Expiry and Security

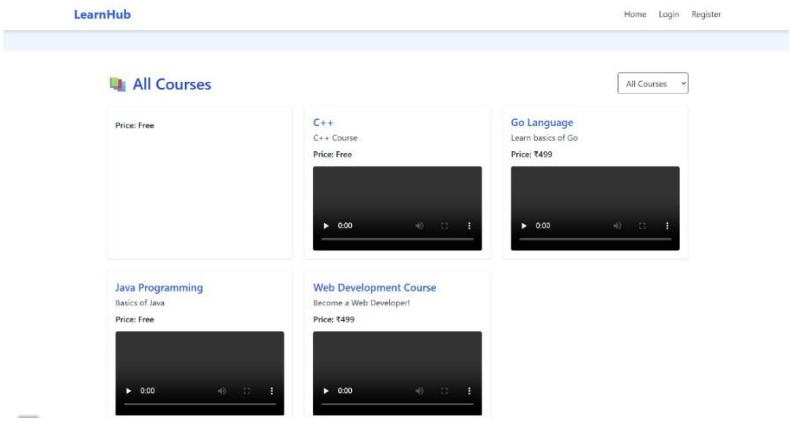
- Each JWT token is generated with a defined **expiry time** (e.g., 1h or 24h) to enhance security.
- Upon expiry, the user must log in again to obtain a new token.
- Middleware functions are used to validate tokens and return appropriate HTTP status codes (401 Unauthorized) when invalid, missing, or expired tokens are encountered.

9. USER INTERFACE

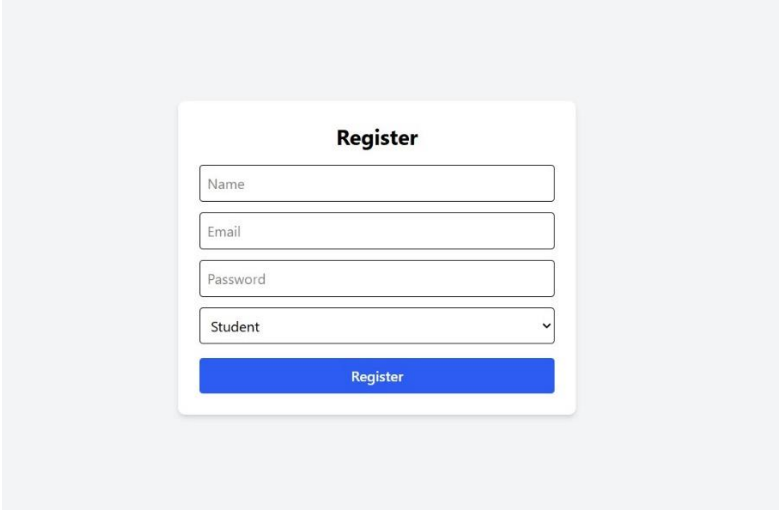
Dashboard



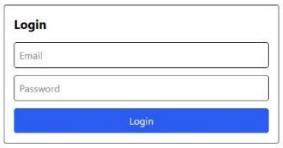
All courses



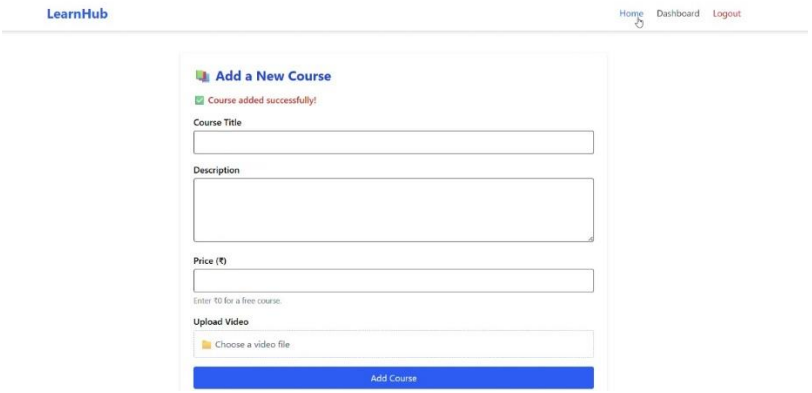
Registration Dashboard



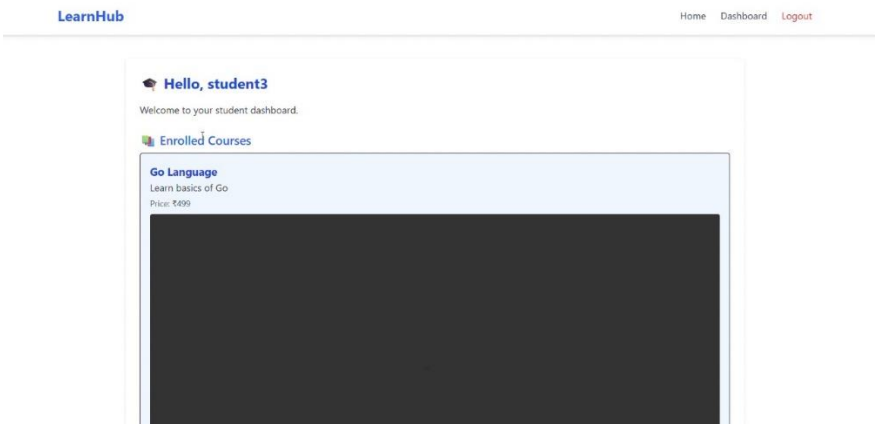
Login Dashboard



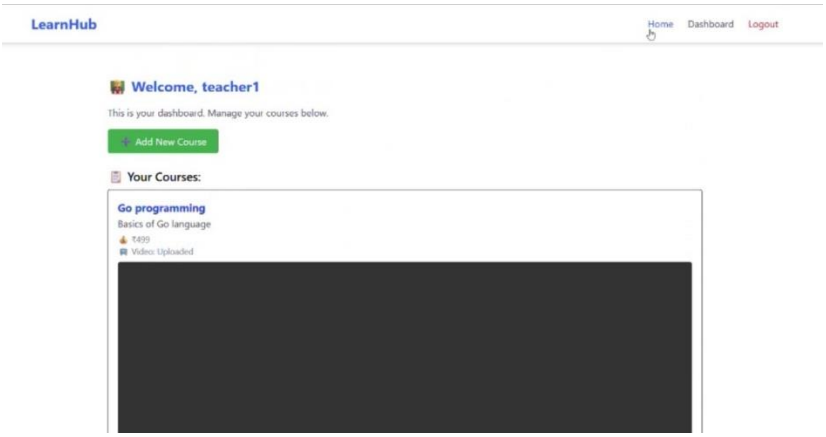
Add new course



Student dashboard



Teacher dashboard



10. TESTING

Testing is an essential phase in the development lifecycle of the LearnHub platform to ensure the correctness, reliability, and security of its functionalities. The application is tested at various levels to validate both the frontend and backend components.

Manual Testing

- All major workflows such as user registration, login, course creation, enrollment, and dashboard rendering were manually tested.
- Multiple user roles (student, teacher, admin) were tested to ensure role-based access control and expected functionality.
- Error scenarios (e.g., invalid login, unauthorized access, duplicate registrations) were manually simulated to confirm proper handling.

API Testing

- Backend endpoints were tested using tools like Postman and Thunder Client.
- Each REST API was verified for:
- Correct request/response format
- HTTP status codes (200, 201, 400, 401, 404, 500)
- Authorization and JWT token validation

Common test cases included:

- Register/Login flows
- Course creation, retrieval, and deletion
- Student course enrollment
- Admin dashboard data retrieval
- Component Testing (Frontend)
- Key React components (e.g., login form, course cards, dashboards) were tested for:
- UI responsiveness across devices
- Correct rendering of dynamic data
- State and form validation

Tools Used

- Postman: For backend API testing and workflow simulation
- Browser Developer Tools: To test frontend responsiveness, layout issues, and error logging
- Console Logging: Used during development to verify data flow and debug backend logic
- *(Optional Future Enhancement)*: Integration of testing frameworks such as Jest (for React) and Mocha/Chai (for Node.js) to automate unit and integration testing.

11.SCREENSHOTS AND DEMO

Demo link :

https://drive.google.com/file/d/1q6X1rG0WTpSlsfoq-y75QU_797gu7oEv/view

12. KNOWN ISSUES

While **LearnHub** provides a functional and stable experience for students, teachers, and administrators, there are a few known limitations and issues that developers and users should be aware of. These issues are planned to be addressed in future updates as the platform evolves.

1. No Payment Gateway Integration

- **Description:** Although paid courses are supported, there is no real payment or transaction processing system integrated.
- **Impact:** Users can "enroll" in paid courses without actual payment validation.
- **Planned Fix:** Integration of secure payment gateways like Razorpay, Stripe, or PayPal in future versions.

2. No Email Verification or Password Reset

- **Description:** Users can register with any email without verification. Additionally, there is no "Forgot Password" feature implemented.
- **Impact:** This reduces account security and may lead to unauthorized account creation or difficulty recovering lost credentials.
- **Planned Fix:** Email verification and password reset functionality via OTP/email link to be introduced.

3. Video Upload Size Limitations

- **Description:** Course video uploads use **Multer**, which may cause performance issues with large file sizes or slow networks.
- **Impact:** Uploading large files may time out or crash the server if not properly configured.
- **Planned Fix:** Use cloud storage solutions like AWS S3 or Cloudinary with streaming support.

4. No Real-time Progress Tracking

- **Description:** Course progress tracking is static or simulated. There is no real-time progress update based on video completion or quiz submissions.
- **Impact:** Students can't accurately track how much of a course they've completed.
- **Planned Fix:** Implement frontend progress tracking tied to actual content consumption (video watch time, quizzes, etc.)

5. Role Switching Not Supported

- **Description:** A user account is permanently locked to a single role (student/teacher). Switching roles requires separate accounts.
- **Impact:** Inconvenient for users who may want to act as both teacher and learner.
- **Planned Fix:** Add role-switching functionality via profile settings.

6. Limited Mobile Responsiveness

- **Description:** While Tailwind CSS provides responsive design, some dashboard components may render improperly on smaller screen sizes.
- **Impact:** Suboptimal experience on mobile devices.
- **Planned Fix:** Conduct thorough responsive testing and optimize key views for mobile compatibility.

7. Admin Moderation is Missing

- **Description:** Course uploads and enrollments do not go through any admin approval or moderation process.
- **Impact:** Potential for low-quality or inappropriate content if not monitored.
- **Planned Fix:** Introduce content moderation workflows and approval dashboards for admins.

13. FUTURE ENHANCEMENTS

To ensure continuous improvement and scalability, the LearnHub platform has been designed with extensibility in mind. Below are several potential enhancements and features that could be implemented in future iterations to enrich user experience, improve functionality, and increase platform utility.

1. Secure Payment Integration

- **Goal:** Enable real-time processing of payments for paid courses.
- **Description:** Integrate payment gateways like **Razorpay**, **Stripe**, or **PayPal** to allow users to make secure transactions.
- **Benefit:** Monetize courses, manage subscriptions, and offer seamless checkout experiences.

2. Email Verification and Password Reset

- **Goal:** Improve user account security and recovery.
- **Description:** Implement features like email-based account verification, OTP login, and password reset through secure email links.
- **Benefit:** Prevent fake registrations and offer account recovery options.

3. Course Progress Tracker

- **Goal:** Enable real-time monitoring of student progress.
- **Description:** Track video views, quiz attempts, and assignment submissions to provide accurate progress percentages.
- **Benefit:** Encourages learner accountability and helps teachers monitor engagement.

4. Quizzes and Assessments

- **Goal:** Make learning more interactive and measurable.
- **Description:** Introduce module-wise quizzes and final course assessments with automatic scoring.
- **Benefit:** Improves learning outcomes and allows teachers to evaluate student understanding.

5. Digital Certificates

- **Goal:** Reward course completion and skill acquisition.
- **Description:** Generate downloadable, verifiable digital certificates upon successful course completion.
- **Benefit:** Adds value to the learner's profile and encourages course completion.

6. Cloud Media Storage

- **Goal:** Improve performance and scalability of media handling.
- **Description:** Shift video and file storage to cloud services like **AWS S3**, **Firebase Storage**, or **Cloudinary**.
- **Benefit:** Reduces server load and allows smooth handling of large media files.

7. Discussion Forums or Chat Modules

- **Goal:** Facilitate interaction between students and teachers.
- **Description:** Add discussion boards or live chat within courses to ask questions and share insights.
- **Benefit:** Builds a community-driven learning environment.

8. Admin Moderation Tools

- **Goal:** Give admins more control over platform content.
- **Description:** Enable course approval, content moderation, user suspension, and report handling features.
- **Benefit:** Maintains content quality and platform integrity.

9. Mobile Application

- **Goal:** Increase accessibility on-the-go.
- **Description:** Develop native or hybrid mobile apps for Android and iOS.
- **Benefit:** Broaden user base and enhance user engagement through push notifications and mobile-first UI.

10. Multi-language Support

- **Goal:** Make the platform accessible to a wider audience.
- **Description:** Implement language selection options and translations for UI and content.
- **Benefit:** Inclusive education for non-English speaking users