Full Stack Development with MERN

Project Documentation format

1. Introduction

Project Title : LearnHUb:Your Center For Skill Enhancement
Team : 1) V.BHANU SRI
- 2)T.V.K.PRADEEPTHI
  3)T.SANTHI KUMARI
  4)U. KRISHNA SIVANI

Team ID:LTVIP2025TMID53135

2. Project Overview

- **Purpose:**LearnHub is a modern, minimalist digital learning platform that enables students to enroll in courses, track progress, and receive certificates, while teachers and admins can manage courses and users.
- **Features:**
  - User authentication (student, teacher, admin roles)
  - Course browsing, filtering, and enrollment
  - Video content delivery with progress tracking
  - Certificate generation and download
  - Admin dashboard for user and course management
  - Responsive, accessible, and cohesive UI

3. Architecture

- **Frontend:**

  - Built with React (Vite, Material-UI)

  - Component-based structure (pages, commonadmin, user, etc.)

  - State management via React Context and hooks

  - Responsive design with modern CSS

- **Backend:**

  - Node.js with Express.js,

  - RESTful API structure,

  - JWT-based authentication and role-based authorization,

  - Multer for file uploads (videos, thumbnails)

- **Database:**

  - MongoDB (Mongoose ODM)

- Schemas for users, courses,

- enrollments, announcements, etc.

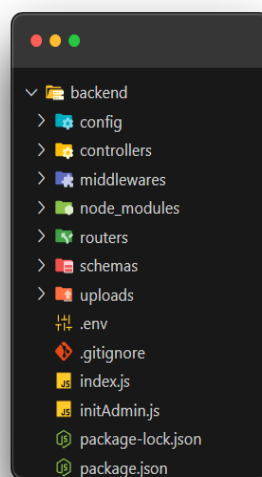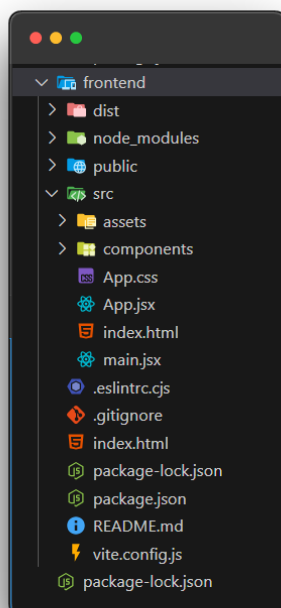- Relationships: Users ↔ Courses ↔ Enrollments

4. Setup Instructions

- **Prerequisites:**

- Node.js (v16+ recommended),

- MongoDB (local or Atlas)

- Git

- **Installation:**git clone [your-repo-url],

- cd code/backend,

- npm install,

- cd ../frontend,

- npm install,

5. Folder Structure

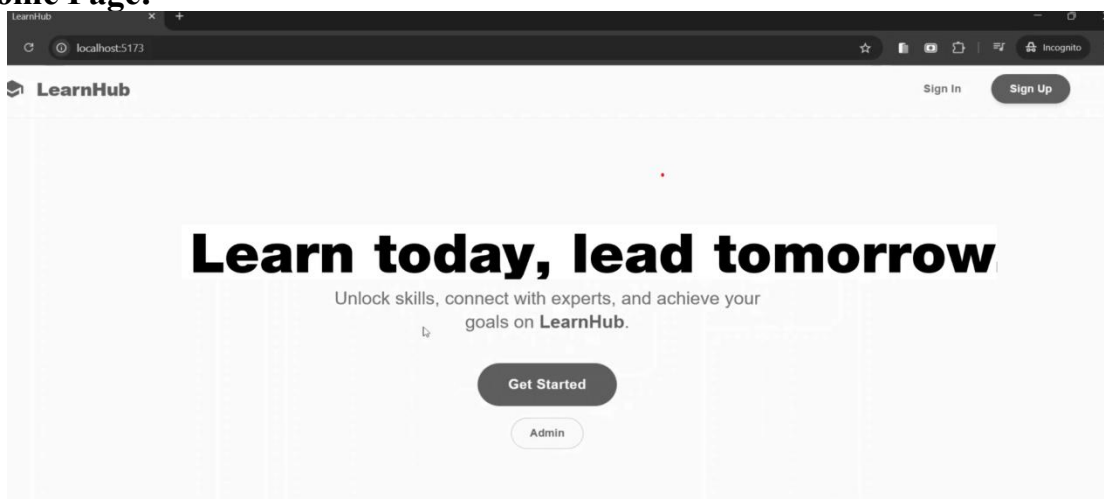- **Client:**   **Server:**

6. API Documentation:

Method	Endpoint	Description	Request Body / Params	Auth Required	Roles Allowed	Response (Success)
POST	/api/auth/register	Register a new user	`{ name, email, password, ... }`	No	Public	User object, token
POST	/api/auth/login	Login user	`{ email, password }`	No	Public	User object, token
GET	/api/users/me	Get current user profile	Token (header)	Yes	User/Admin/Teacher	User object
PUT	/api/users/me	Update current user profile	`{ name, ... }`	Yes	User/Admin/Teacher	Updated user object
GET	/api/courses	List all courses	Query params (optional)	No	Public	Array of courses
GET	/api/courses/:id	Get course details by ID	`:id in URL`	No	Public	Course object
POST	/api/courses	Create a new course	`{ title, description, ... }`	Yes	Teacher/Admin	Created course object
PUT	/api/courses/:id	Edit a course	`{ title, ... }`	Yes	Teacher (own)/Admin	Updated course object
DELETE	/api/courses/:id	Delete a course	`:id in URL`	Yes	Teacher (own)/Admin	Success message

7. Authentication

1. JWT tokens issued on login, stored in localStorage.
2. Role-based access (student, teacher, admin).
3. Protected routes via middleware on backend and route guards on frontend.

8. User Interface:

Home Page:

Registration Form:

9. Testing:

```
// frontend/src/components/common/__tests__/Login.test.jsx
import { render, screen, fireEvent } from '@testing-library/react';
import Login from '../Login';

test('renders login form and submits', () => {
  render(<Login />);
  fireEvent.change(screen.getByLabelText(/email/i), { target: { value: 'test@example.com' } });
  fireEvent.change(screen.getByLabelText(/password/i), { target: { value: 'password' } });
  fireEvent.click(screen.getByRole('button', { name: /login/i }));
  // Add assertions for expected behavior
});
```

```
// backend/tests/course.test.js
const request = require('supertest');
const app = require('../index'); // Your Express app

describe('GET /api/courses', () => {
  it('should return all courses', async () => {
    const res = await request(app).get('/api/courses');
    expect(res.statusCode).toBe(200);
    expect(Array.isArray(res.body)).toBe(true);
  });
});
```

10. Demo

- DEMO LINK -
  https://drive.google.com/file/d/1sjjk_5GQP281vXkWEBuPwU4wwu5XBg2c/view?usp=sharing

11. Known Issues:
- No real payment gateway (demo only)
- UI layout inconsistencies on some small screens
- Certificate print formatting varies by browser

12. FutureEnhancements:
- Integrate real payment gateway (Stripe, Razorpay)
- Add notifications and messaging
- Implement advanced analytics for admins
- Add more media/content types (PDFs, quizzes)
- Improve accessibility and add mo're automated test