

Full Stack Development with MERN

Project Documentation

1. Introduction

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

- **Team Members:** Pasala Rajatkumar - Frontend Developer

Latheesh M - UI/UX Designer

Poliseti Obulesu - Backend Developer

Kuruva Madhu - Tester

2. Project Overview

- **Purpose:** The purpose of this project is to develop LearnHub, a comprehensive online learning management system that bridges the gap between students, teachers, and administrators. The platform aims to provide a centralized hub where users can easily create, manage, and participate in educational courses.

- **Goals of the project include:**

- It makes all the process easy to set up a course, enroll users and provide content anywhere.
- To provide secure authentication and role-based access to the platform, such that students, teachers and admin have their own experiences and unique set of permissions.
- To facilitate the uploading, organizing and sharing of educational content, such as videos and documents by teachers.
- To provide students the ability to register for a course, view resources to study.
- To enable administrators to control users, track platform activity and monitor course quality.
- To have a robust user-friendly interface that will be responsive and function across various platforms.
- To make it scalable and maintainable, enable the platform to expand and change as educational demands evolve.

- **Features:**

- Secure user authentication and registration for students, teachers, and administrators
- Role-based dashboards with tailored access and controls
- Course creation, editing, and management by teachers and admins
- Student enrollment in courses and access to course content
- Media uploads and streaming for course videos and materials
- Progress tracking for students
- Admin panel for managing users, courses, and platform settings
- Responsive and user-friendly interface compatible with multiple devices

- Real-time updates and notifications for important activities
- Data security and privacy for all user information

3. Architecture

- **Frontend:**

We built the frontend in React, so your experience is dynamic and snappy. It is used with a component-based architecture, where different components deal with functionality for teachers, students, and admin. Routing is supported with React Router, and state is managed with react's out of the box hooks and context api. Using libraries such as Bootstrap, Material UI, Ant Design the UI looks more modern. Axios for sending requests to the backend.

- **Backend:**

Backend is written in Node.js and Express.js, following a modular structure. It has 3 different routers for user, admin and course orientated endpoints. Controllers manage business logic, middleware take care of authentication and request validation. The backend provides RESTful APIs for all basic operations including: authentication, course creation and media importing. Security JWT and bcryptjs are used for authentication and password hashing respectively.

- **Database:**

The database is MongoDB and ODM library used is Mongoose. There are tables for users, courses, enrollments, and course payments in the database schema. Every collection has meaningful relationships—courses point to teachers, enrollments point to courses, etc. We use Mongoose models to deal with CRUD operations i.e. read, create, update and delete of data objects, and to ensure constancy and optimal query functionality.

4. Setup Instructions :-

- **Prerequisites:**

- Node.js (v16 or above)
- MongoDB (latest stable version)
- npm (Node Package Manager)
- React (frontend library)
- Express.js (backend framework)
- Mongoose (MongoDB ODM for Node.js)
- Axios (HTTP client for frontend)
- Bootstrap, Material UI, Ant Design, mdb-react-ui-kit (UI libraries)
- bcryptjs (password hashing)
- jsonwebtoken (JWT authentication)
- multer (file uploads)
- dotenv (environment variable management)
- cors (Cross-Origin Resource Sharing)
- nodemon (development server auto-reload)

- **Installation:**

1. **Clone the Repository :**

```
git clone [your-repository-link]
cd [your-project-folder]
```

2. Install Dependencies :

Backend-

```
cd backend  
npm install
```

Frontend-

```
cd frontend  
npm install
```

3. Set Up Environment Variables

- In the backend folder, create a .env file.
- Add the following variables (example values, update as needed):
MONGO_URI=mongodb://localhost:27017/learnhub
JWT_SECRET=your_jwt_secret
PORT=5000
- In the frontend folder, if needed, create a .env file for frontend environment variables (e.g., API base URL):
VITE_API_URL=<http://localhost:5000>

4. Start the Application

Backend-

```
cd backend  
npm start
```

Frontend-

```
cd frontend  
npm run dev
```

5. Folder Structure

• Client (React Frontend)-

```
frontend/  
├─ public/  
├─ src/  
│   ├─ components/  
│   ├─ assets/  
│   ├─ App.jsx  
│   └─ main.jsx  
├─ package.json  
└─ vite.config.js
```

- public/: Contains static files and the main HTML template.
- src/components/: Houses all reusable UI components, organized by user roles.
- src/assets/: Stores images and other static resources.
- App.jsx: The main application component.
- main.jsx: Entry point for rendering the React app.
- package.json: Lists dependencies and scripts for the frontend.
- vite.config.js: Configuration for the Vite build tool.

- **Server (Node.js Backend)-**

```
backend/  
├── config/  
├── controllers/  
├── middlewares/  
├── routers/  
├── schemas/  
├── uploads/  
├── index.js  
└── package.json
```

- config/: Handles database connection setup.
- controllers/: Contains business logic for handling requests.
- middlewares/: Custom middleware for authentication and validation.
- routers/: Defines Express API routes.
- schemas/: Mongoose models for MongoDB collections.
- uploads/: Stores uploaded files such as course videos.
- index.js: Entry point for starting the backend server.
- package.json: Lists dependencies and scripts for the backend.

6. Running the Application

Start the Backend Server-

```
cd backend  
npm start
```

Start the Frontend Server-

```
cd frontend  
npm run dev
```

- The backend server will run at: **http://localhost:5000**
- The frontend development server will run at: **http://localhost:5173**

7. API Documentation

User Authentication

- **POST /api/auth/register**

Description: Register a new user (student, teacher, or admin)

Request Body:

```
{  
  "name": "John Doe",  
  "email": "john@example.com",  
  "password": "password123",  
  "role": "student"  
}
```

Response:

```
{
```

```
    "message": "User registered successfully",
    "user": { ... }
  }
```

- **POST /api/auth/login**

Description: User login

Request Body:

```
{
  "email": "john@example.com",
  "password": "password123"
}
```

Response:

```
{
  "token": "jwt_token_here",
  "user": { ... }
}
```

Courses

- **GET /api/courses**

Description: Get all available courses

Response:

```
[
  {
    "_id": "courseId",
    "title": "Course Title",
    "description": "Course Description",
    "teacher": "teacherId"
  },
  ...
]
```

- **POST /api/courses**

Description: Create a new course (teacher/admin only)

Request Body:

```
{
  "title": "Course Title",
  "description": "Course Description"
}
```

Response:

```
{
  "message": "Course created successfully",
  "course": { ... }
}
```

Enrollment

- **POST /api/enroll**

Description: Enroll a student in a course

Request Body:

```
{
  "courseId": "courseId",
  "studentId": "studentId"
}
```

Response:

```
{
  "message": "Enrollment successful"
}
```

Media Upload

• POST /api/upload

Description: Upload course content (video, etc.)

Request: multipart/form-data with file and courseId

Response:

```
{
  "message": "File uploaded successfully",
  "fileUrl": "/uploads/filename.mp4"
}
```

• POST /api/addcourse

Response:

```
{
  "message": "Course added successfully",
  "course": { ... }
}
```

Get All Courses

GET /api/getallcourses

Response:

```
[
  {
    "_id": "60f1b2c4e1a2b3d4e5f6a7b8",
    "C_title": "Introduction to React",
    "C_description": "Learn the basics of React.",
    "teacher": "60f1b2c4e1a2b3d4e5f6a7b7"
  },
  {
    "_id": "60f1b2c4e1a2b3d4e5f6a7b9",
    "C_title": "Node.js Fundamentals",
    "C_description": "Backend development with Node.js.",
    "teacher": "60f1b2c4e1a2b3d4e5f6a7b6"
  }
]
```

Delete a Course

DELETE /api/deletecourse/:courseId

Response:

```
{
  "message": "Course deleted successfully"
}
```

Add Section to Course

POST /api/addsection/:courseId

Response:

```
{
  "message": "Section added successfully",
  "section": {
    "_id": "60f1b2c4e1a2b3d4e5f6a7c1",
    "title": "Getting Started",
    "content": "Introduction to the course...",
    "videoUrl": "/uploads/section1.mp4"
  }
}
```

Enroll in a Course

POST /api/enrolledcourse/:courseId

Response:

```
{
  "message": "Enrollment successful",
  "enrollment": {
    "_id": "60f1b2c4e1a2b3d4e5f6a7d1",
    "student": "60f1b2c4e1a2b3d4e5f6a7b5",
    "course": "60f1b2c4e1a2b3d4e5f6a7b8"
  }
}
```

Unenroll from a Course

DELETE /api/enrolledcourse/:courseId

Response:

```
{
  "message": "Unenrolled from course successfully"
}
```

Get Course Content

GET /api/coursecontent/:courseId

Response:

```
{
  "courseId": "60f1b2c4e1a2b3d4e5f6a7b8",
  "sections": [
    {
      "title": "Getting Started",
      "content": "Introduction to the course...",
      "videoUrl": "/uploads/section1.mp4"
    },
    {
      "title": "Advanced Topics",
      "content": "Deep dive into React...",
      "videoUrl": "/uploads/section2.mp4"
    }
  ]
}
```

```
}  
}
```

8. Authentication

Authentication and authorization in this project are handled using JSON Web Tokens (JWT).

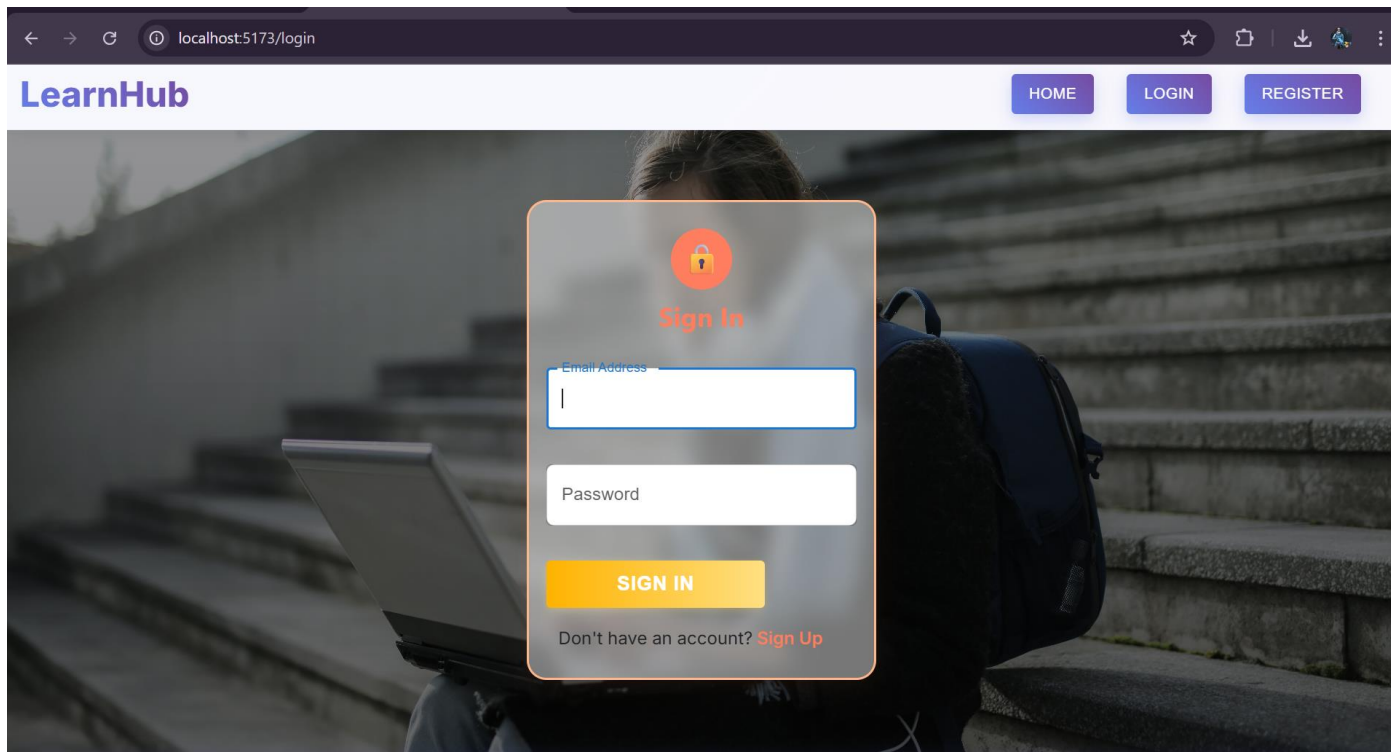
- **User Login:**
When a user logs in, the backend verifies their credentials. If valid, a JWT is generated and sent to the client.
- **Token Usage:**
The JWT contains user information and is stored on the client side (usually in localStorage or as an HTTP-only cookie). For all protected routes, the client includes the token in the Authorization header as a Bearer token.
- **Authorization:**
Middleware on the backend checks the validity of the token for each protected request. It also verifies the user's role (admin, teacher, student) to control access to specific resources and actions.
- **Session Management:**
No server-side sessions are used; authentication is stateless and fully managed via JWTs.
- **Security:**
Passwords are hashed using bcryptjs before storage. Tokens are signed with a secret key and can be set to expire after a certain period for added security.

9. User Interface

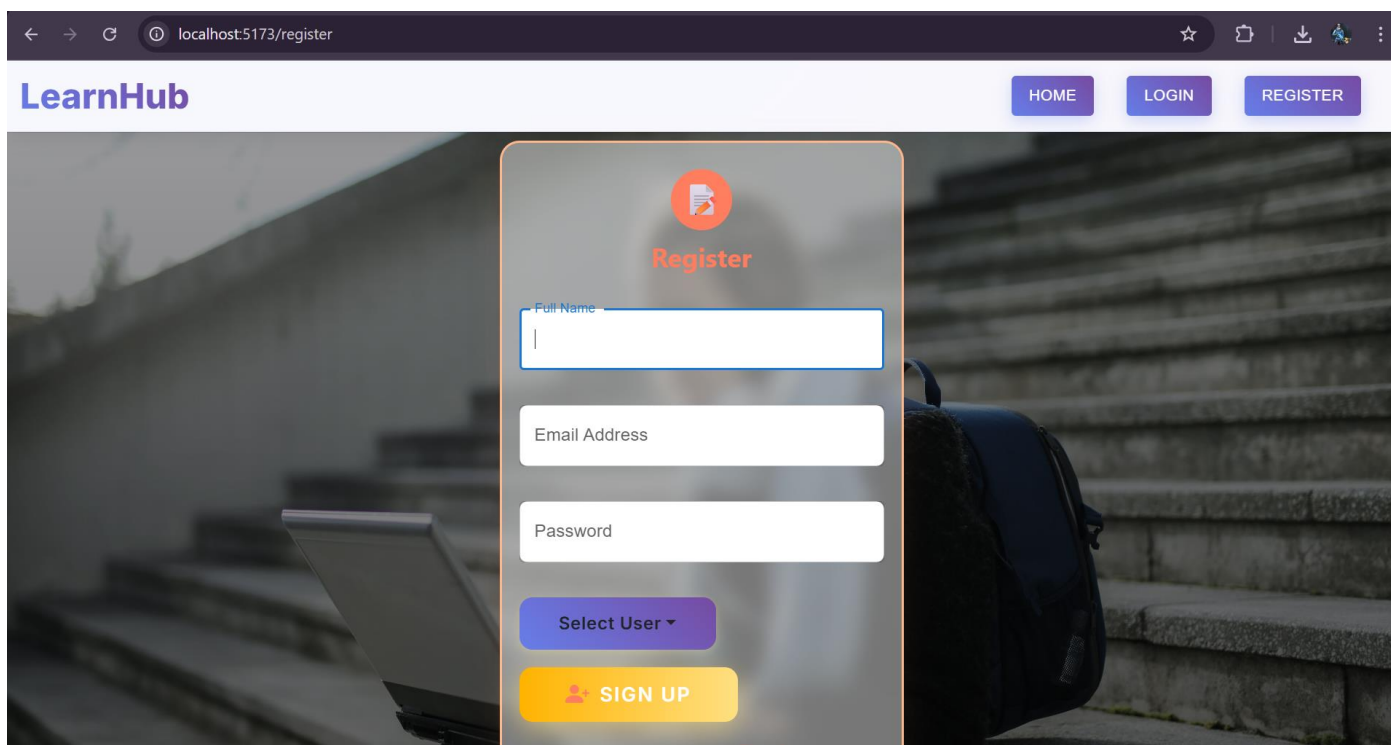
📄 Home Page



📄 Login Page



Registration Page



Admin Dashboard

localhost:5173/dashboard

LearnHub Dashboard Manage Courses

Rajatkumar Admin

Admin Dashboard

4
All Users


2
Teachers

2
Students

0
Courses

0
Enrollments

ALL USERS TEACHERS STUDENTS COURSES ENROLLMENTS





User	Email	Type	Action
 Rajatkumar 685e7175712510dc6931efa4	teacher@gmail.com	Teacher	<button>DELETE</button>

localhost:5173/dashboard

LearnHub Dashboard Manage Courses

Rajatkumar Admin

ALL USERS TEACHERS STUDENTS COURSES ENROLLMENTS

User	Email	Type	Action
 Rajatkumar 685e7175712510dc6931efa4	teacher@gmail.com	Teacher	<button>DELETE</button>
 Dhumbu 685e71a5712510dc6931efa8	dhumbuteacher@gmail.com	Teacher	<button>DELETE</button>
 Rajatkumar 685e71be712510dc6931efab	student@gmail.com	Student	<button>DELETE</button>
 Dhumbu 685e71db712510dc6931efae	dhumbustudent@gmail.com	Student	<button>DELETE</button>

 Teacher Dashboard

localhost:5173/dashboard

LearnHub

Dashboard

Add Course

Rajatkumar

Teacher

Body Building

Description:

Daily food to eat in order to strength o [Read More](#)

Category: Personal Development

Sections: 1

Enrolled students: 0

Add Sec...

Edit Cou...

Delete

How to gain money

Description:

how to save the money and increase the p [Read More](#)

Category: Finance & Accounting

Sections: 1

Enrolled students: 1

Add Section

Edit Course

CSE

Description:

hello welcome to cse section

Category: IT & Software

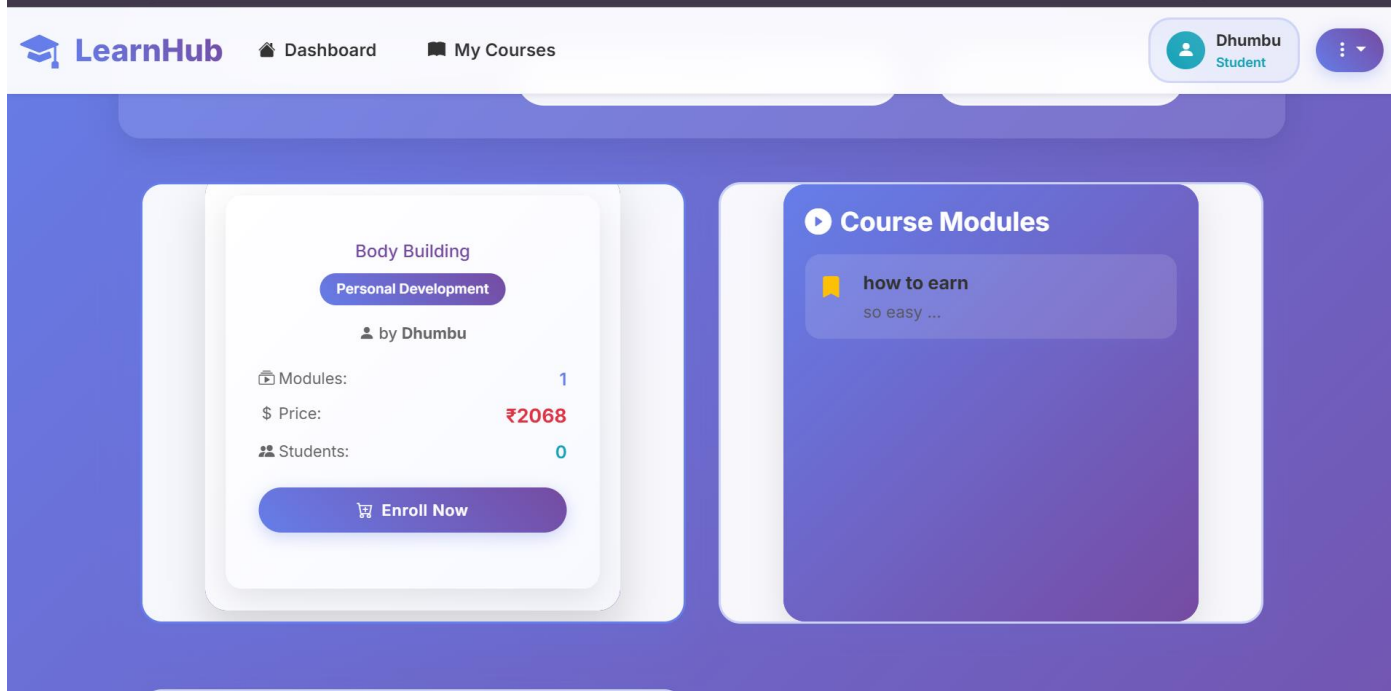
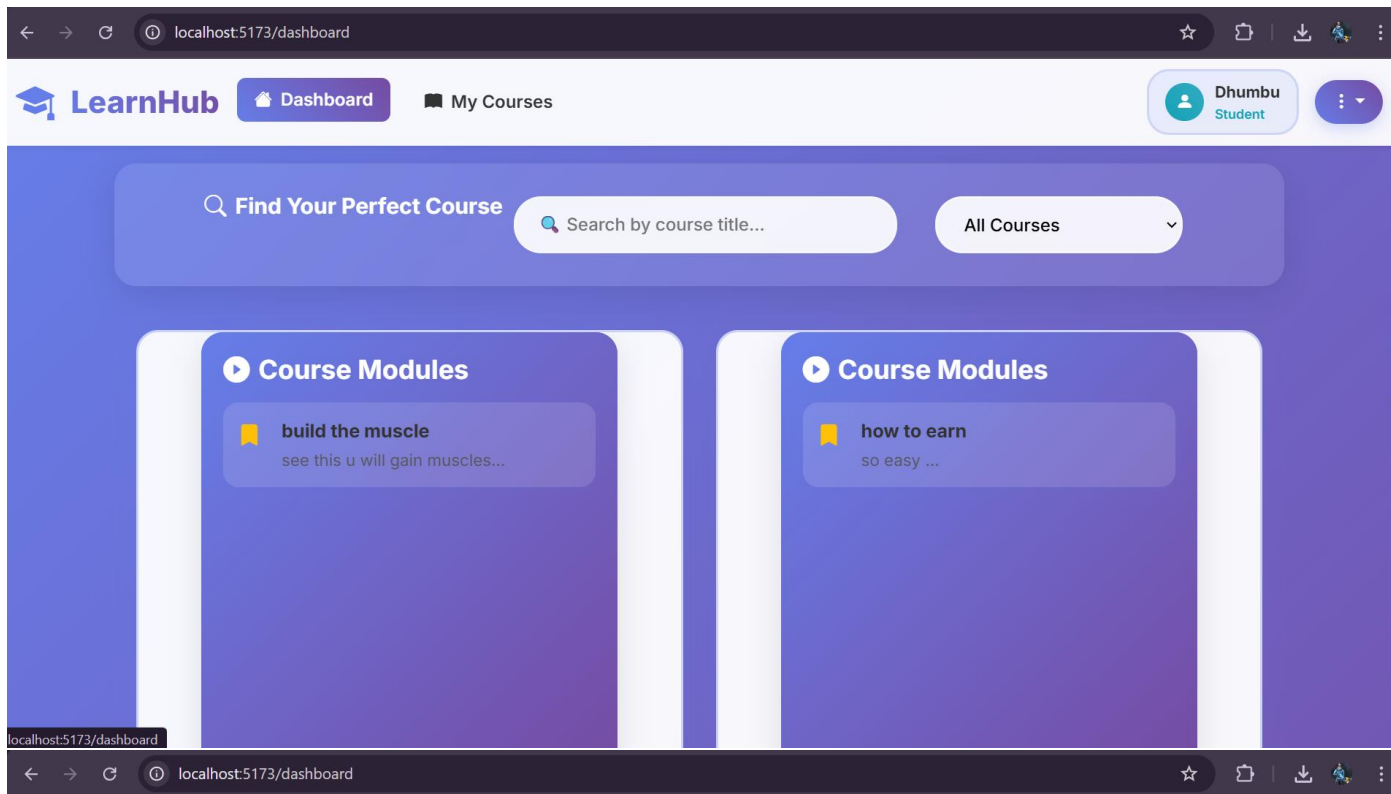
Sections: 1

Enrolled students: 1

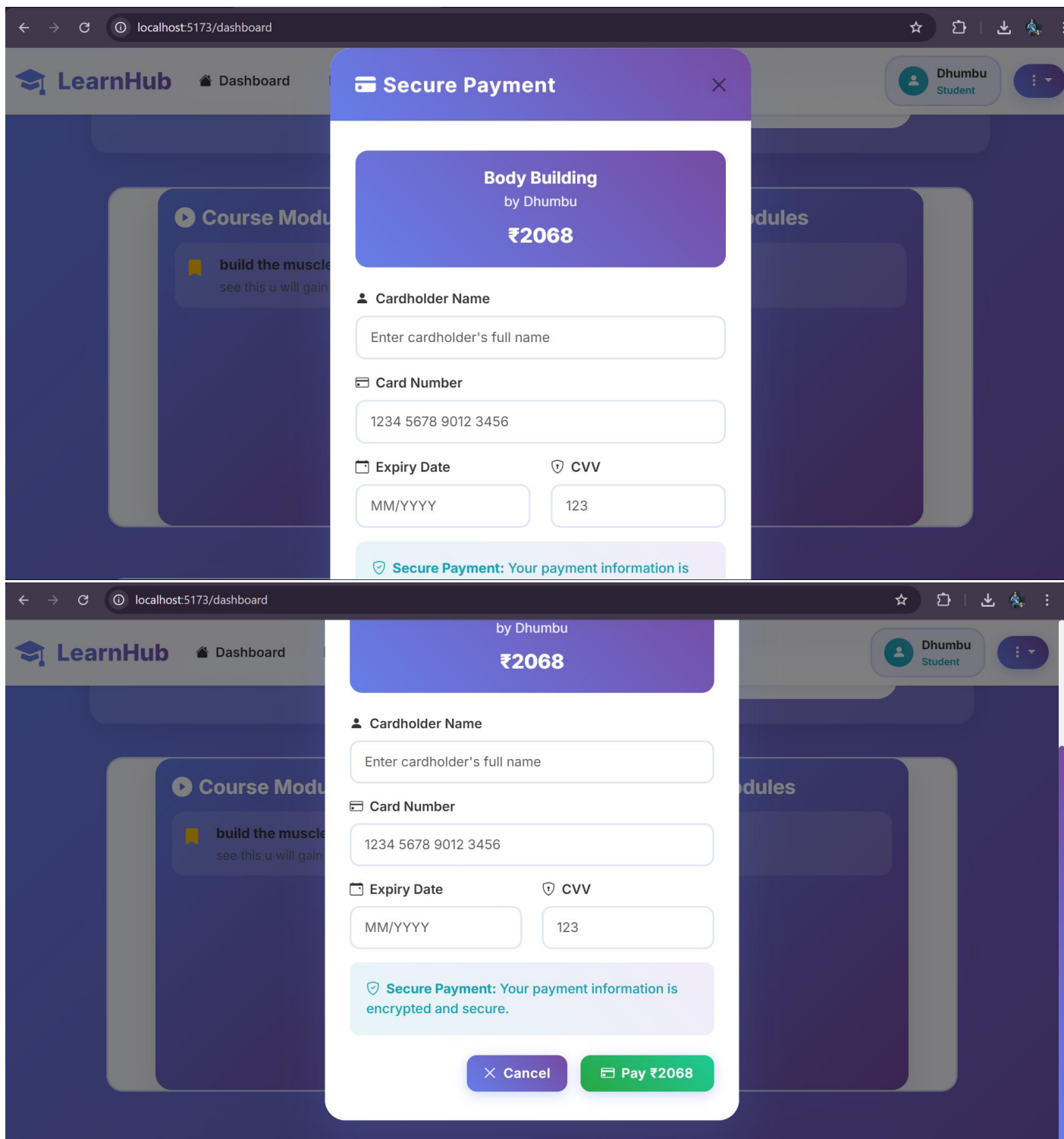
Add Section

Edit Course

Student Dashboard



Payment Page



Course Creation

Course Type

Select categories

Course Title

Enter Course Title

Course Educator

Enter Course Educator

Course Price(Rs.)

for free course, enter (

Course Description

Enter Course
description

 Add Section

Submit

 Course Edit Page

localhost:5173/teacher/edit-course/685e73c5712510dc6931efc1

🔍 ☆ 📄 📥 👤 ⋮

Edit Course

Body Building
Description: Daily food to eat inorder to strength or build the body
Category: Personal Development
Sections: 1
Enrolled students: 0

Course Type

Personal Development

Course Title

Body Building

Course Educator

Dhumbu

Course Price

Rs. 2068

Course Description

Daily food to eat inorder to strength or build the body

Sections

build the muscle
see this u will gain muscles

0:00 / 4:04

🔊 🔍 ⋮

Section Title

build the muscle

Section Description

see this u will gain muscles

Save Changes

Media Upload

localhost:5173/teacher/add-section

Add Section

Section Title

Section Content (Video or Image)

Choose File | No file chosen

Section Description

Add Section

Media Playback

localhost:5173/courseSection/685e746c712510dc6931efc5/How%20to%20gain%20money

← BACK

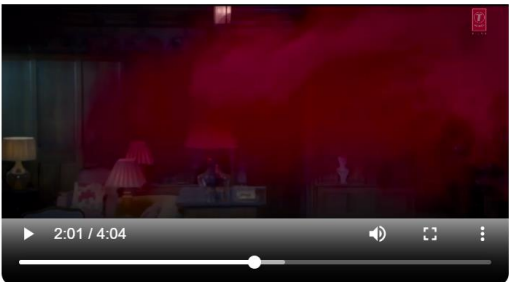
Welcome to the course: How to gain money

how to earn

so easy

PLAY VIDEO

COMPLETED



2:01 / 4:04

PLAY PAUSE STOP

PREVIOUS NEXT

10. Testing

The project uses a combination of manual and automated testing strategies to ensure reliability and correctness.

- **Manual Testing:**
All major user flows (registration, login, course creation, enrollment, media upload, and

dashboard navigation) were manually tested across different browsers and devices to **verify** functionality and responsiveness.

- **Automated Testing:**

Backend APIs can be tested using tools like Postman for endpoint validation and response checking.

- **Error Handling:**

The application was tested for invalid inputs, unauthorized access, and edge cases to ensure proper error messages and security.

- **Performance Testing:**

Key endpoints were checked for response time and stability under typical usage.

11. Demo video

- The Demo video drive link to showcase the application.

[Demo Video link](#)

12. Known Issues

- Occasional delays may occur during large media uploads, depending on network speed.
- No email verification is implemented for new user registrations.
- Error messages may not always be user-friendly or descriptive.
- Mobile responsiveness is good, but some UI elements may require further optimization on smaller screens.
- No password reset functionality is currently available.
- Real-time notifications and chat features are not yet implemented.
- Some edge cases (such as simultaneous course edits) may not be fully handled.

13. Future Enhancements

- Implement email verification and password reset functionality for improved security.
- Add real-time chat and discussion forums for better user interaction.
- Develop a mobile application for iOS and Android platforms.
- Integrate advanced analytics and reporting tools for teachers and admins.
- Enable support for additional media types and interactive content.
- Improve UI/UX for enhanced accessibility and mobile responsiveness.
- Add multi-language support to reach a broader audience.
- Integrate third-party learning tools and APIs.
- Implement real-time notifications for important events and updates.
- Automate testing and continuous integration for more robust development.