LearnHub Your Center for Skill Enhancement

Project Documentation

1.Introduction

- Project title : **LearnHub: Your Center for Skill Enhancement**
- Team leader : Boya Lokeshwari
- Team member : Choutapalle Bhargavi
- Team member : Gajjala suguna
- Team member : Gangarapu Charitha

2.project overview

- Purpose :

  The purpose of the LearnHub Online Learning Platform is to provide a secure, structured, and scalable digital learning environment for students, teachers, and administrators. By leveraging the MERN stack architecture and modern authentication mechanisms, LearnHub enables efficient course management, video-based learning, payment verification, progress tracking, and certificate generation.For students, the platform offers an organized learning path with structured video lessons, enrollment verification, and automated certification upon completion. For teachers, it provides a dedicated dashboard to create and manage courses easily. For administrators, the system ensures controlled access and centralized management through role-based authentication.

  Ultimately, LearnHub bridges technology and education by creating an efficient, accessible, and secure online learning ecosystem that supports digital skill development and knowledge sharing.

- **Features:**

1. Role-Based Authentication System

Key Point: Secure access control
Functionality: Implements JWT-based authentication and bcrypt password hashing to manage Admin, Teacher, and Student roles securely.

2. Video-Based Course Management

Key Point: Structured learning content
Functionality: Allows teachers to upload video lessons organized into sections using Multer for file handling.

3. Course Enrollment with Payment Verification

Key Point: Controlled access to content
Functionality: Ensures students must complete payment simulation before enrolling in a course.

4. Progress Tracking System

Key Point: Learning monitoring
Functionality: Tracks completed sections and automatically updates course completion status.

5. Automated Certificate Generation

Key Point: Proof of course completion
Functionality: Generates downloadable PDF certificates using PDFKit once the course is completed.

6. Dedicated Dashboards

Key Point: Organized workflow
Functionality: Separate dashboards for Admin, Teacher, and Student to manage platform activities efficiently.

7. RESTful API Communication

Key Point: Seamless integration
Functionality: Uses REST APIs and Axios for secure frontend-backend communication.

8. Secure Data Management

Key Point: Reliable data storage
Functionality: Stores user, course, payment, and enrollment data in MongoDB using structured schemas.

9. Responsive User Interface

Key Point: User-friendly experience
Functionality: Developed using React.js and Bootstrap to ensure accessibility across devices.

3. Architecture

Frontend (Stream lit):

The frontend of LearnHub is developed using **React.js**, providing an interactive and dynamic web interface for students, teachers, and administrators. The application includes multiple pages such as Home, Login, Register, Admin Dashboard, Teacher Dashboard, Student Dashboard, Course Listing, Course Content, and Certificate Download. Routing is handled using **React Router**, and API communication is managed through **Axios**. The UI is designed using **Bootstrap and custom CSS**,

ensuring responsiveness and scalability. Each component is modularized to support maintainability and future expansion.

Backend (Fast API):

The backend is built using **Node.js and Express.js**, which serve as the RESTful API framework. It manages user authentication, course creation, payment verification, enrollment handling, progress tracking, and certificate generation. The backend follows a structured MVC-like architecture with separate folders for routes, controllers, middleware, and schemas. Middleware is used for authentication (JWT verification) and role-based access control. The system ensures secure API communication and efficient handling of asynchronous requests.

Authentication & Security (JWT & bcrypt)

Authentication is implemented using JSON Web Tokens (JWT) for secure session management and bcrypt for password hashing. After login, a token is generated and attached to subsequent requests for protected routes. Role-based access control ensures that Admin, Teacher, and Student functionalities are securely separated.

Database (MongoDB & Mongoose)

LearnHub uses MongoDB as its NoSQL database, with Mongoose for schema modeling. Data collections include Users, Courses, CoursePayment, and EnrolledCourse. Relationships between collections are managed using object references. The database efficiently stores structured course data, user records, payment information, and progress tracking details.

File Handling (Multer)

Video uploads for courses are managed using Multer, which stores video files locally in the uploads directory. Each course section contains metadata including title and video URL, enabling structured learning content.

Certificate Generation (PDFKit)

Upon course completion, certificates are dynamically generated using **PDFKit**. The system creates downloadable PDF certificates that include student name, course title, and completion date.

API Communication (REST & Axios)

Communication between frontend and backend is handled using **RESTful APIs**, with Axios used on the frontend for sending and receiving requests. Protected routes require JWT tokens for secure access.

4. Setup Instructions

Prerequisites:

- `Node.js (v16 or later)`
- npm (Node Package Manager)
- MongoDB (Local installation or MongoDB Atlas)
- Git (for cloning repository)
- Internet connection for package installation

Installation Process:

- Clone the repository
- Navigate to the backend directory and install dependencies using npm install
- Navigate to the frontend directory and install dependencies using npm install
- Create a .env file in the backend folder and configure:
  - MongoDB connection string (MONGO_URI)
  - JWT secret key (JWT_SECRET)
  - Server port number
- Start the backend server using npm start or node index.js
- Start the frontend application using npm run dev
- Open the browser at http://localhost:5173
- Register as Admin/Teacher/Student and begin interacting with the platform

5. Folder Structure

Backend Structure

- **config/** – Contains database connection setup (MongoDB connection file).
- **controllers/** – Contains business logic for authentication, course management, payment processing, and enrollment handling.
- **middlewares/** – Includes JWT authentication middleware, role-based access control, and video upload configuration (Multer).

- **routes/** – Defines API endpoints for admin, teacher, and student operations.
- **schemas/** – Contains Mongoose models such as:
  - userModel.js
  - courseModel.js
  - enrolledCourseModel.js
  - coursePaymentModel.js
- **uploads/** – Stores uploaded course video files.
- **index.js / server.js** – Entry point for running the backend server.

Frontend Structure

- **src/components/** – Contains all React components divided into:
  - admin/
  - user/teacher/
  - user/student/
  - common/
- **common/AxiosInstance.js** – Configures Axios for API communication and token handling.
- **App.js** – Main routing file using React Router.
- **index.js** – Entry point of the React application.

6. Running the Application

To start the project:

- Launch the backend server (Node.js & Express) to expose REST API endpoints.
- Start the React frontend application to access the web interface.
- Navigate through pages using the navigation bar and role-based dashboards.
- Register or log in as Admin, Teacher, or Student.
- Teachers can upload courses and videos.

- Students can browse courses, make payment, enroll, watch videos, and download certificates.
- All interactions are real-time and use REST APIs to dynamically update the frontend through Axios.

Frontend (React.js):

The frontend is developed using **React.js**, providing an interactive web interface with multiple pages including Home, Login, Register, Admin Dashboard, Teacher Dashboard, Student Dashboard, Course Listing, and Course Content. Navigation is handled using **React Router**, and API communication is managed using **Axios**. The interface is responsive and modularized into reusable components for scalability and maintainability.

Backend (Node.js & Express.js)

The backend uses Express.js as the REST framework that powers API endpoints for authentication, course management, payment processing, enrollment tracking, progress updates, and certificate generation. It supports asynchronous operations and follows a structured MVC-like architecture with routes, controllers, middleware, and schemas.

7. API Documentation

Backend APIs available include:

POST /users/register – Registers a new user with role selection.

POST /users/login – Authenticates user and returns JWT token.

GET /admin/courses – Retrieves all courses (Admin access).

POST /admin/add-course – Allows teacher to upload a new course with videos.

DELETE /admin/course/:id – Deletes a course (Admin only).

POST /users/pay – Processes payment simulation before enrollment.

POST /users/enroll – Enrolls student in selected course after payment verification.

GET /users/my-courses – Retrieves enrolled courses for the logged-in student.

GET /users/course/:id – Fetches course content for enrolled student.

POST /users/complete-section – Updates progress when a section is completed.

GET /users/certificate/:id – Generates and downloads completion certificate

All endpoints are tested using tools like Postman and verified through frontend integration.

8. Authentication

LearnHub implements secure authentication using:

- JWT (JSON Web Token) for secure session management
- bcrypt for password hashing
- Role-based access control (Admin, Teacher, Student)
- Protected routes using middleware
- Token verification for sensitive operations

Planned enhancements include:

- Refresh tokens for extended sessions
- OAuth-based login (Google/GitHub)
- User activity history tracking

9. User Interface

The user interface is designed to be clean, responsive, and easy to navigate. It includes:

- Navigation bar for easy access
- Role-based dashboards
- Course cards with pricing and educator details
- Structured video player interface
- Progress tracking indicators
- Payment modal interface
- Certificate download button
- Responsive design using Bootstrap

10. Testing

Testing was done in multiple phases:

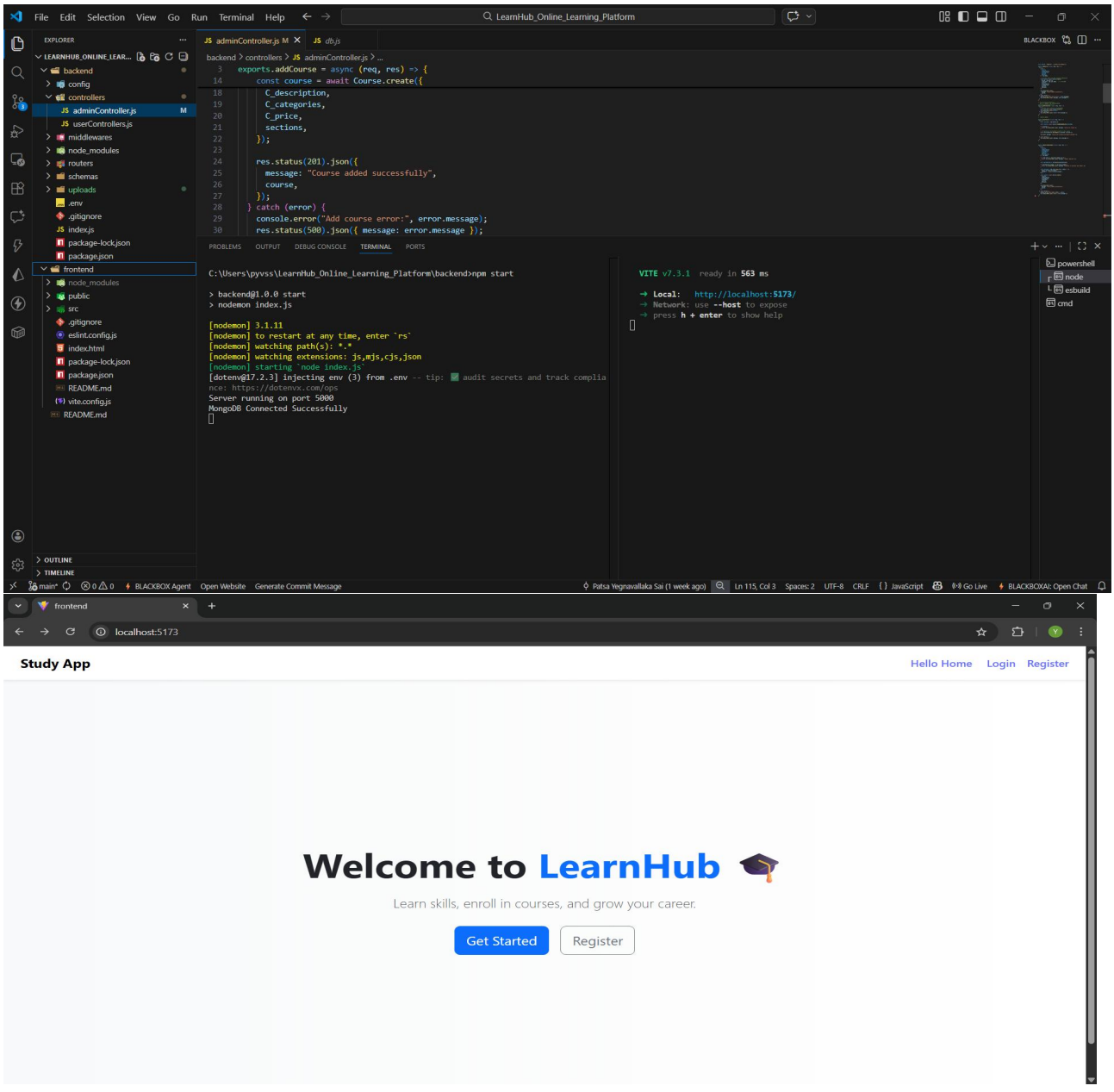
Unit Testing:
Validation of authentication logic, progress tracking, and certificate generation functions.

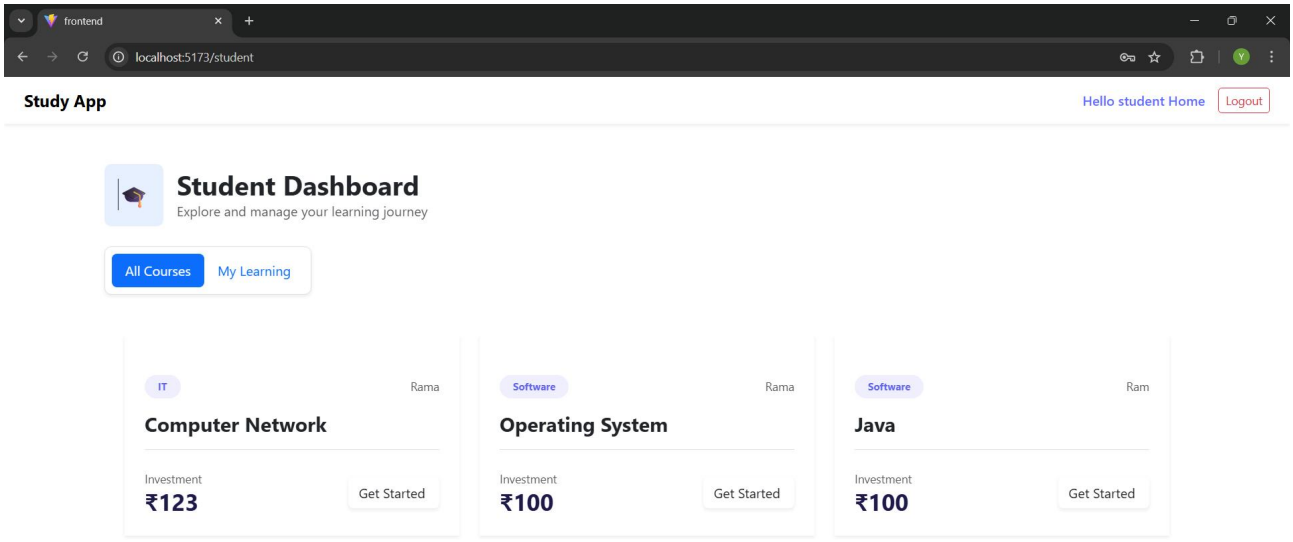
API Testing:
Endpoints tested using Postman to verify correct responses and error handling.

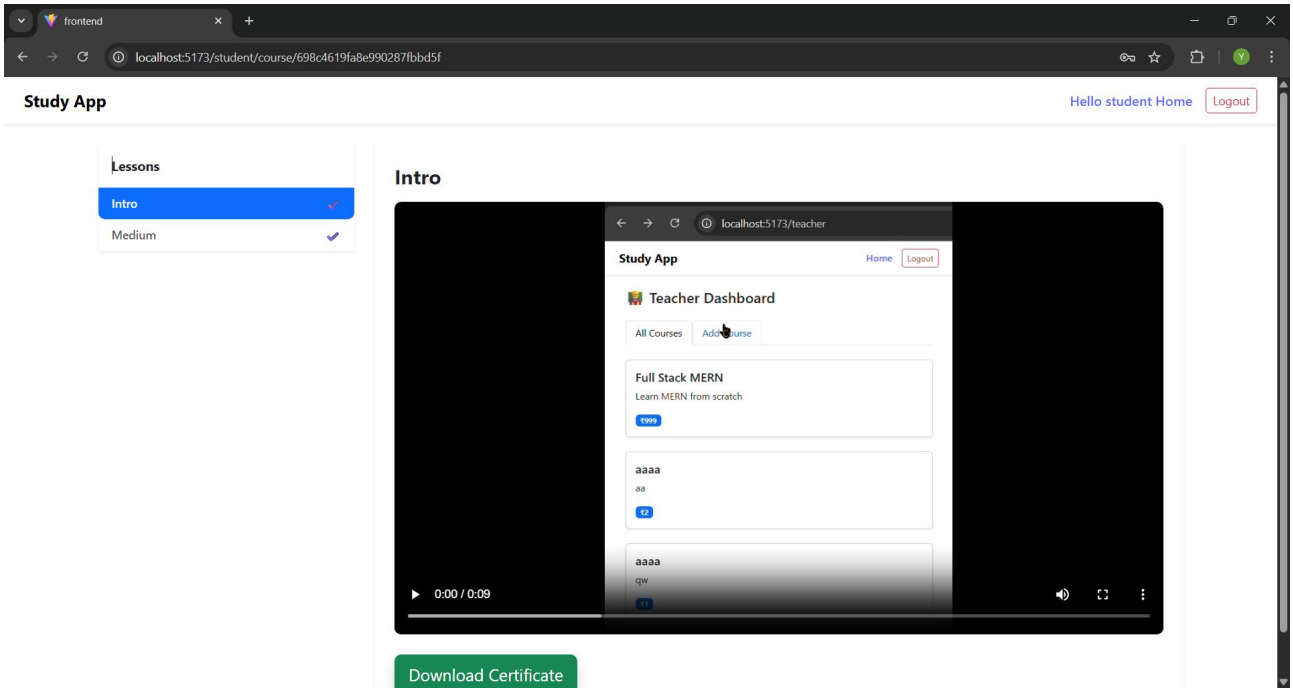
Manual Testing:
Testing user registration, login, course upload, payment simulation, enrollment, video completion, and certificate download.

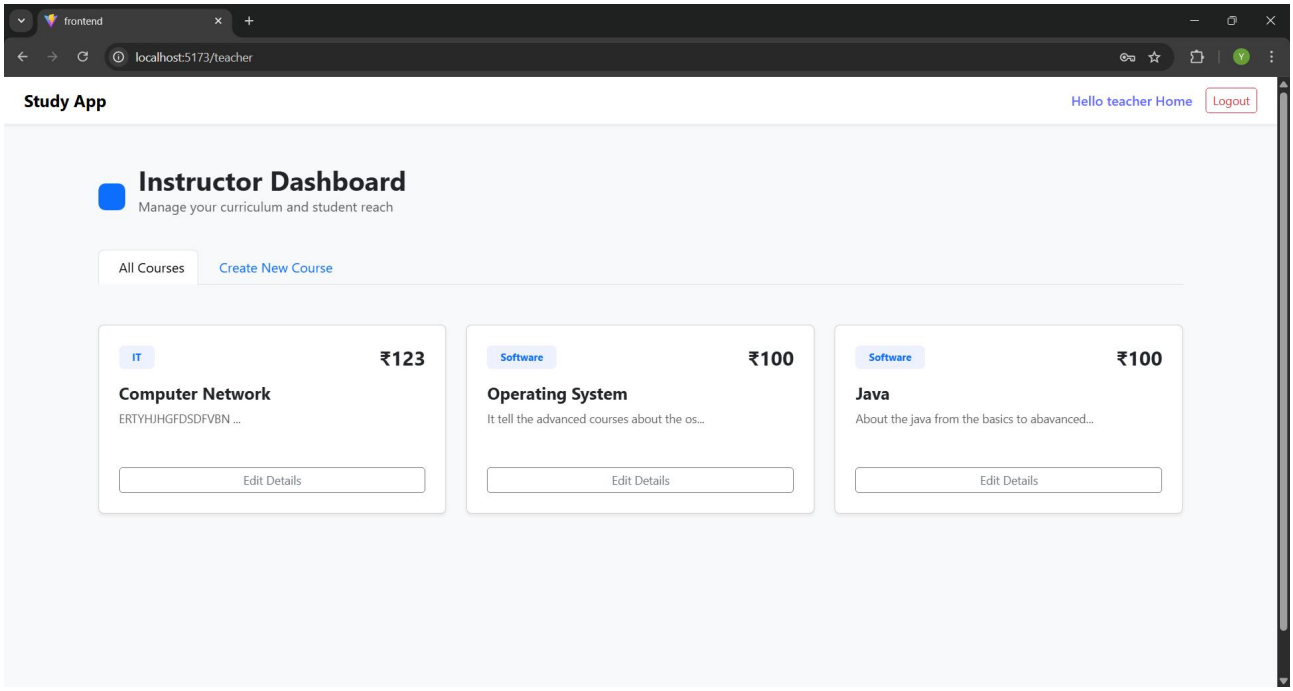
Edge Case Handling:
Handled invalid login credentials, duplicate enrollment attempts, unauthorized access, and missing video uploads.

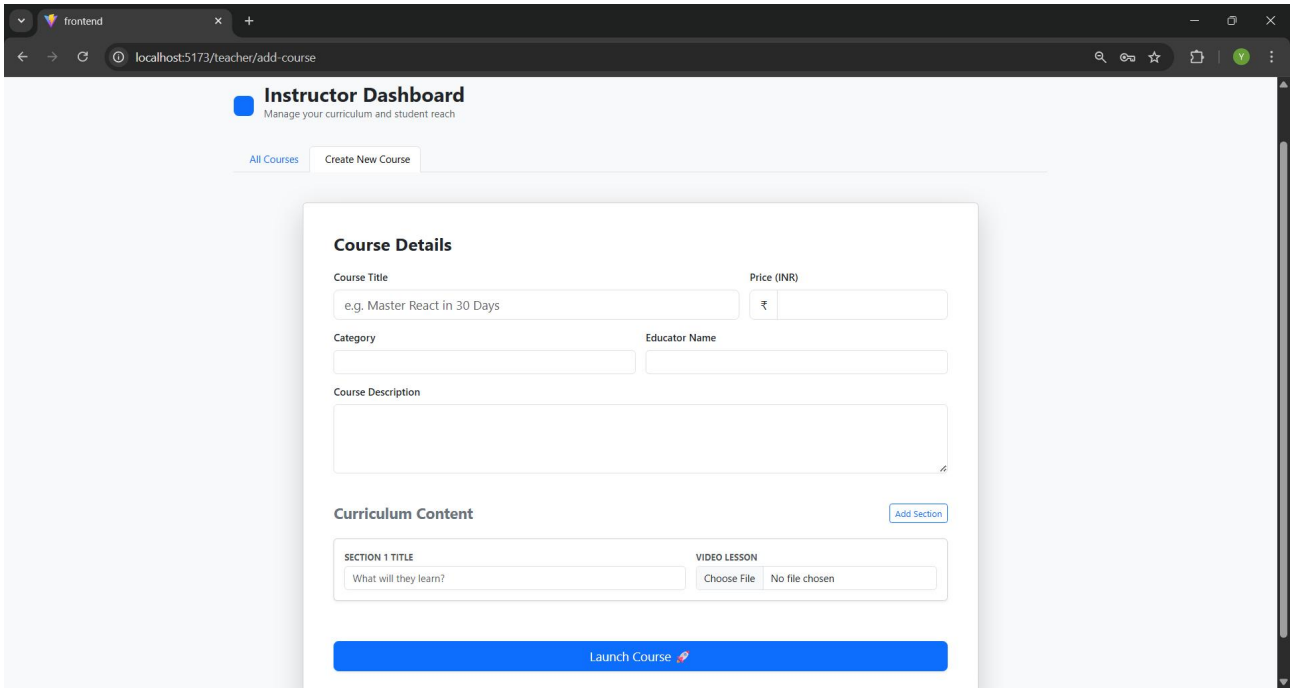
11.Screen Shots




frontend
localhost:5173/student
Study App
Hello student Home
Logout
Student Dashboard
Explore and manage your learning journey
All Courses
My Learning
IT
Rama
Computer Network
Investment
₹123
Get Started
Software
Rama
Operating System
Investment
₹100
Get Started
Software
Ram
Java
Investment
₹100
Get Started


frontend
localhost:5173/student/course/698c4619fa8e990287fbbd5f
Study App
Hello student Home
Logout
Lessons
Intro
Medium
Intro
localhost:5173/teacher
Study App
Home
Logout
Teacher Dashboard
All Courses
Full Stack MERN
Learn MERN from scratch
₹999
aaaa
aa
₹2
aaaa
qw
₹1
0:00 / 0:09
Download Certificate





12. Known Issues

Video files are stored locally, which may cause storage limitations in large-scale deployments.

Payment system is currently simulated and does not integrate with real payment gateways.

No cloud storage integration for uploaded course videos (e.g., AWS S3 or Google Cloud).

Limited scalability in demo/local environment for handling large concurrent user traffic.

No refresh token mechanism implemented for long session management.

Lack of advanced input validation for certain form fields (e.g., payment card format validation).

No real-time notification system for course updates or enrollment confirmation.

Performance may reduce with large datasets due to non-optimized indexing in MongoDB.

No AI-based recommendation system for personalized course suggestions.

Requires manual deployment configuration for production hosting.

13. Future enhancement

• **Adaptive Learning Recommendation System**
Functionality: Implement machine learning models to continuously improve course recommendations based on user behavior, learning patterns, course completion history, and trending topics.

• **Multilingual Platform Support**
Functionality: Add multiple language options to make the platform accessible to diverse learners across different regions.

• **Real-Time Live Classes Integration**
Functionality: Integrate WebRTC or video conferencing tools to support live lectures, interactive sessions, and real-time teacher-student communication.

• **Real Payment Gateway Integration**
Functionality: Replace payment simulation with secure integration of Razorpay, Stripe, or PayPal for real transactions.

• **Cloud-Based Video Storage**
Functionality: Store course videos on AWS S3 or Google Cloud Storage to improve scalability and performance.

• **Advanced Analytics Dashboard**
Functionality: Provide teachers and admins with detailed analytics including student performance trends, course engagement statistics, and revenue insights.

• **Discussion Forums & Chat System**
Functionality: Add real-time chat and discussion boards to promote collaborative learning among students.

• **Docker-Based Deployment & CI/CD**
Functionality: Use Docker containers and GitHub Actions for automated deployment, scalability, and continuous integration.

• **Long-Term Learning Trend Visualization**
Functionality: Store and visualize student performance trends over months or years for academic analysis.

• **AI-Based Assessment & Auto-Grading**
Functionality: Implement automated quizzes, assignment evaluation, and grading systems using AI.

• **Mobile Application Development**
Functionality: Develop Android and iOS mobile apps to increase accessibility and user engagement.

• **Enhanced Input Validation & Security Improvements**
Functionality: Add stronger schema validation, rate limiting, and enhanced session management.