Full Stack Development with MERN

LearnHub – Your Center for Skill Enhancement

1. Introduction

Project Title: LearnHub – Your Center for Skill Enhancement

Team Members:

1. Bharath Munakala – Backend Developer (API, database, server logic)
2. Gudari Devadasu – Frontend Developer (UI design, React components)
3. Akshay Kumar Kodi – Full Stack Support (integration, testing, bug fixing)
4. Dude Sri Venkata Sai Hemasundar – Documentation & Deployment (reports,hosting)

2. Project Overview

Purpose:

LearnHub is designed to be a centralized online learning platform that enables users to acquire and improve technical and professional skills from anywhere. The project aims to simplify the learning process by providing structured courses, interactive content, and progress tracking in one place. The main goal is to create a user-friendly system that bridges the gap between learners and quality educational resources while promoting self-paced learning.

Goals of the Project:

- To provide an accessible digital learning environment for students and beginners
- To allow users to explore courses based on their interests and skill levels
- To support continuous learning through structured modules and progress monitoring
- To help administrators easily manage course content and users
- To build a scalable full-stack web application using modern technologies

Key Features & Functionalities:

- **User Authentication:** Secure signup and login system for learners and admins
- **Course Management:** Display of courses with descriptions, categories, and duration
- **Learning Modules:** Lessons may include videos, notes, or tutorials
- **Progress Tracking:** Users can monitor completed lessons and course progress
- **Admin Dashboard:** Admin can add, update, or delete courses and manage users
- **Search & Filter Options:** Users can quickly find courses by category or keywords
- **Responsive UI:** Works smoothly on mobile, tablet, and desktop devices

Expected Benefits:

- Encourages skill development and self-learning
- Saves time by organizing learning resources in one platform
- Helps beginners follow a structured path instead of random tutorials
- Provides practical exposure to full-stack development concepts

3. Architecture

The LearnHub system follows a **three-tier architecture** consisting of the frontend, backend, and database layers. This architecture ensures separation of concerns, scalability, and maintainability of the application.

3.1 Frontend Architecture (React.js)

The frontend is developed using React.js, which follows a **component-based architecture**. In this approach, the entire user interface is divided into independent and reusable components. Each component manages its own structure, styling, and logic, making the application modular and easier to maintain.

Component Structure :

The frontend is organized into multiple components such as:

- **Layout Components:** Navbar, Footer, Sidebar
- **User Components:** Login, Register, Profile, Dashboard
- **Course Components:** Course List, Course Details, Video Player
- **Admin Components:** Add Course, Edit Course, Manage Users

These components are grouped into folders for better organization and scalability.

Routing & Navigation :

React Router is used to manage navigation between pages such as:

- Home page
- Login/Register page
- Course page
- User dashboard
- Admin dashboard

This allows smooth page transitions without refreshing the browser.

API Communication

The frontend communicates with the backend using HTTP requests via Axios or Fetch API. These requests allow the frontend to:

- Send login credentials
- Fetch course data
- Submit user progress And Load personalized dashboards

3.2 Backend Architecture (Node.js & Express.js)

The backend is built using Node.js with Express.js and follows a **layered MVC-like structure**.

Backend Layers

1. Routes Layer :

Defines API endpoints such as:

- /api/auth/login
- /api/courses
- /api/users
- /api/progress

Routes receive requests from the frontend and forward them to controllers.

2. Controller Layer :

Controllers contain the application logic.

They process requests, validate input, and call database operations.

For example:

- Login controller verifies user credentials
- Course controller fetches course details
- Admin controller handles adding/editing courses

3. Middleware Layer :

Middleware functions are used for:

- Authentication verification (JWT token checking)
- Error handling
- Request logging
- Security checks

This ensures only authorized users access protected routes.

4. Server Layer :

The Express server handles:

- API request processing
- JSON parsing
- Cross-origin resource sharing (CORS)
- Connecting to the database

3.3 Database Architecture (MongoDB)

MongoDB is used as the database because it stores data in flexible, JSON-like documents, making it ideal for web applications.

Collections Used

1. Users Collection :

Stores user information such as:

- Name
- Email
- Password (hashed)
- Role (student/admin)
- Enrolled courses

2. Courses Collection :

Stores course details including:

- Course title
- Description
- Category
- Instructor name
- Video/content links
- Duration

3. Progress Collection :

Tracks learning activity:

- User ID
- Course ID
- Completed lessons
- Progress percentage

Schema Management :

Mongoose is used to define schemas and models for each collection.

This provides:

- Data validation
- Relationships between documents
- Easy querying and updates

4. Setup Instructions

1. Prerequisites

Before running the LearnHub project, ensure the following software and tools are installed on your system:

- **Node.js** (v16 or later recommended) – Required to run the backend server and install packages
- **npm** – Comes with Node.js for managing dependencies
- **MongoDB** (local installation or MongoDB Atlas cloud account) – Used as the database
- **Git** – Required to clone the project repository
- Code editor such as **Visual Studio Code**
- Web browser like Chrome/Edge for testing

2. Installation Steps

Step 1: Clone the Repository

Open terminal or Git Bash and run:

git clone https://github.com/your-username/learnhub.git

cd learnhub

Step 2: Install Backend Dependencies

Navigate to backend folder:

cd backend

npm install

This installs required backend packages such as Express, Mongoose, JWT, and dotenv.

Step 3: Install Frontend Dependencies

Open a new terminal and navigate to frontend folder:

cd frontend

npm install

This installs React, Axios, Router, and UI libraries.

Step 4: Configure Environment Variables

Inside the **backend** folder, create a .env file and add:

PORT=5000

MONGO_URI=your_mongodb_connection_string

JWT_SECRET=your_secret_key

If using MongoDB Atlas, copy the connection string from your cluster and paste it as MONGO_URI.

Step 5: Run the Backend Server

Inside backend folder:

npm run dev

Server will start on:

http://localhost:5000

Step 6: Run the Frontend Application

Inside frontend folder:

npm start

Application will open at:

http://localhost:3000

3. Verification

After setup:

- Open browser at http://localhost:3000
- Register a new user
- Login and verify courses load from database
- Confirm backend API is responding

If all steps work, the LearnHub platform is successfully running locally.

5. Folder Structure

The LearnHub project is organized into two main parts: the **client (frontend)** and the **server (backend)**. This separation improves maintainability, scalability, and team collaboration.

1. Frontend Folder (React Frontend)

The client folder contains the user interface of the application. It follows a component-based structure for better reusability and organization.

Typical Client Structure

frontend/

```
│
├── public/              → Static files like index.html, favicon, images
│
├── src/
│   ├── assets/          → Images, icons, fonts
│   ├── components/      → Reusable UI components (Home, Login, Courses ,Navbar, Footer, Cards)
│   ├── context/         → Global state management (AuthContext etc.)
│   ├── styles/          → CSS or Tailwind styling files
│   ├── App.js           → Main component controlling routes
│   ├── index.js         → Entry point of React application
│
├── package.json         → Frontend dependencies and scripts
```

Explanation

- **components/** contains reusable UI blocks used across pages
- **pages/** holds full-screen views mapped to routes
- **services/** stores API communication logic with backend
- **context/** manages global user state like login session
- **App.js** defines navigation and routing

This structure ensures the UI is modular and easy to expand with new features.

2. Backend Folder (Node.js)

The server folder contains the backend logic, APIs, and database interactions. It follows a layered architecture to separate concerns.

Typical Server Structure

backend/

```
│
├── config/              → Database connection configuration
│   └── db.js
│
├── schemas/             → Database schemas (User, Course, Progress)
│
├── controllers/         → Business logic for handling requests
│
├── routes/              → API route definitions
│
├── middleware/          → Authentication, error handling, security checks
│
├── utils/               → Helper functions (token generation, validators)
│
├── .env                 → Environment variables
├── server.js            → Entry point of backend application
├── package.json         → Backend dependencies and scripts
```

Explanation

- **models/** define MongoDB schemas using Mongoose
- **controllers/** contain functions that process user requests
- **routes/** map URLs to controller functions
- **middleware/** handles authentication, logging, and error responses
- **config/db.js** establishes the database connection
- **server.js** starts the Express server and loads routes

This structure keeps the backend organized and allows easy addition of new APIs or features.

6. Running the Application

To run the LearnHub project locally, both the frontend and backend servers must be started separately. Ensure dependencies are installed and environment variables are configured before running the application.

Start the Backend Server

1. Open a terminal
2. Navigate to the server folder : cd backend
3. Start the backend : npm start

4 .The backend server will run on : http://localhost:5000

This server handles APIs, authentication, and database communication.

Start the Frontend Application

1. Open another terminal
2. Navigate to the client folder : cd frontend
3. Start the React application : npm start
4. The frontend will run on : http://localhost:3000

This opens the LearnHub interface in your browser.

Important Notes

- Backend must run before frontend to fetch data properly
- Ensure MongoDB connection is active
- If ports are busy, change them in .env or package configuration

7. API Documentation

The LearnHub backend exposes RESTful APIs that allow the frontend to perform authentication, course management, and progress tracking. All APIs use JSON for requests and responses.

Base URL: http://localhost:5000/api

1. Authentication APIs

1.POST /auth/register

Register a new user.

Parameters: name, email, password.

Response: success message with authentication token

2.POST /auth/login

Authenticates user credentials.

Parameters: email, password.

Response: JWT token and user details

2. Course APIs

1.GET/courses
Retrieves the list of all available courses.

2.GET/courses/:id
Fetches details of a specific course by ID.

3.POST/courses *(Adminonly)*

Adds a new course**.**

Parameters : title, category, description, duration .

4. **PUT /courses/:id**

Updates an existing course.

5 .DELETE /courses/:id

Removes a course from the system.

3. Progress APIs

1. **POST/progress/enroll**
   Enrolls a user in a course.
   Parameters: userId, courseId
2. **PUT /progress/update**

   Updates course completion progress.

   Parameters: userId, courseId, progress percentage

8. Authentication

Authentication and authorization in the LearnHub system ensure that only valid users can access the platform and that protected resources are available only to authorized roles such as admins.

User Authentication

LearnHub uses **JWT (JSON Web Token)–based authentication**.

1. When a new user registers, their details (name, email, password) are stored in the database.
2. The password is encrypted before saving to ensure security.
3. During login, the server verifies the user's credentials.
4. If valid, the backend generates a JWT token and sends it to the client.

This token acts as the user's identity for future requests.

Token Handling

- The token is stored on the client side (usually in local storage).
- For every protected API request, the token is sent in the request header:

Authorization: Bearer <token>

- The backend middleware verifies the token before allowing access to secure routes.

If the token is invalid or expired, access is denied and the user must log in again.

Authorization

Authorization controls what actions a user can perform.

- **Students** can view courses, enroll, and track progress.
- **Admins** have additional privileges such as adding, updating, or deleting courses and managing users.

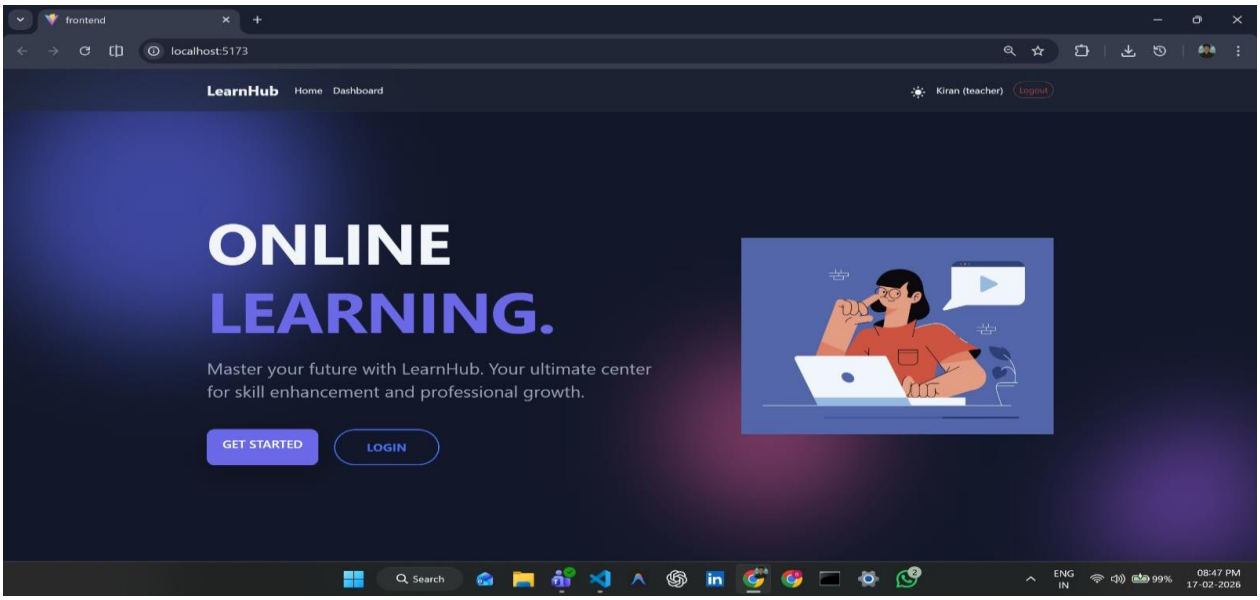
The backend checks the user's role stored in the token to determine permissions before executing protected operations.

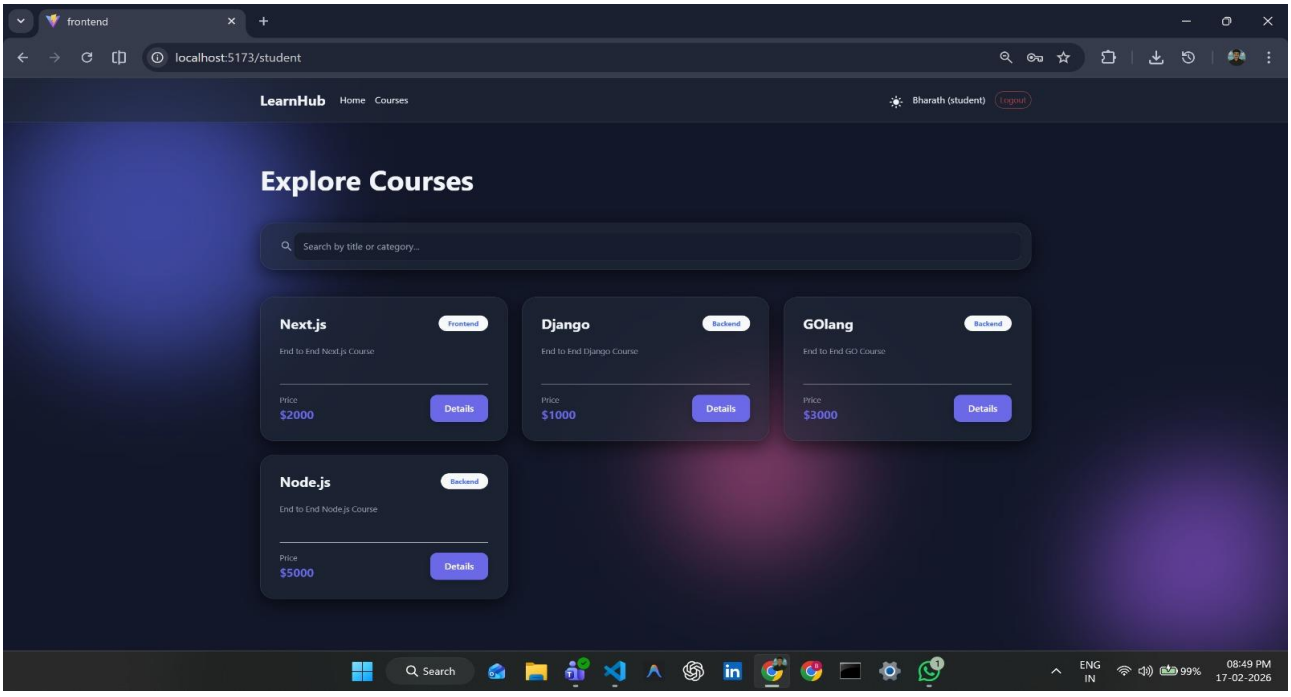
Security Measures

- Passwords are hashed before storage
- JWT tokens prevent unauthorized API access
- Middleware protects admin routes
- Environment variables store secret keys securely

This authentication system ensures secure login, protected data access, and proper role-based control within the LearnHub platform.

9. User Interface





10. Testing

Testing in the LearnHub project was performed to ensure the application works correctly, securely, and efficiently across different modules. Both manual and functional testing approaches were used.

Testing Strategy

The testing process focused on verifying each module individually and then testing the complete system after integration.

- **UnitTesting:**
  Individual components such as login forms, API routes, and database models were tested separately to ensure they function correctly.
- **IntegrationTesting:**
  Interaction between frontend, backend, and database was tested to confirm data flows properly (e.g., login → API → database → response).
- **SystemTesting:**
  The entire application was tested as a whole to ensure all features like registration, course browsing, enrollment, and admin actions work smoothly.
- **UserInterfaceTesting:**
  The responsiveness and usability of the interface were checked on different screen sizes and browsers.

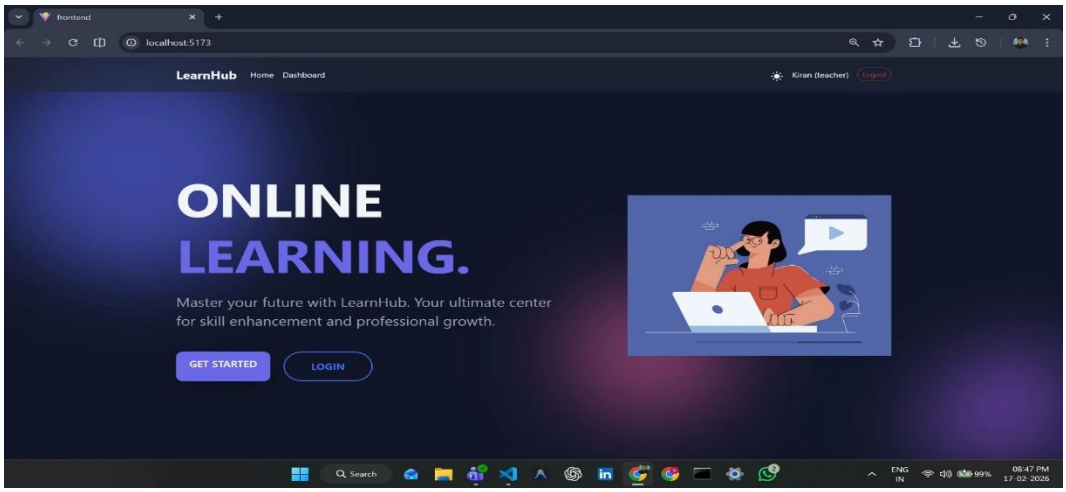
Tools Used

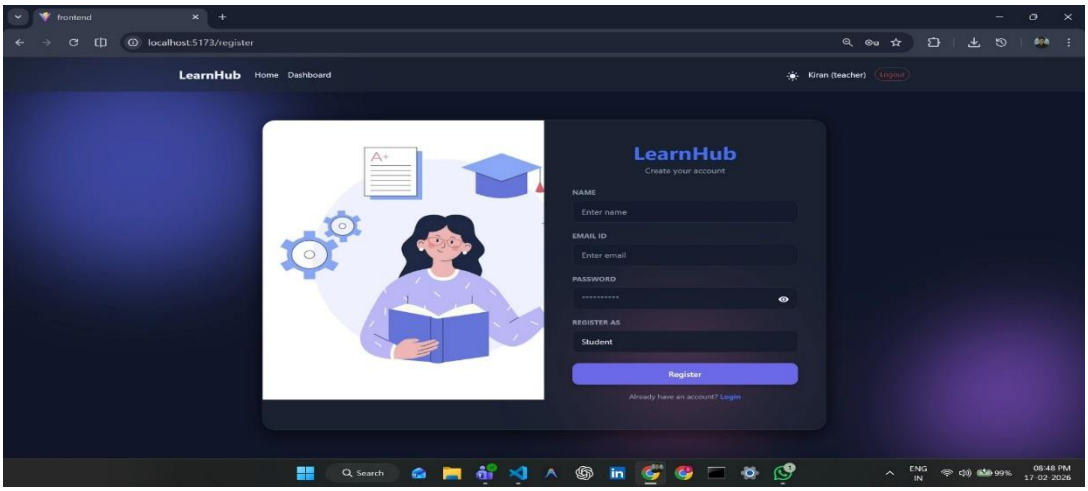
- **Browser Developer Tools** – Used to inspect UI, debug errors, and monitor API calls
- **Postman** – Used to test backend API endpoints and responses
- **MongoDB Compass** – Used to verify stored data and database operations
- **VS Code Debugger** – Used to identify and fix code issues

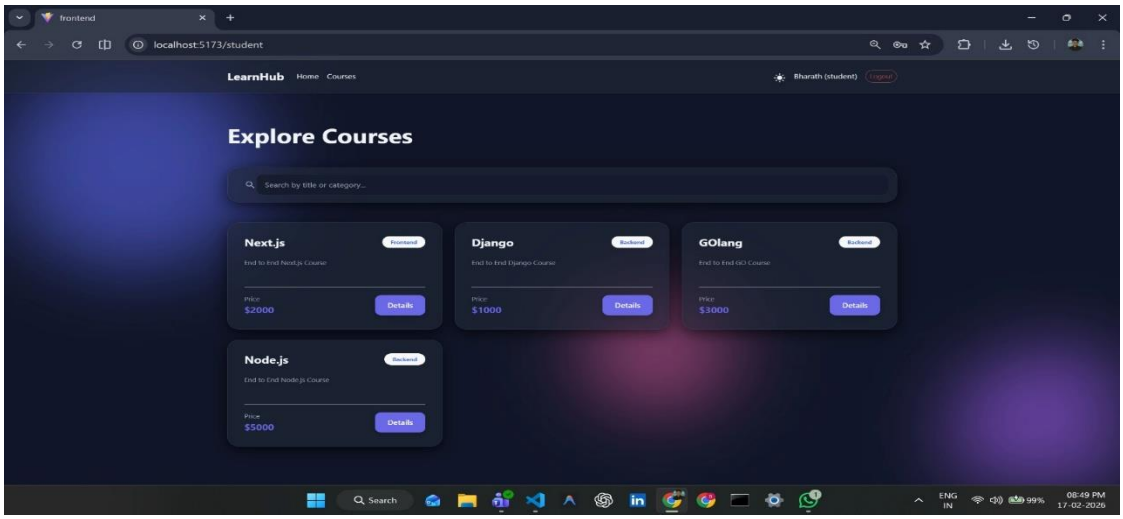
Test Results

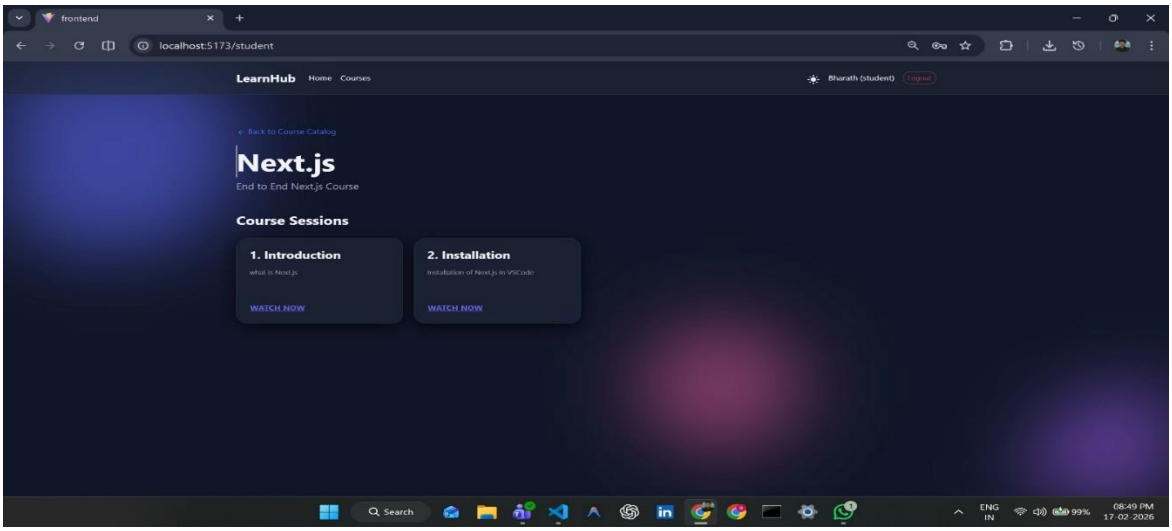
All major functionalities such as authentication, course management, and progress tracking were tested successfully. Minor UI issues were corrected during development, and the system performed reliably under normal usage conditions.

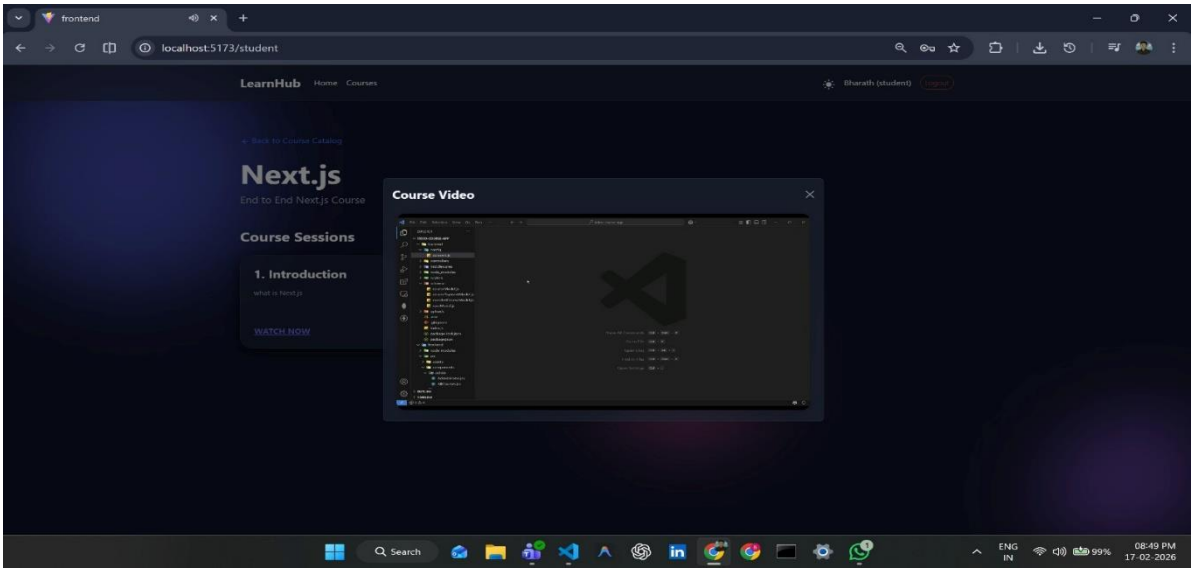
11. Screenshots or Demo

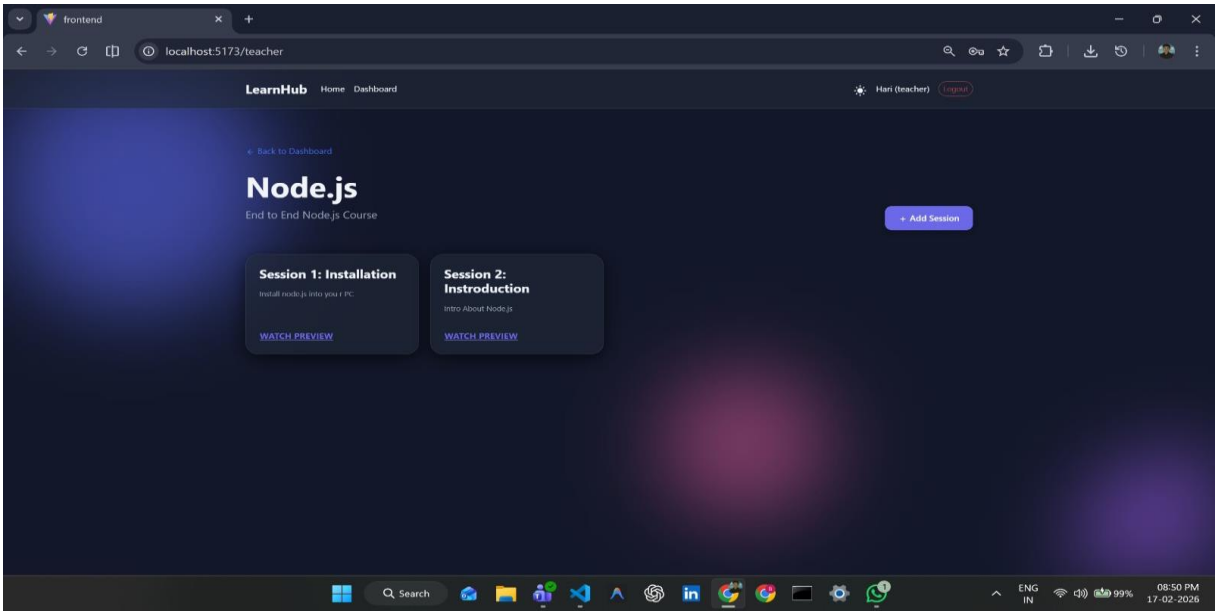







frontend
localhost:5173/student
LearnHub
Home
Courses
Bharath (student)
Logout
← Back to Course Catalog
Next.js
End to End Next.js Course
Course Sessions
1. Introduction
what is Next.js
WATCH NOW
2. Installation
Installation of Next.js in VSCode
WATCH NOW
Search
ENG
IN
99%
08:49 PM
17-02-2026


frontend
localhost:5173/student
LearnHub
Home
Courses
Bharath (student)
Logout
← Back to Course Catalog
Next.js
End to End Next.js Course
Course Sessions
1. Introduction
what is Next.js
WATCH NOW
Course Video
Search
ENG
IN
99%
08:49 PM
17-02-2026


frontend
localhost:5173/teacher
LearnHub
Home
Dashboard
Hari (teacher)
Logout
← Back to Dashboard
Node.js
End to End Node.js Course
+ Add Session
Session 1: Installation
Install node.js into your PC
WATCH PREVIEW
Session 2: Instroduction
Intro About Node.js
WATCH PREVIEW
Search
ENG
IN
99%
08:50 PM
17-02-2026

12. Known Issues

Although the LearnHub application functions correctly for core features, a few limitations and minor issues were identified during testing and development.

- **Limited Course Content:**
  Currently, only sample courses are available. The platform requires more real-world content to fully demonstrate scalability**.**

- **Basic Progress Tracking:**
  Progress is tracked at a simple percentage level and does not yet support detailed analytics such as quiz performance or time spent learning.

- **Authentication Token Expiry Handling:**
  If a token expires, users may need to manually log in again instead of being automatically redirected to the login page.

- **UI Enhancements Needed:**
  Some pages require improved styling and better spacing for a more professional user experience.

- **No Email Verification:**
  The system does not currently verify user email addresses during registration.

- **Deployment Configuration Issues:**
  When deploying to hosting platforms, environment variables must be configured carefully; otherwise, database connections may fail.

13. Future Enhancements

The LearnHub platform can be further improved by adding advanced features to enhance usability, scalability, and learning effectiveness.

- **Live Classes Integration:**
  Add real-time virtual classrooms using video conferencing so instructors can conduct live sessions and interact with students.
- **Quiz and Certification System:**
  Introduce quizzes, assignments, and automatic certificate generation after course completion to improve learner engagement.
- **Advanced Progress Analytics:**
  Provide detailed dashboards showing learning time, quiz scores, and performance insights for better tracking.
- **Email Notifications:**
  Implement automated emails for course enrollment, reminders, and completion updates.
- **Payment Gateway Integration:**
  Allow paid courses with secure online payment options to support premium learning content.
- **Mobile Application:**
  Develop an Android/iOS mobile app for better accessibility and on-the-go learning.
- **AI-Based Course Recommendations:**
  Suggest courses to users based on their interests, progress, and learning history.
- **Discussion Forum & Community Features:**
  Add student discussion boards to encourage peer learning and doubt clarification.

Conclusion

The LearnHub project successfully demonstrates the development of a full-stack online learning platform using modern web technologies. The system provides essential features such as user authentication, course management, and progress tracking, creating a structured digital environment for skill development.

Through this project, we gained practical experience in building a scalable MERN-stack application, integrating frontend and backend systems, designing databases, and implementing secure authentication mechanisms. The platform achieves its primary goal of offering an accessible and user-friendly learning solution while also serving as a strong foundation for future enhancements.

Overall, LearnHub highlights how web technologies can be effectively used to create educational platforms that support continuous learning and digital skill growth.