

# **LEARNHUB: ONLINE LEARNING PLATFORM USING MERN**

## **Introduction:**

In today's digital era, online learning platforms play a crucial role in providing accessible, flexible, and personalized education to students across the world. Traditional classroom learning methods often lack scalability, real-time interaction, and centralized content management.

LearnHub is a full-stack web application designed to provide a structured online learning environment where Admins, Teachers, and Students can interact efficiently. The system allows teachers to create and manage courses, students to enroll and learn through structured content, and administrators to monitor and control the entire platform.

The platform is developed using the MERN stack (MongoDB, Express.js, React.js, Node.js) to ensure scalability, performance, and modern user experience. LearnHub aims to simplify course delivery, improve engagement, and provide secure authentication with role-based access control.

## **Scenario-Based Case Study – Learning A New Skill**

### **User Registration:**

A student creates an account using email and password.

### **Browsing Courses:**

Courses are displayed based on category, difficulty, and popularity.

### **Enrollment:**

The learner enrolls and accesses videos, materials, and assignments.

### **Learning Progress:**

Progress is saved so learning can continue anytime.

### **Interaction:**

Discussion forums and webinars enable communication with instructors.

### **Completion & Certification:**

After finishing modules and exams, the learner receives a certificate.

### **Paid Courses:**

Premium courses require payment before access.

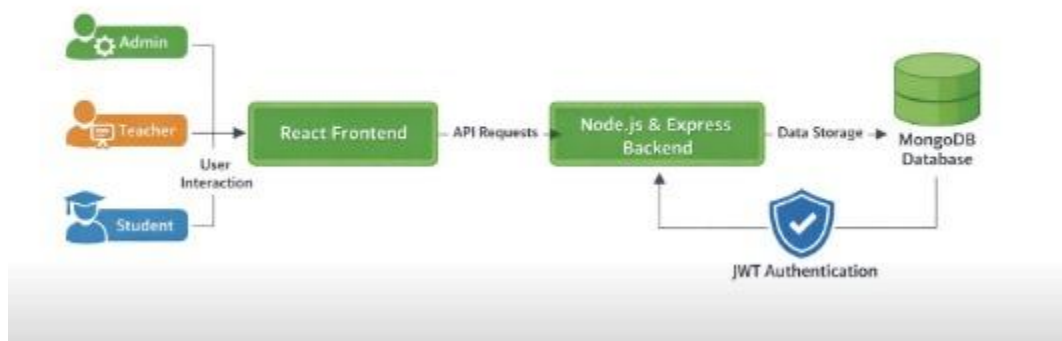
### **Teacher Role:**

Teachers create, manage, and monitor courses.

## Admin Role:

Admins manage users, courses, and overall platform activity.

## Technical Architecture:



## Database Design:

### Users Collection

- `_id`
- `name`
- `email`
- `password`
- `type`

### Courses Collection

- `userID`
- `_id`
- `educator`
- `categories`
- `title`
- `description`
- `sections`
- `price`
- `enrolled count`

## Pre-requisites:

- Vite – frontend build tool
- Node.js & npm – server runtime
- Express.js – backend framework
- MongoDB – NoSQL database
- React.js – UI library

- HTML, CSS, JavaScript – core web technologies
- Mongoose – database connectivity and CRUD operations

## Required Packages

### Frontend (React):

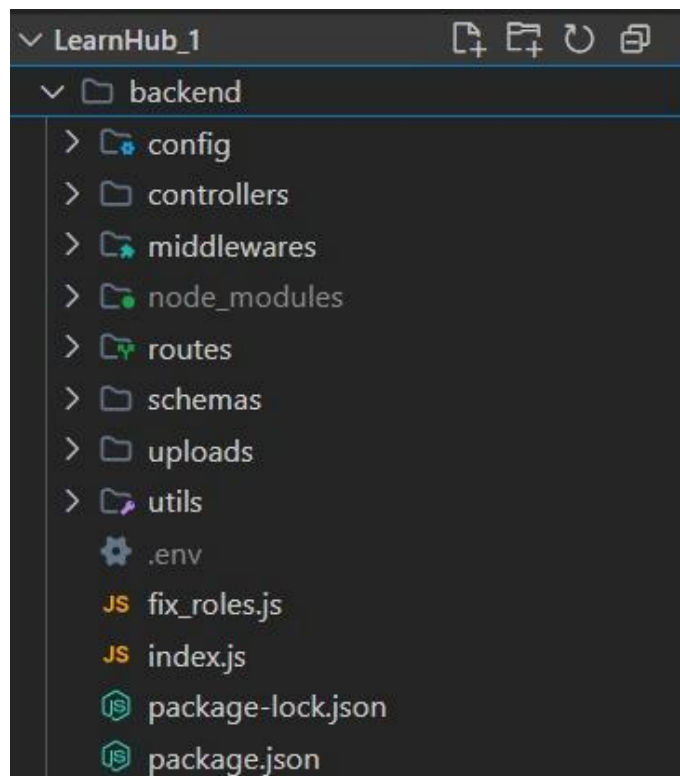
- react, react-router-dom
- axios
- tailwindcss / css
- vite

### Backend (Node.js):

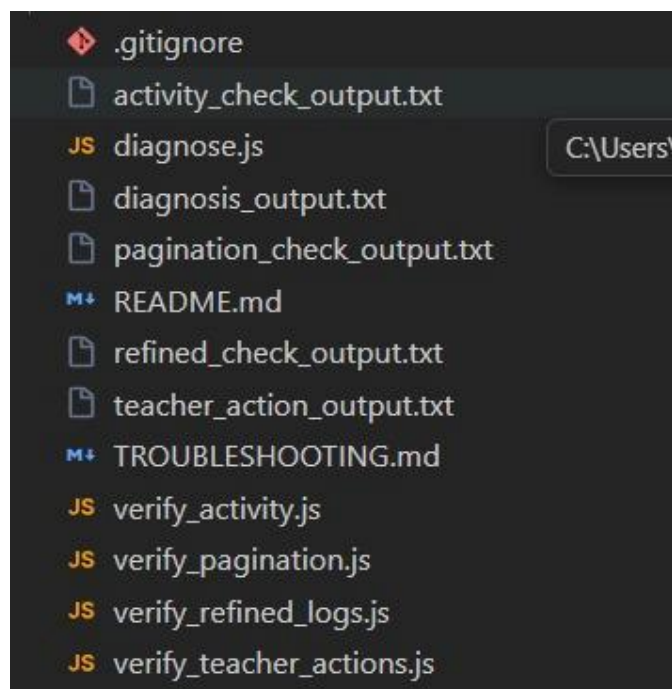
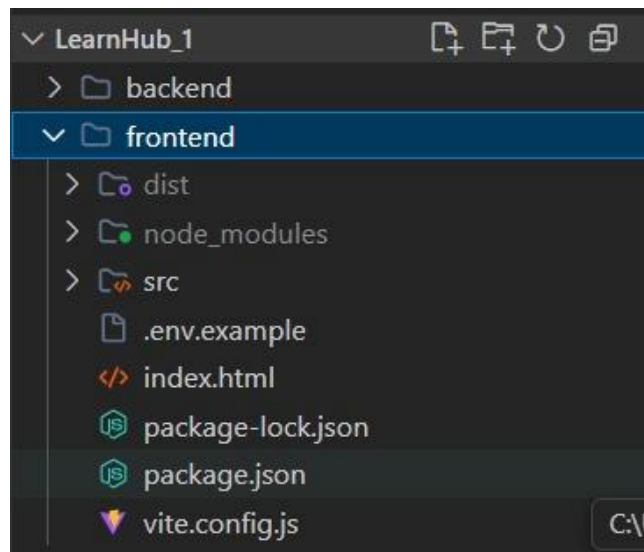
- express
- mongoose
- dotenv
- jsonwebtoken (JWT authentication)
- bcryptjs (password hashing)
- cors

## project Structure

### Backend:



## Frontend:



## Application Flow

### Teacher

The teacher can create and add new courses for students on the platform.

The teacher can delete existing courses when required.

The teacher can organize course content by adding different sections to each course.

### Student

Students can enroll in one or more available courses based on their interests.

Students can resume learning from the point where they previously stopped.

After completing a course, students can download their completion certificates.

For paid courses, students must complete the purchase process before accessing the content.

Students can search and filter courses using criteria such as course name or category.

## **Admin**

The admin manages all users and courses available on the platform.

The admin monitors student enrollments and overall system activity.

The admin ensures the proper functioning, security, and integrity of the entire platform.

## **Development Milestones**

### **Milestone 1: Data & System Design**

Designed database schema:

Users collection

Courses collection

Enrollments collection

Defined REST API routes

Planned role-based access

### **Milestone 2: Frontend Development**

Built responsive UI using React + Tailwind CSS

#### **Implemented:**

Login & Register pages

Dashboards for Admin, Teacher, Student

Course listing & enrollment pages

Integrated frontend with backend using Axios

```

frontend > src > pages > Login.jsx > ...
1 import React, { useState } from 'react';
2 import { Form, Button, Container, Alert } from 'react-bootstrap';
3 import { useAuth } from '../context/AuthContext';
4 import { useNavigate, Link } from 'react-router-dom';
5
6 const Login = () => {
7   const [email, setEmail] = useState('');
8   const [password, setPassword] = useState('');
9   const [error, setError] = useState('');
10  const { login } = useAuth();
11  const navigate = useNavigate();
12
13  const handleSubmit = async (e) => {
14    e.preventDefault();
15    setError('');
16    try {
17      await login(email, password);
18      navigate('/');
19    } catch (err) {
20      setError(err.response?.data?.message || 'Login failed');
21    }
22  };
23

```

```

frontend > src > pages > Register.jsx > ...
1 import React, { useState } from 'react';
2 import { Form, Button, Container, Alert } from 'react-bootstrap';
3 import { useAuth } from '../context/AuthContext';
4 import { useNavigate, Link } from 'react-router-dom';
5
6 const Register = () => {
7   const [name, setName] = useState('');
8   const [email, setEmail] = useState('');
9   const [password, setPassword] = useState('');
10  const [role, setRole] = useState('student');
11  const [error, setError] = useState('');
12  const { register } = useAuth();
13  const navigate = useNavigate();
14
15  const handleSubmit = async (e) => {
16    e.preventDefault();
17    setError('');
18    try {
19      await register(name, email, password, role);
20      navigate('/');
21    } catch (err) {
22      setError(err.response?.data?.message || 'Registration failed');
23    }
24  };
25

```

```

frontend > src > pages > StudentDashboard.jsx > ...
1 import React, { useEffect, useState } from 'react';
2 import { Card, Button, Row, Col, Tabs, Tab, Alert, ProgressBar, Pagination } from 'react-bootstrap';
3 import api from '../utils/api';
4
5 const StudentDashboard = () => {
6   const [courses, setCourses] = useState([]);
7   const [enrolledCourses, setEnrolledCourses] = useState([]);
8   const [message, setMessage] = useState('');
9
10  // Pagination State
11  const [browsePage, setBrowsePage] = useState(1);
12  const [learningPage, setLearningPage] = useState(1);
13  const itemsPerPage = 10;
14
15  useEffect(() => {
16    fetchCourses();
17    fetchEnrolledCourses();
18  }, []);
19
20  const fetchCourses = async () => {
21    try {
22      const { data } = await api.get('/courses');
23      if (Array.isArray(data)) {

```

```

frontend > src > pages > TeacherDashboard.jsx > ...
1  import React, { useState, useEffect } from 'react';
2  import { Button, Table, Modal, Form, Pagination } from 'react-bootstrap';
3  import api from '../utils/api';
4
5  const TeacherDashboard = () => {
6      const [courses, setCourses] = useState([]);
7      const [showModal, setShowModal] = useState(false);
8      const [isEditing, setIsEditing] = useState(false);
9      const [currentCourseId, setCurrentCourseId] = useState(null);
10     const [newCourse, setNewCourse] = useState({ title: '', description: '', category: '', pr
11
12     // Pagination State
13     const [currentPage, setCurrentPage] = useState(1);
14     const itemsPerPage = 10;
15
16     useEffect(() => {
17         fetchCourses();
18     }, []);
19
20     const fetchCourses = async () => {
21         try {
22             // Fetch courses created by the logged-in teacher
23             const { data } = await api.get('/courses/my');

```

```

frontend > src > pages > AdminDashboard.jsx > AdminDashboard > useEffect() callback
1  import React, { useEffect, useState } from 'react';
2  import { Table, Spinner, Alert, Pagination, Button } from 'react-bootstrap';
3  import api from '../utils/api';
4
5  const AdminDashboard = () => {
6      const [activities, setActivities] = useState([]);
7      const [loading, setLoading] = useState(true);
8      const [error, setError] = useState('');
9      const [currentPage, setCurrentPage] = useState(1);
10     const itemsPerPage = 10;
11
12     useEffect(() => {
13         fetchActivities();
14     }, []);
15
16     const fetchActivities = async () => {
17         try {
18             const { data } = await api.get('/activities');
19             setActivities(data);
20             setLoading(false);
21         } catch (error) {
22             console.error('Failed to fetch activities', error);
23             setError('Failed to load activity logs');

```

## Milestone 3: Backend Development

Created Express.js REST APIs

### Implemented:

User authentication with JWT

Password encryption using bcrypt

CRUD operations for courses

Enrollment management

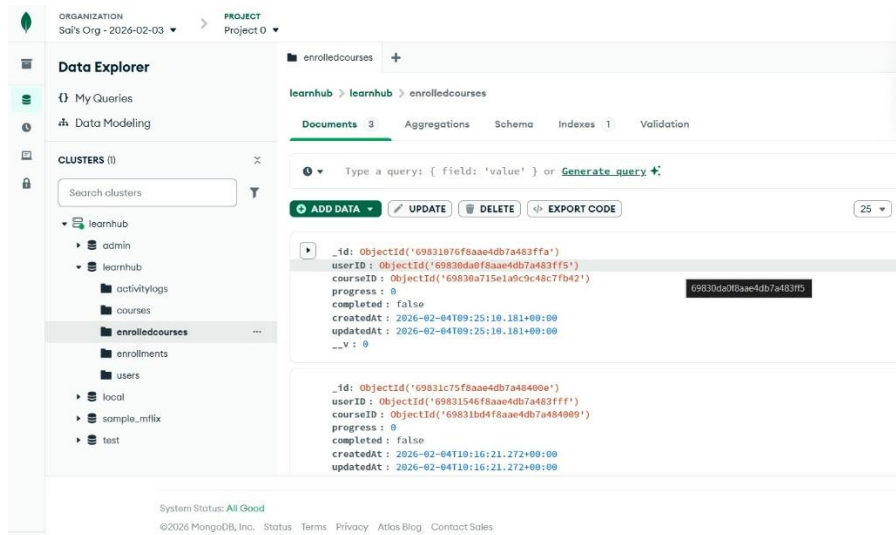
Connected backend to MongoDB Atlas

## Milestone 4: Security & Validation

Protected routes using JWT middleware

Implemented role-based authorization

Added form validation & error handling



## Milestone 5: Implementation

Connected React frontend with Node backend

**Tested:**

Login & registration

Course creation

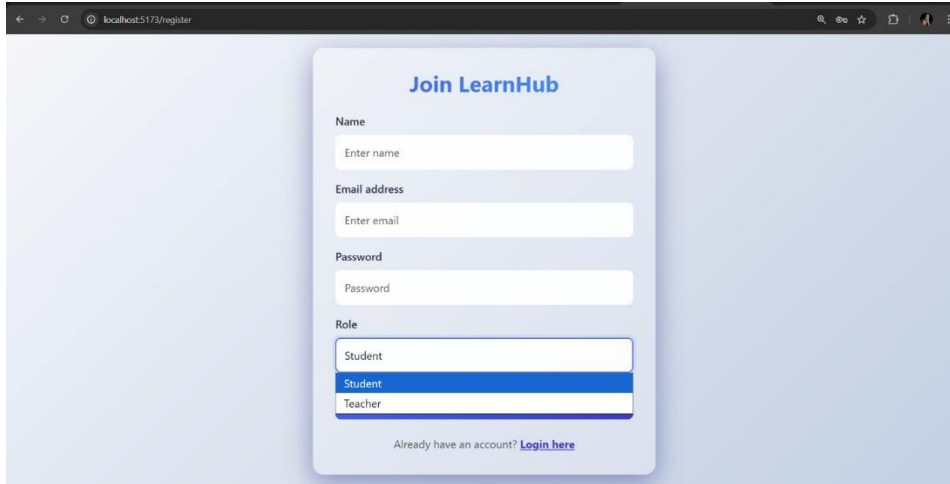
Enrollment flow

Admin controls



# User Interface Modules

## Register Page



A screenshot of a web browser showing the 'Join LearnHub' registration page. The page has a light blue background with a central white card. The card contains the title 'Join LearnHub' in blue, followed by form fields for 'Name', 'Email address', 'Password', and 'Role'. The 'Role' field is a dropdown menu with 'Student' selected. At the bottom of the card, there is a link 'Login here' for users who already have an account.

Join LearnHub

Name  
Enter name

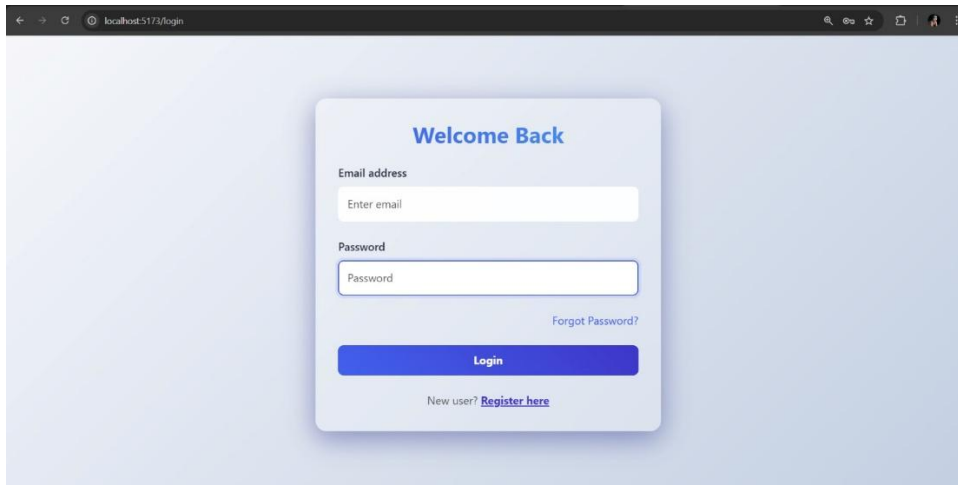
Email address  
Enter email

Password  
Password

Role  
Student  
Student  
Teacher

Already have an account? [Login here](#)

## Login Page



A screenshot of a web browser showing the 'Welcome Back' login page. The page has a light blue background with a central white card. The card contains the title 'Welcome Back' in blue, followed by form fields for 'Email address' and 'Password'. There is a 'Forgot Password?' link and a blue 'Login' button. At the bottom, there is a link 'Register here' for new users.

Welcome Back

Email address  
Enter email

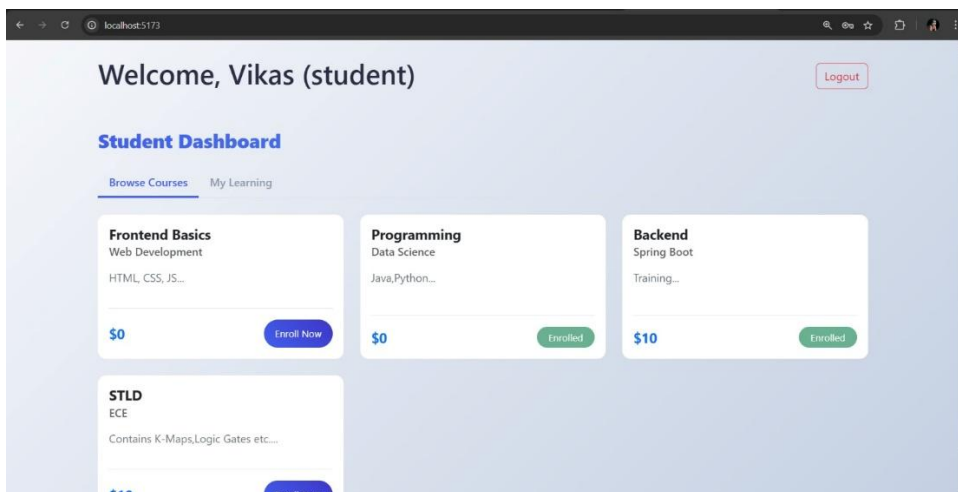
Password  
Password

[Forgot Password?](#)

Login

New user? [Register here](#)

## Student Dashboard



A screenshot of a web browser showing the 'Student Dashboard' for a user named Vikas. The page has a light blue background. At the top, there is a 'Welcome, Vikas (student)' message and a 'Logout' button. Below this, there is a 'Student Dashboard' section with two tabs: 'Browse Courses' (active) and 'My Learning'. Under 'Browse Courses', there are four course cards: 'Frontend Basics' (Web Development, HTML, CSS, JS..., \$0, Enroll Now), 'Programming' (Data Science, Java, Python..., \$0, Enrolled), 'Backend' (Spring Boot, Training..., \$10, Enrolled), and 'STLD' (ECE, Contains K-Maps, Logic Gates etc..., \$10, Enroll Now).

Welcome, Vikas (student) Logout

Student Dashboard

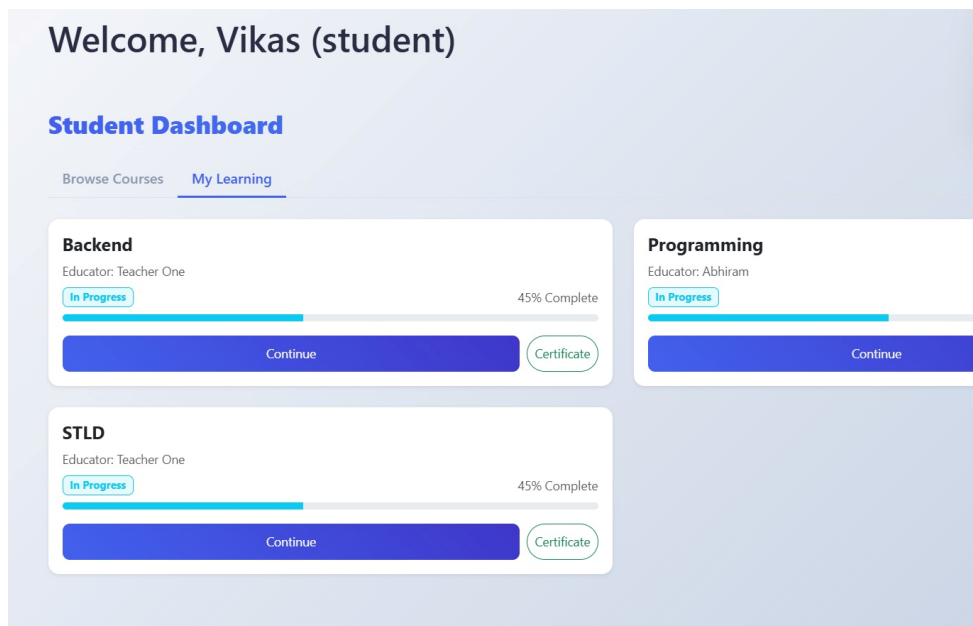
Browse Courses My Learning

**Frontend Basics**  
Web Development  
HTML, CSS, JS...  
\$0 Enroll Now

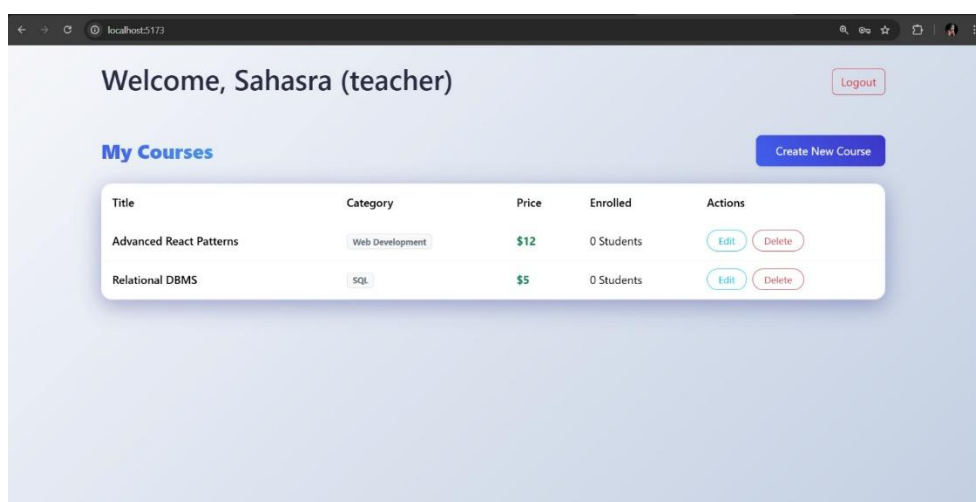
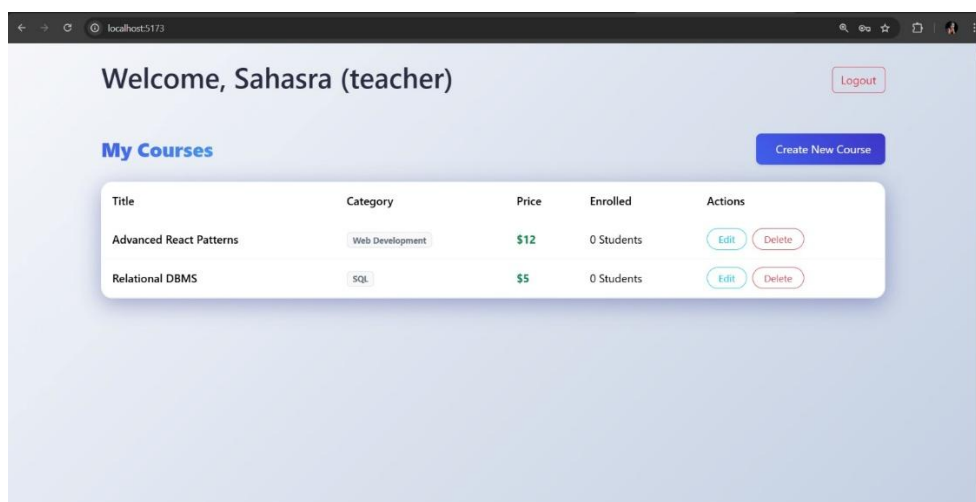
**Programming**  
Data Science  
Java, Python...  
\$0 Enrolled

**Backend**  
Spring Boot  
Training...  
\$10 Enrolled

**STLD**  
ECE  
Contains K-Maps, Logic Gates etc...  
\$10 Enroll Now



## Teacher Dashboard



# Admin Dashboard

localhost:5173

Welcome, Admin One (admin)

Logout

Admin Dashboard

Monitor system-wide user activity logs

Download Logs

Time	User	Role	Action	Details
16/02/2026, 19:26:08	Sahasra	teacher	CREATE COURSE	Created course: Relational DBMS
16/02/2026, 19:25:19	Sahasra	teacher	CREATE COURSE	Created course: Advanced React Patterns
16/02/2026, 19:23:44	Sahasra	teacher	REGISTER	User registered as teacher
16/02/2026, 19:22:08	Abhi	student	LOGIN	User logged in
16/02/2026, 19:20:53	Vikas	student	LOGIN	User logged in
16/02/2026, 19:20:38	Sakhi	teacher	LOGIN	User logged in
16/02/2026, 13:19:33	Sakhi	teacher	LOGIN	User logged in
13/02/2026, 08:51:28	Sakhi	teacher	LOGIN	User logged in
13/02/2026, 08:50:45	Vikas	student	LOGIN	User logged in
12/02/2026, 21:12:21	Vikas	student	LOGIN	User logged in

<

1

2

3

4

5

6

7

8

>