

A PROJECT REPORT

**As partial requirement for the degree of
Master Of Science (Computer Application)**

ON

Online Learning Platform (LearnX)

Group No: _____

SUBMITTED BY :

University Exam No.

**UNDER GUIDANCE OF
Prof. _____**

**COMPUTER SCIENCE AND APPLICATION DEPARTMENT,
Dolat-Usha Institute Of Applied Sciences And Dhiru-Sarla
Institute of Management & Commerce, Valsad**

**Affiliated with
Veer Narmad South Gujarat University, Surat
* 2025-2026 ***

**DOLAT-USHA INSTITUTE OF APPLIED SCIENCES AND
DHIRU-SARLA INSTITUTE OF MANAGEMENT & COMMERCE, VALSAD.**



Computer Science and Application Department

CERTIFICATE

This is to certify that the project work is satisfactorily carried out on **Online Learning Platform (LearnX) Website** is a work done by **Ms. Sonali G. Tandel** in partial fulfillment of the requirement for the award of the degree of Master of Science (Computer Application) during the academic year 2025 - 2026.

I appreciate their hard work, honesty and discipline during the training for concerned project in our organization. The project work was carried out under our supervision and guidance of internal guide.

Prof. Manish Dodia
Project Internal Guide

Dr. Snehal Joshi
I / C Principal and HOD

Project Certified

Date: _____

At M.SC (CA) Examination
Examiners:

1. _____
2. _____

**DOLAT-USHA INSTITUTE OF APPLIED SCIENCES AND
DHIRU-SARLA INSTITUTE OF MANAGEMENT & COMMERCE,
VALSAD**

Project Assessment Detail (internal)

Project Code: - _____

Project Title: - _____

Internal Guide: - _____

Student Name: - _____

Week No.	Week From – Week To	Date of Report	Topic	Task Schedule for Next Week	Remarks of Internal Guide	Signature of Internal Guide
1	28/07/25 TO 02/08/25		Project Definition	Requirement Analysis		
2	04/08/25 TO 09/08/25		Requirement Analysis	System Design		
3	11/08/25 TO 16/08/25		System Design	System Design		
4	18/08/25 TO 23/08/25		System Design	Form Design		
5	25/08/25 TO 30/08/25		Form Design	Coding		
6	01/09/25 TO 06/09/25		Coding	Coding		
7	08/09/25 TO 13/09/25		Coding	Documentation		
8	15/09/25 TO 20/09/25		Documentation	---		

ACKNOWLEDGEMENT

We would like to express my sincere gratitude to **Prof. Manish Dodia** for his invaluable guidance, continuous support, and encouragement throughout the development of my **Online Learning Platform** project. His insights and suggestions have been instrumental in shaping this work.

I also extend my heartfelt thanks to our **Head of Department (HOD)** and all the faculty members for their valuable guidance and motivation. Their support has played a crucial role in helping me complete this project successfully.

A special note of appreciation to my **parents**, whose constant encouragement and belief in me kept me motivated. Lastly, I am grateful to my **friends** for their assistance and valuable feedback. This project would not have been possible without the collective efforts of all.

Thank you!

SONALI TANDEL

INDEX

SR. No	Topics	Page No
1	Project Profile 1.1 Project goal and objectives 1.2 Scope of the project	6
2	System Environment Hardware Software environment	10
3	Platform Environment 3.1 Front End Used 3.2 Back End Used	11
4	System Requirement Analysis	12
5	System Design 5.1 System Flow Diagram 5.2 Data flow diagram 5.3 ER diagram	16
6	Data Dictionary	22
7	Form Designing and coding	28
8	Reports	364
9	Testing	365
10	Conclusion and Future Enhancement	366
11	References/ Bibliography	367

1. Project Profile

1.1 Project goal and objectives

The primary goal of this Learning Management System (LMS) is to create a seamless, interactive, and highly efficient digital learning environment for students, instructors, and administrators. The platform is designed to simplify content delivery, enhance learning outcomes, and streamline academic management through modern digital tools.

- Project Goals**

- 1. Enable Interactive Digital Learning**

Provide a dynamic and engaging learning experience through multimedia lessons, quizzes, assignments, and real-time feedback.

- 2. Centralize Course and Academic Management**

Offer a unified platform where courses, lessons, exams, and student records can be managed efficiently by instructors and administrators.

- 3. Improve Accessibility and Learning Convenience**

Ensure students can easily access course content anytime, track their progress, and view performance insights with minimal effort.

- 4. Empower Instructors with Smart Tools**

Supply instructors with easy-to-use tools for creating content, managing students, reviewing submissions, and monitoring learners' overall growth.

- 5. Support Scalable and Secure Learning Environments**

Deliver a reliable platform that handles increasing user load, protects data, and ensures smooth learning operations.

- **Project Objectives**
- 1. Promote interactive and engaging learning experiences**

Through digital resources such as videos, PDFs, quizzes, discussions, and automated assessments.
 - 2. Enhance student learning outcomes**

By enabling quick access to study materials, progress tracking dashboards, personalized feedback, and performance reports.
 - 3. Provide seamless course and exam management**

With structured modules for adding/editing lessons, scheduling tests, evaluating submissions, and managing course status (Draft, Pending Approval, Published).
 - 4. Streamline administrative workflows**

Including content approval, instructor management, commission tracking, system settings, and revenue distribution.
 - 5. Ensure transparency and clarity for instructors**

By providing clear guidelines on course approval criteria, commission percentages, and publishing requirements.
 - 6. Offer a user-friendly, responsive, and mobile-optimized platform**

So learners and educators can access the system from any device without usability issues.
 - 7. Enable data-driven decision making**

Through analytics on course performance, enrollment trends, revenue insights, and student engagement levels.

1.2 Scope of the project

The scope of this Learning Management System (LMS) includes all features, functionalities, and processes required to support digital learning, course management, and administrative operations. The system is designed to cater to students, instructors, and administrators by providing a centralized platform for creating, delivering, monitoring, and managing educational content.

In-Scope Features

- Student Module
 - **Manage account** (view, update)
 - **Enroll in courses** and access lessons, videos, and resources.
 - **Attempt exams** online.
 - **Track personal progress** and view results instantly.
 - Engage in an **interactive and user-friendly learning environment**.
 - **Download certificates** after course completion.
- Instructor Module
 - **Manage account** (view, update)
 - **Add new courses** with titles, descriptions, and categories.
 - **Upload lessons** with content, videos, and resources.
 - **Create, schedule, and manage exams** for students.
 - Track student performance and **generate exam reports (results)**.
 - Edit course and lesson materials anytime.
- Admin Module
 - Manage **users** (add, update, delete Admins, Instructors, and Students).
 - **Approve / reject courses** submitted by instructors.
 - Maintain **system security and data integrity**.
 - Generate and view **reports on courses, categories, exams, and user activity**.
 - **Monitor revenue reports** (total earnings, instructor payouts, commissions).
 - Oversee the **overall functioning of the LMS**.

Platform-Level Features

- Secure authentication and role-based access
- Responsive UI supporting mobile, tablet, and desktop
- Integrated payment processing for course enrollments
- Automatic enrollment, expiry management, and renewal handling
- Centralized dashboard for all user roles
- Real-time notifications and email alerts

2. System Environment

2.1 Hardware Software environment

2.1.1. Hardware Environment:

- **Processor:** Intel i3 / i5 or equivalent
- **RAM:** 4 GB (Client), 8 GB (Server)
- **Storage:** 100 GB (Client), 256 GB SSD (Server)
- **Internet:** Stable high-speed connection

2.1.2. Software Environment:

- **Operating System:**
 - Server: Linux (Ubuntu, CentOS) or Windows Server.
 - Client: Windows, macOS
- **Development Tools:**
 - Frontend: React.js, HTML, CSS, JavaScript.
 - Backend: Node.js with Express.js
 - Database: MongoDB
 - Tools and library: Bootstrap, Axios, API calls
- **Other Tools:**
 - Code Editor: VS Code
 - Version Control: Git.

3. Platform Environment

3.1 Front End Used

- **React.js** – For building dynamic and responsive user interfaces.
- **HTML, CSS, JavaScript** – Core web technologies for structure and styling.

3.2 Back End Used

Node.js – Server-side runtime environment.

Express.js – Web framework for building APIs and handling requests.

- **Database Used:**
 - **MongoDB** – NoSQL database to store users, courses, lessons, and results.
- **Tools & Libraries:**
 - **Multer** – For file uploads (videos, images, documents).
 - **JWT** – For authentication and session management.
 - **Postman** – For testing APIs.

4. System Requirement Analysis

System Requirement Analysis identifies the functional and non-functional requirements of the LMS to ensure the system meets the needs of students, instructors, and administrators. This phase outlines what the system must do, how it must perform, and the constraints under which it should operate. It forms the foundation for design, development, testing, and deployment.

4.1 Functional Requirements

These define *what* the system should do.

4.1.1 User Management

- Users can register and log in with secure authentication.
- Admin can manage (add, update, delete) Instructors, Students, and other Admins.
- All users can manage their accounts (view & update profile).

4.1.2 Course Management

- Instructors can create, edit, and delete courses.
- Courses include title, description, category, price, and thumbnail.
- Admin reviews courses and can approve/reject them.
- Students can browse, enroll in, and access courses.

4.1.3 Lesson & Content Management

- Instructors can upload lessons, videos, documents (PDFs), and resources.
- Students can access lessons, download materials, and continue from their last viewed lesson.

4.1.4 Exam & Assessment Management

- Instructors can create quizzes/exams and set schedules.
- Students can attempt exams online.
- System automatically evaluates objective questions.
- Instructors can manually grade descriptive questions.
- Students can view results instantly or after evaluation.

4.1.5 Enrollment & Payment Management

- Students can pay and enroll in courses.
- Admin manages commissions and revenue distribution.
- Enrollment includes auto-expiry and renewal options.

4.1.6 Progress & Performance Tracking

- Students can view progress, completed lessons, and scores.
- Instructors can track student performance reports.
- Admin can generate platform-wide reports.

4.1.7 Notifications & Alerts

- Email or in-app notifications for new courses, approvals, exam schedules, results, etc.

4.1.8 Reporting & Analytics

- Admin can generate reports on users, courses, revenue, exams, and activity logs.
- Instructors get analytics on enrollments, earnings, and performance.

4.2 Non-Functional Requirements

These define *how* the system should behave.

4.2.1 Performance Requirements

- System should support multiple concurrent users with minimal latency.
- Dashboard and content should load within acceptable response times.

4.2.2 Security Requirements

- Passwords must be encrypted.
- JWT-based secure authentication.
- Role-based access control for Admin, Instructor, Student.
- Secure payment gateway integration.

4.2.3 Reliability & Availability

- System should operate continuously with minimal downtime.
- Automatic recovery mechanisms in case of server failure.

4.2.4 Usability Requirements

- User-friendly interface for all roles.
- Responsive design suitable for mobile, tablet, and desktop.
- Simple navigation and consistent UI.

4.2.5 Scalability Requirements

- System must support growing numbers of users, courses, and content.
- Cloud hosting support for scaling storage and performance.

4.2.6 Maintainability Requirements

- Code should follow modular structure for easy updates.
- Well-documented API endpoints.

4.2.7 Data Integrity Requirements

- Ensure consistency across courses, users, exams, and payments.
- Prevent data duplication and unauthorized access.

4.3 User Requirements

Defines what each user expects from the system.

Admin

- Full control over courses, users, exams, and system configuration.
- Ability to approve/reject content.
- Access to comprehensive reports.

Instructor

- Easy tools to create and manage courses, lessons, and exams.
- Monitor student performance.
- Manage earnings and course analytics.

Student

- Easy enrollment and access to learning materials.
- Simple exam attempt flow and instant results.
- Track progress and achievements.

4.4 System Constraints

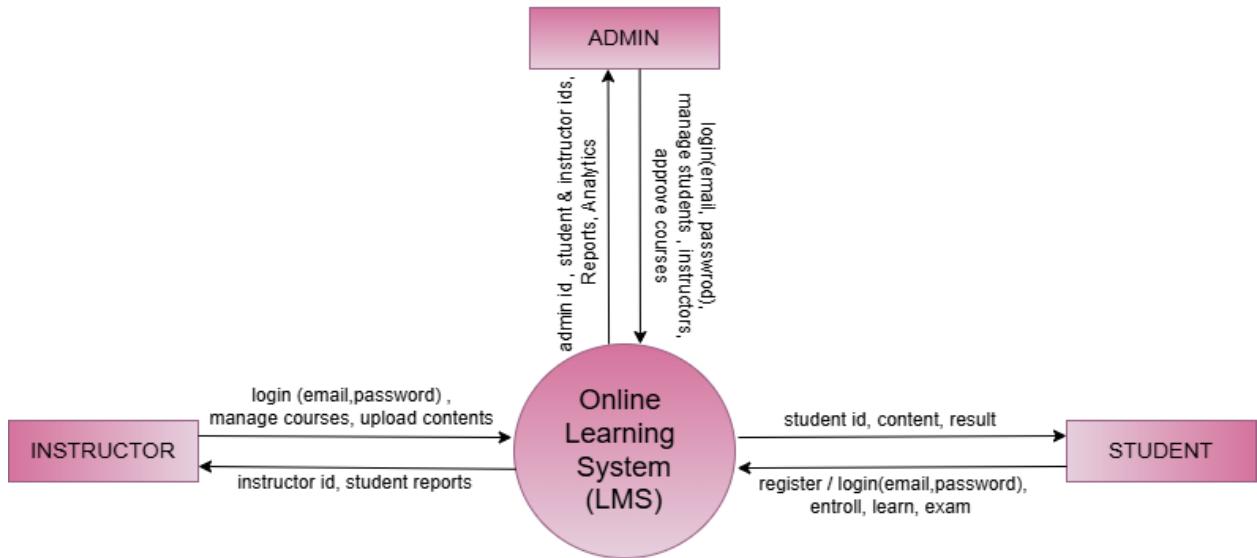
- Internet-dependent system (requires stable connection).
- Browser-based environment, must run on modern browsers.
- Limited access depending on user role permissions.

5. System Design

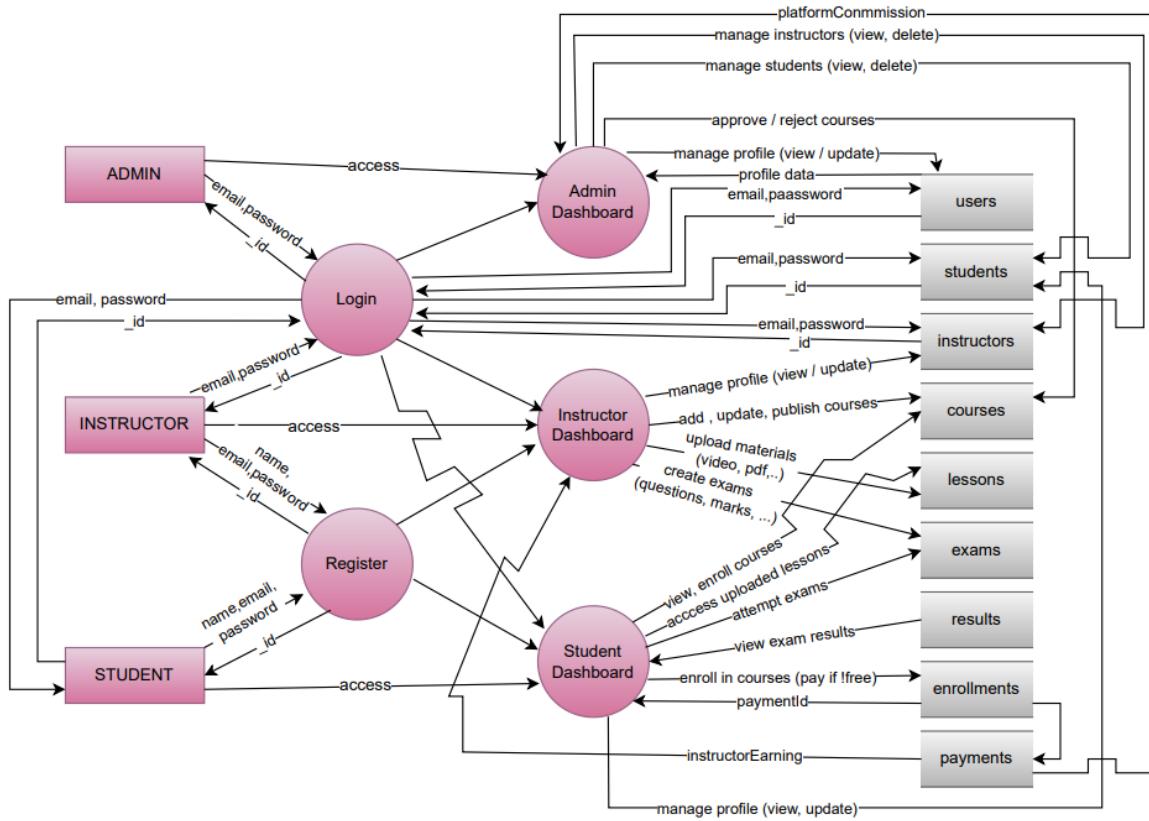
5.1 System Flow Diagram

5.2 Data flow diagram

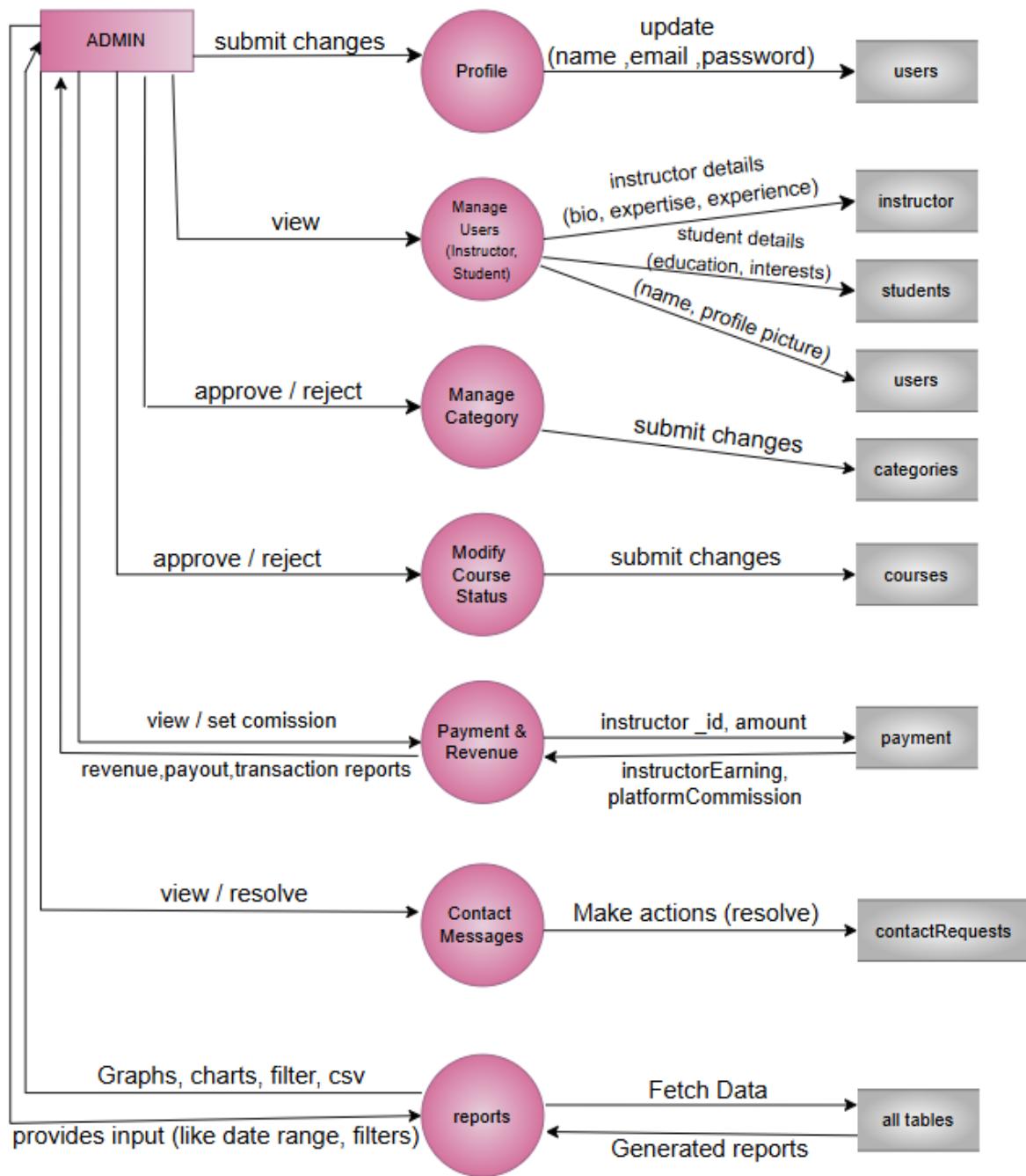
- 0th (Context) level:



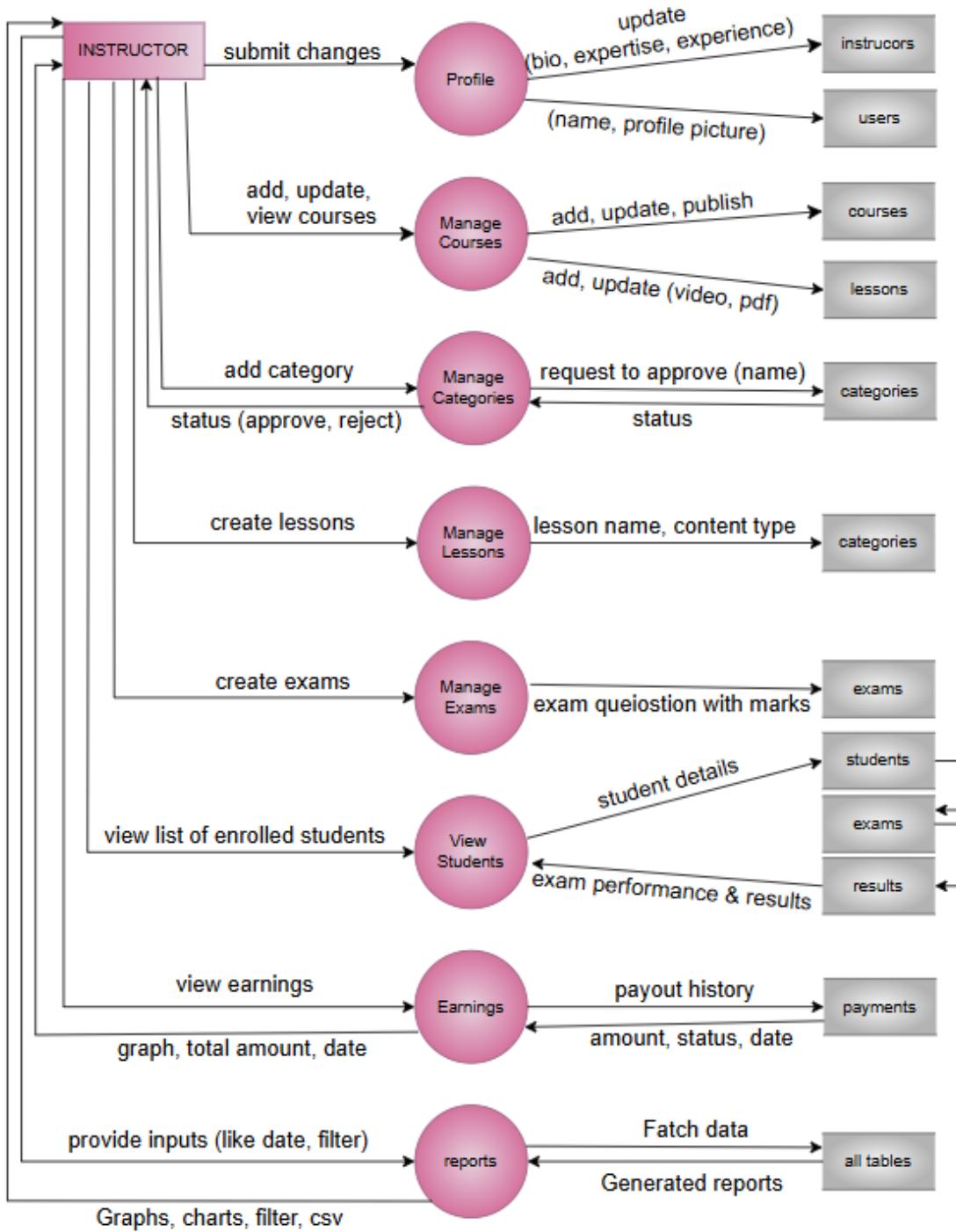
■ 1st level:



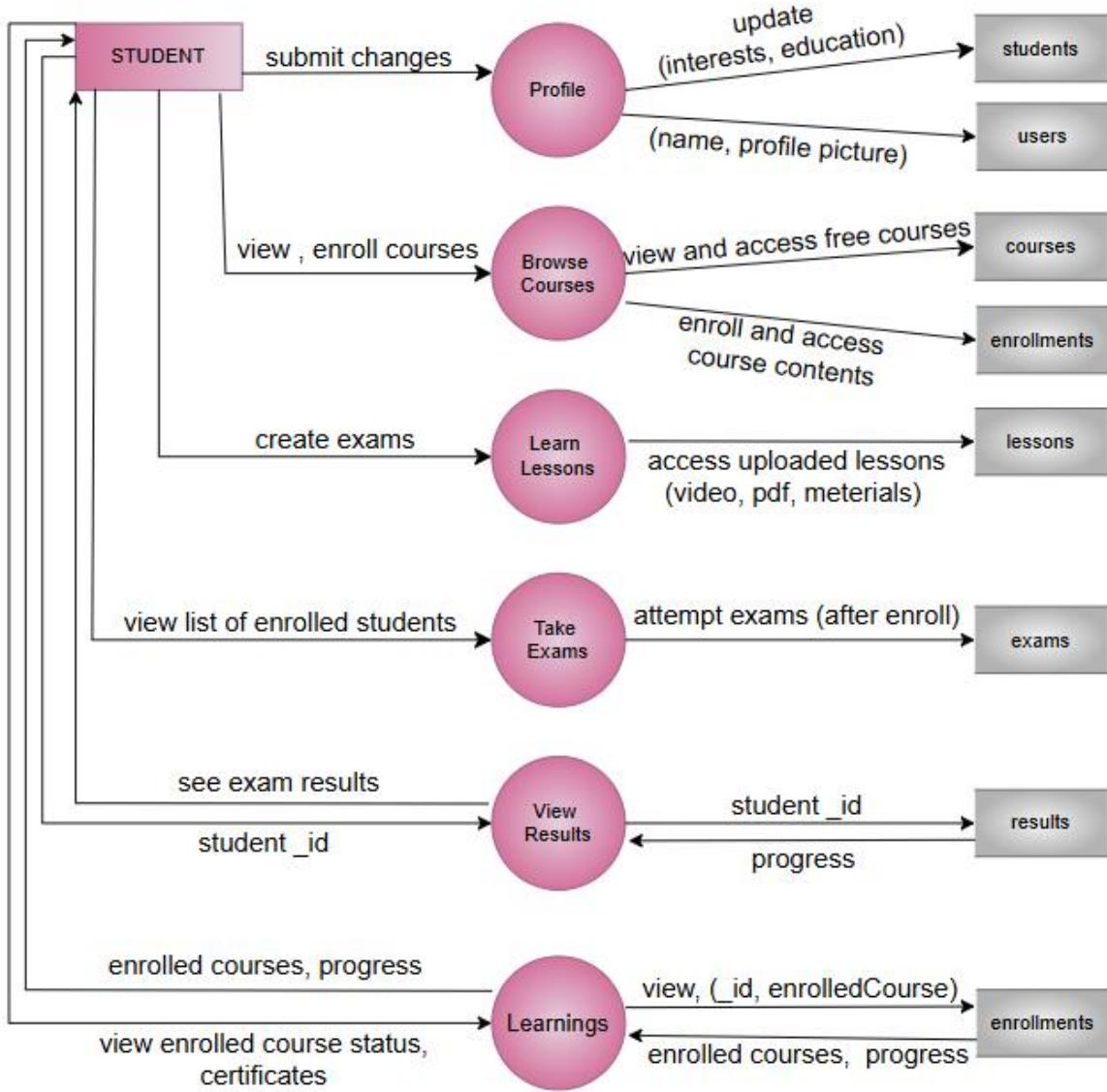
- 2nd level (Admin):



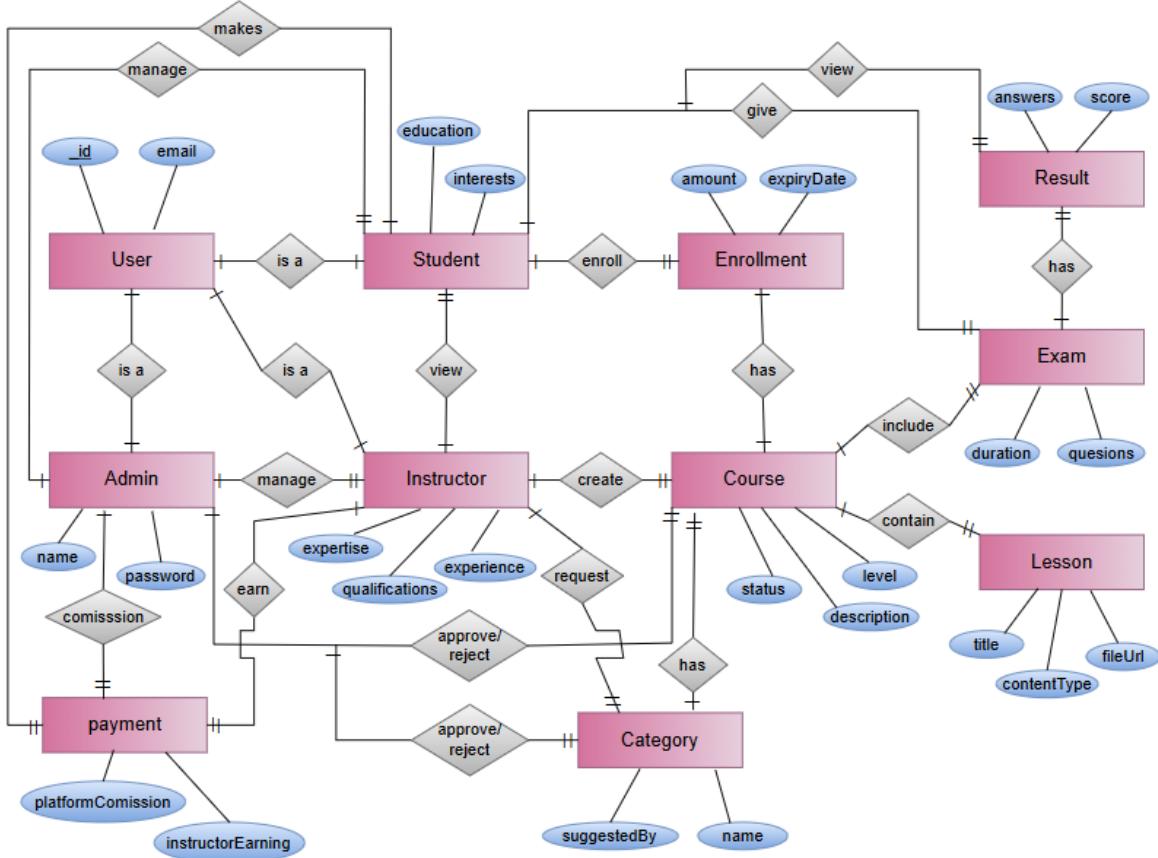
- **2nd level (Instructor):**



- 2nd level (Student):



5.3 ER diagram



6. Data Dictionary

- **Users**

Field	Type	Key	Default
_id	String	Primary	-
name	String	-	-
email	String	Unique	-
password	String	-	-
role	String (enum: admin, instructor, student)	-	-
profilePic	String	-	/upload/default.png

- **Instructors**

Field	Type	Key	Default
_id	String	Primary	-
user	String	Ref: user	-
bio	String	-	-
expertise	[String]	-	-
qualifications	[String]	-	-
experience	number	-	-
courseCreated	String	Ref: course	-

- Students

Field	Type	Key	Default
_id	String	Primary	-
user	String	Ref: user	-
education	String	-	-
interests	[String]	-	-
enrolledCourses	String	Ref: course	-

- Categories

Field	Type	Key	Default
_id	String	Primary	-
name	String	-	-
Status	String (enum: approved, pending, rejected)	-	approved
suggestedBy	String	Ref: user	-
createdAt	Date	-	Date.now

● Courses

Field	Type	Key	Default
_id	String	Primary	-
title	String	-	-
description	String	-	-
level	String	-	-
category	String	Ref: category	-
instructor	String	Ref: user	-
lesson	String	Ref: lesson	-
exams	String	Ref: exam	-
price	Number	-	0
thumbnail	String	-	-
status	String (enum: draft, pendingApproval, approved, rejected)	-	draft

● Enrollments

Field	Type	Key	Default
_id	String	Primary	-
student	String	Ref: user	-
course	String	Ref: course	-
amount	Number	-	-
paymentId	String	-	-
orderId	String	-	-
paymentStatus	String (enum: pending, complete, failed)	-	Pending
paymentDate	Date	-	Date.now
status	String (enum: active, completed, cancelled)	-	Active
expiryDate	Date	-	After 6 months
progress	Number	-	0, Min: 0, Max: 100
completedLessons	String	Ref: lesson	-
examProgress:			
examId	String	Ref: exam	-
attempts	Number	-	0, Max: 3
bestScore	Number	-	0
lastAttemptAt	Date	-	-
isCompleted	Boolean	-	False
certificate:	String	-	null

- Lessons

Field	Type	Key	Default
_id	String	Primary	-
course	String	Ref: course	-
title	String	-	-
contentType	String (enum: video, pdf, text)	-	-
fileUrl	String	-	-
description	String	-	-
isPreviewFree	Boolean	-	False

- Exams

Field	Type	Key	Default
_id	String	Primary	-
course	String	Ref: course	-
title	String	-	-
duration	Number	-	-
questions:			
questionText	String	-	-
options	[String]	-	-
correctAnswer	String	-	-
marks	Number	-	1

- **Results**

Field	Type	Key	Default
_id	String	Primary	-
exam	String	Ref: exam	-
student	String	Ref: user	-
answers			-
questionId	String	-	
selectedOption	String	-	
Score	Number	-	0
attemptNumber	Number	-	1

- **Payments**

Field	Type	Key	Default
_id	String	Primary	-
student	String	Ref: user	-
instructor	String	Ref: user	-
course	String	Ref: course	-
amount	String	-	-
platformCommission	Number	-	30 (%)
instructorEarning	number	-	-
status	String (enum: pending, completed, failed)	-	completed
paymentId	String		-
paymentDate	Date	-	Date.now
paymentMethod	String (enum: Razorpay, PayPal, Bank Transfer)		Razorpay

- **Contacust**

Field	Type	Key	Default
_id	String	Primary	-
name	String	-	-
email	String	-	-
subject	String	-	-
Message	String	-	-
status	String (<u>enum</u> : pending, resolved)	-	Pending

7. Form Designing and coding

App.jsx:

```
import { useState, useEffect } from 'react';
import { BrowserRouter as Router, Routes, Route, useLocation } from "react-router-dom";
import './App.css';
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import Home from "./pages/dashboards/home";
import Login from './pages/auth/login';
import Register from './pages/auth/register';
import Navbar from './components/navbar';
import Footer from './components/footer';
import Profile from './pages/dashboards/profile';
import AdminDashboard from './pages/dashboards/admin/adminDashboard';
import AllUsers from './pages/dashboards/admin/allUsers';
import AllCourses from './pages/dashboards/admin/allCourses';
import PendingCourses from './pages/dashboards/admin/pendingCourses';
import RejectedCourses from './pages/dashboards/admin/rejectedCourses';
import Transactions from './pages/dashboards/admin/transactions';
import CoursePerformance from
'./pages/dashboards/admin/coursePerformance';
import Instructors from './pages/dashboards/admin/instructors';
import Students from './pages/dashboards/admin/students';
import Revenue from './pages/dashboards/admin/revenue';
import Payouts from './pages/dashboards/admin/payouts';
import InstructorDashboard from
'./pages/dashboards/instructor/InstructorDashboard';
import InstructorCourses from './pages/dashboards/instructor/instructorCourses';
import AddCourse from './pages/dashboards/instructor/addCourses';
import PendingApprovals from
'./pages/dashboards/instructor/pendingApprovals';
import ManageLessons from './pages/dashboards/instructor/manageLessons';
import EnrolledStudents from './pages/dashboards/instructor/enrolledStudents';
```

```
import StudentProgress from './pages/dashboards/instructor/studentProgress';
import ManageExams from './pages/dashboards/instructor/manageExams';
import ExamResults from './pages/dashboards/instructor/examResults';
import Courses from './pages/dashboards/student/courses';
import CourseDetail from './pages/dashboards/student/courseDetail';
import Lesson from './pages/dashboards/student/lessons';
import Exams from './pages/dashboards/student/exams';
import MyLearnings from './pages/dashboards/student/myLearnings';
import EnrollmentStats from './pages/dashboards/admin/enrollmentStats';
import ForgotPassword from './pages/auth/forgotPassword';
import CourseAnalytics from './pages/dashboards/instructor/courseAnalytics';
import InstructorEarnings from './pages/dashboards/instructor/earnings';
import PayoutHistory from './pages/dashboards/instructor/payoutHistory';
import ManageCategories from './pages/dashboards/admin/manageCategories';
import CategorySuggestions from
'./pages/dashboards/admin/categorySuggestions';
import RequestCategory from './pages/dashboards/instructor/requestCategory';
import ContactUs from './pages/contactus';
import PrivacyPolicy from './pages/privacyPolicy';
import Terms from './pages/terms';
import Support from './pages/support';
import AboutUs from './pages/aboutus';
import AdminContactMessages from
'./pages/dashboards/admin/contactMessages';

function App() {
  const [user, setUser] = useState(null);
  const location = useLocation();

  useEffect(() => {
    const storedUser = localStorage.getItem("user");
    if (storedUser) setUser(JSON.parse(storedUser));
  }, []);

  const shouldShowNavbar = !(  

    location.pathname.startsWith("/admin-dashboard") ||  

    location.pathname.startsWith("/instructor-dashboard")
```

```

);
return (
<>
{shouldShowNavbar && <Navbar user={user} setUser={setUser} />}
<Routes>
<Route path="/" element={<Home />} />
<Route path="/login" element={<Login setUser={setUser} />} />
<Route path="/register" element={<Register setUser={setUser} />} />
<Route path="/profile" element={<Profile user={user} setUser={setUser} />} />
<Route path="/learning" element={<MyLearnings />} />
<Route path="/courses" element={<Courses />} />
<Route path="/courses/:id" element={<CourseDetail />} />
<Route path="/course/:courseId/lessons/:lessonId" element={<Lesson />} />
<Route path="/course/:courseId/exam/:examId" element={<Exams />} />
<Route path="/forgot-password" element={<ForgotPassword />} />
<Route path="/aboutus" element={<AboutUs />} />
<Route path="/contactus" element={<ContactUs />} />
<Route path="/privacy" element={<PrivacyPolicy />} />
<Route path="/terms" element={<Terms />} />
<Route path="/support" element={<Support />} />

<Route path="/admin-dashboard" element={<AdminDashboard />} />
<Route path="profile" element={<Profile user={user} setUser={setUser} />} />
<Route path="users" element={<AllUsers />} />
<Route path="instructors" element={<Instructors />} />
<Route path="students" element={<Students />} />
<Route path="courses" element={<AllCourses />} />
<Route path="pending-courses" element={<PendingCourses />} />
<Route path="rejected-courses" element={<RejectedCourses />} />
<Route path="categories" element={<ManageCategories />} />
<Route path="category-suggestions" element={<CategorySuggestions />} />
<Route path="revenue" element={<Revenue />} />
<Route path="payouts" element={<Payouts />} />
<Route path="transactions" element={<Transactions />} />

```

```

<Route path="reports/enrollments" element={<EnrollmentStats />} />
<Route path="reports/courses" element={<CoursePerformance />} />
<Route path="contact-messages" element={<AdminContactMessages />} />
</Route>

<Route path="/instructor-dashboard" element={<InstructorDashboard />}>
  <Route index element={<InstructorCourses />} />
  <Route path="instructor_courses" element={<InstructorCourses />} />
  <Route path="add_courses" element={<AddCourse />} />
  <Route path="pending_approvals" element={<PendingApprovals />} />
  <Route path="request-category" element={<RequestCategory />} />
  <Route path="manage_lessons/:id?" element={<ManageLessons />} />
  <Route path="enrolled_students" element={<EnrolledStudents />} />
  <Route path="manage_exams" element={<ManageExams />} />
  <Route path="exam_results" element={<ExamResults />} />
  <Route path="earnings" element={<InstructorEarnings />} />
  <Route path="payout-history" element={<PayoutHistory />} />
  <Route path="course_analytics" element={<CourseAnalytics />} />
  <Route path="student_progress" element={<StudentProgress />} />
  <Route path="profile" element={<Profile user={user} setUser={setUser} />} />
</Route>
</Routes>
<Footer />
<ToastContainer position="bottom-center" autoClose={3000} />
</>
);
}
}

export default function WrappedApp() {
  return (
    <Router>
      <App />
    </Router>
  );
}

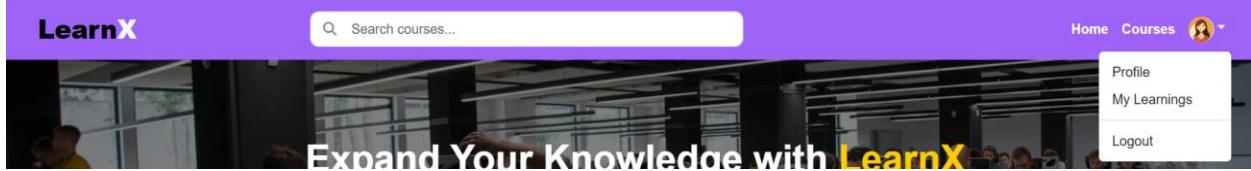
```

Header (Navbar):

(before login)



(after login)



Navbar.jsx

```
import React, { useState, useEffect } from "react";
import { Link, useNavigate, useLocation } from "react-router-dom";
import { FaUserCircle } from "react-icons/fa";
import "../styles/home.css";

const Navbar = ({ user, setUser }) => {
  const navigate = useNavigate();
  const location = useLocation();
  const [searchQuery, setSearchQuery] = useState("");
  const BASE_URL = import.meta.env.VITE_BASE_URL;

  useEffect(() => {
    const params = new URLSearchParams(location.search);
    setSearchQuery(params.get("search") || "");
  }, [location.search]);

  const handleLogout = () => {
    localStorage.removeItem("user");
    localStorage.removeItem("role");
    localStorage.removeItem("token");
    setUser(null);
  };

  return (
    <nav>
      <div>
        <Link to="/">LearnX</Link>
        <input type="text" value={searchQuery} onChange={(e) => setSearchQuery(e.target.value)} />
        <div>
          <a href="#">Home</a>
          <a href="#">Courses</a>
          <div>
            <img alt="User profile icon" />
            <ul>
              <li>Profile</li>
              <li>My Learnings</li>
              <li>Logout</li>
            </ul>
          </div>
        </div>
      </div>
      <div>
        <img alt="Background image of a classroom" />
        <h1>Expand Your Knowledge with LearnX</h1>
      </div>
    </nav>
  );
}

export default Navbar;
```

```

    navigate("/login");
};

const handleSearch = (e) => {
  e.preventDefault();
  navigate(`/courses${searchQuery.trim() ? `?search=${encodeURIComponent(searchQuery)}` : ""}`);
};

return (
  <nav
    className="navbar navbar-expand-lg navbar-light"
    style={{ backgroundColor: "#9f64f7ff", width: "100vw", position: "fixed", top: 0, left: 0, zIndex: 1000 } }
  >
    <div className="container-fluid" style={{ marginRight: "3%", fontWeight: "bold" }}>
      <Link
        className="navbar-brand"
        to="/"
        style={{ marginLeft: "2%", fontWeight: "bolder", fontSize: "30px" }}
      >
        Learn<span style={{ color: "#f2fafdff" }}>X</span>
      </Link>

      <button className="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav">
        <span className="navbar-toggler-icon"></span>
      </button>

      <div className="collapse navbar-collapse" id="navbarNav">
        <form className="d-flex mx-auto position-relative" style={{ width: "40%" }}>
          onSubmit={handleSearch}>
            <input
              className="form-control ps-5"
              type="search"
              placeholder="Search courses..."
              value={searchQuery}
              onChange={(e) => setSearchQuery(e.target.value)}
              style={{ borderRadius: "10px", padding: "8px 15px", position: "relative" }}
            />
            <i
              className="fa fa-search"
              style={{
                position: "absolute",

```

```

        left: "15px",
        top: "50%",
        transform: "translateY(-50%)",
        color: "#888",
        fontSize: "16px",
        pointerEvents: "none",
    )}
></i>
</form>
<ul className="navbar-nav ms-auto align-items-center">
    <li className="nav-item">
        <Link className="nav-link text-white fw-bold" to="/">Home</Link>
    </li>
    <li className="nav-item">
        <Link className="nav-link text-white fw-bold" to="/courses">Courses</Link>
    </li>
    {!user && (
        <li className="nav-item">
            <Link className="nav-link text-white fw-bold" to="/login">Login</Link>
        </li>
    )}
    {user && (
        <li className="nav-item dropdown">
            <Link
                className="nav-link dropdown-toggle text-white fw-bold"
                to="#"
                id="profileDropdown"
                role="button"
                data-bs-toggle="dropdown"
                aria-expanded="false"
            >
                {user.profilePic ? (
                    <img
                        src={user.profilePic.startsWith("http") ? user.profilePic :
                            `${BASE_URL}${user.profilePic}`}
                        alt="Profile"
                        style={{ width: 30, height: 30, borderRadius: "50%", objectFit: "cover" }}
                    />
                ) : (
                    <FaUserCircle size={30} />
                )}
            </Link>
            <ul className="dropdown-menu dropdown-menu-end" aria-
labelledby="profileDropdown">

```

```
<li><Link className="dropdown-item" to="/profile">Profile</Link></li>
<li><Link className="dropdown-item" to="/learning">My Learnings</Link></li>
<li><hr className="dropdown-divider" /></li>
<li><button className="dropdown-item"
onClick={handleLogout}>Logout</button></li>
</ul>
</li>
)}
</ul>
</div>
</div>
</nav>
);
};

export default Navbar;
```

Home.jsx

Search courses...

Home Courses

Expand Your Knowledge with LearnX

Explore top-rated courses and start learning new skills today!

Explore Courses

Why Choose LearnX?

- Expert Instructors**
Learn from industry professionals.
- Flexible Learning**
Study anytime, anywhere.
- Certified Courses**
Get recognized certificates.
- Community Support**
Join our learner community.

Browse by Category

Next JS Data Science Frontend Development Fullstack Development MERN Stack Development Backend Development Javascript Python

Newly Added Courses

 MERN Stack Bootcamp Fullstack development using MongoDB, Express, React, and Node.js with hands-on p... Advanced ₹499	 Backend Development with Node.js Learn to build scalable backend applications using Node.js, Express, and MongoDB... Intermediate ₹299	 Advanced Python for Data Science Master Python libraries like Pandas, NumPy, and Matplotlib for data analysis and... Advanced ₹399	 Mastering Next JS: Part 1 Learn the fundamentals of JavaScript, including variables, functions, loops, and... Beginner ₹99
 Fullstack Development Project Build fullstack applications using MERN stack and deploy them. Advanced ₹399	 Web development Bootcamp Master HTML, CSS, JavaScript, and modern frontend frameworks. Beginner ₹149	 Data Science with Python Learn data analysis, visualization, and machine learning using Python. Intermediate ₹299	 Mastering Next JS: Part 1 Learn server-side rendering, routing, and API integration with Next JS. Beginner ₹199

About LearnX

At LearnX, we are dedicated to empowering learners worldwide. Explore high-quality courses, learn from expert instructors, and gain skills that truly matter. Our mission is to make learning accessible, engaging, and impactful for everyone.

[Learn More](#)

LMS Platform
A complete online learning system for students and instructors. Learn, teach, and grow your skills with ease.

Quick Links
Courses
About Us
Contact Us

Resources
Privacy Policy
Terms & Conditions
Support

Follow Us

Home.jsx

```

import { useState, useEffect } from "react";
import { Link, useNavigate } from "react-router-dom";
import "bootstrap/dist/css/bootstrap.min.css";
import "bootstrap/dist/js/bootstrap.bundle.min.js";
import "../..//styles/home.css";
import api from "../..//api/api";

function Home() {
  const [categories, setCategories] = useState([]);
  const [newCourses, setNewCourses] = useState([]);
  const navigate = useNavigate();
  const BASE_URL = import.meta.env.VITE_BASE_URL;

  // ----- FECTH CATEGORIES -----
  useEffect(() => {
    const fetchCategories = async () => {
      try {
        const res = await api.get("courses/categories", { withCredentials: true });
        setCategories(res.data.categories || []);
      } catch (err) {
        console.error("Fetch Categories Error:", err);
        setCategories([]);
      }
    };
    fetchCategories();
  }, []);

  // ----- FECTH LATEST COURSES -----
  useEffect(() => {
    const fetchNewCourses = async () => {
      try {
        const res = await api.get("/courses", { withCredentials: true });
        const sortedCourses = res.data
          ?.sort((a, b) => new Date(b.createdAt) - new Date(a.createdAt))
          .slice(0, 8);
        setNewCourses(sortedCourses || []);
      } catch (err) {
        console.error("Fetch Courses Error:", err);
        setNewCourses([]);
      }
    };
    fetchNewCourses();
  }, []);
}

```

```

return (
  <div className="home-container">
    /* HERO SECTION */
    <section className="hero-section text-center">
      <h1 className="hero-title">
        Expand Your Knowledge with <span>LearnX</span>
      </h1>
      <p className="hero-subtitle">
        Explore top-rated courses and start learning new skills today!
      </p>
      <button className="explore-btn" onClick={() => navigate("/courses")}>
        Explore Courses
      </button>
    </section>

    /* WHY CHOOSE SECTION */
    <section className="why-choose-section" style={{ margin: "50px 0", textAlign: "center", padding: "0 20px" }}>
      <h2 className="section-title" style={{ fontSize: "2rem", color: "#333", fontWeight: "600", marginBottom: "50px" }}>
        Why Choose <span style={{ color: "#7626edff" }}>LearnX</span>?
      </h2>

      <div className="features-grid" style={{ display: "flex", flexWrap: "wrap", justifyContent: "center", gap: "100px", fontSize: "1rem", color: "#555" }}>
        [
          { icon: "fa fa-graduation-cap", title: "Expert Instructors", desc: "Learn from industry professionals." },
          { icon: "fa fa-clock", title: "Flexible Learning", desc: "Study anytime, anywhere." },
          { icon: "fa fa-certificate", title: "Certified Courses", desc: "Get recognized certificates." },
          { icon: "fa fa-users", title: "Community Support", desc: "Join our learner community." },
        ].map((item, index) =>
          <div key={index} style={{ display: "flex", flexDirection: "column", alignItems: "center", textAlign: "center", minWidth: "150px", maxWidth: "200px" }}>
            <i className={item.icon} style={{ fontSize: "2.3rem", color: "#985e6fff", marginBottom: "10px" }}></i>
            <h4 style={{ margin: "6px 0", fontSize: "1.2rem", fontWeight: "500" }}>{item.title}</h4>
            <p style={{ margin: "0", fontSize: "1rem", color: "#666" }}>{item.desc}</p>
          </div>
        )))
      </div>
    </section>

    /* ----- CATEGORIES SECTION ----- */

```

```

<section className="categories-section" style={{ marginTop: "40px", marginBottom: "40px", textAlign: "center" }}>
  <h2 className="section-title" style={{ marginBottom: "25px", fontSize: "2rem", color: "#333", fontWeight: "600" }}>
    Browse by Category
  </h2>

  <div className="categories-grid" style={{ display: "grid", gridTemplateColumns: "repeat(auto-fit, minmax(150px, 1fr))", gap: "20px", justifyItems: "center", alignItems: "center" }}>
    {categories.length > 0 ? (
      categories.map((cat) => (
        <div
          key={cat._id || cat.name}
          className="category-card"
          onClick={() => {
            const formattedCategory = cat.name.replace(/\ /g, "+");
            navigate('/courses?category=${formattedCategory}');
          }}
          style={{
            background: "#fff",
            borderRadius: "30px",
            padding: "16px",
            width: "100%",
            maxWidth: "180px",
            cursor: "pointer",
            boxShadow: "0 4px 10px rgba(0,0,0,0.1)",
            transition: "transform 0.2s, box-shadow 0.2s",
          }}
          onMouseEnter={(e) => {
            e.currentTarget.style.transform = "translateY(-1px)";
            e.currentTarget.style.boxShadow = "0 8px 20px rgba(0,0,0,0.15)";
          }}
          onMouseLeave={(e) => {
            e.currentTarget.style.transform = "translateY(0)";
            e.currentTarget.style.boxShadow = "0 4px 10px rgba(0,0,0,0.1)";
          }}
        >
          <h5 style={{ margin: "0", fontSize: "0.9rem", color: "#993bf2ff" }}>{cat.name}</h5>
        </div>
      ))
    ) : (
      <p style={{ color: "#888", fontSize: "1rem" }}>No categories available.</p>
    )
  </div>
</section>

```

```

        </div>
    </section>

    {/* ----- NEWLY ADDED COURSES SECTION ----- */}
    <section className="courses-section container my-5">
        <h2 className="section-title text-center mb-4" style={{ marginTop: "-3%" }}>
            Newly Added Courses
        </h2>
        <div className="row">
            {newCourses.length > 0 ? (
                newCourses.map((course) => (
                    <div key={course._id} className="col-md-3 col-sm-6 mb-4">
                        <div className="course-card">
                            <div className="course-image">
                                <img
                                    src={course.thumbnail ? `${BASE_URL}${course.thumbnail}` :
                                    "https://via.placeholder.com/300x180"}
                                    alt={course.title}
                                    style={{ objectFit: "fill" }}
                                />
                            </div>
                            <div className="course-info">
                                <h5 className="course-title">
                                    <Link to={`/courses/${course._id}`}>{course.title}</Link>
                                </h5>
                                <p className="course-description">
                                    {course.description?.length > 80
                                        ? course.description.substring(0, 80) + "..."
                                        : course.description}
                                </p>
                                <div className="course-meta d-flex justify-content-between align-items-center">
                                    <span className="level">{course.level}</span>
                                    <span className="price">{course.price ? `₹${course.price}` : "Free"}</span>
                                </div>
                            </div>
                        </div>
                    </div>
                )))
            ) : (
                <p className="text-center text-muted">No new courses available.</p>
            )
        </div>
    </section>

```

```
/* ----- ABOUT SECTION ----- */
<section className="about-section container my-5">
  <div className="about-grid" style={{ marginTop: "-10%", marginBottom: "-3%" }}>
    <div className="about-image">
      
    </div>
    <div className="about-text">
      <h2>About <span>LearnX</span></h2>
      <p>
        At LearnX, we are dedicated to empowering learners worldwide. Explore high-quality courses, learn from expert instructors, and gain skills that truly matter. Our mission is to make learning accessible, engaging, and impactful for everyone.
      </p>
      <button className="learn-more-btn" onClick={() => navigate("/aboutus")}>Learn More</button>
    </div>
  </div>
</section>
</div>
);
}

export default Home;
```

Register Page:

LearnX

Home Courses Login

Create Your Student Account

Student Instructor

Full Name:

Email:

Education:

Interests:

Password:

Confirm Password:

Clear Register

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

Create Your Instructor Account

Student Instructor

Full Name:

Email:

Profile Picture: No file chosen

Bio:

Expertise:

Qualifications:

Experience (yrs):

Password:

Confirm Password:

Clear Register

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

Register.jsx

```
import { useRef, useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";

import FormButton from "../../components/formButtons";
import FormInput from "../../components/formInputs";
import "../../styles/form.css";
import { registerValidation } from "./validation";
import api from "../../api/api";

function Register({ setUser }) {
  const navigate = useNavigate();
  const [role, setRole] = useState("Student");
  const fileInputRef = useRef(null);
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    password: "",
    con_password: "",
    education: "",
    interests: "",
    profilePic: null,
    bio: "",
    expertise: "",
    qualifications: "",
    experience: ""
  });
}

const [errors, setErrors] = useState({});
const [loading, setLoading] = useState(false);

const handleChange = (e) => {
  const { name, value, type, files } = e.target;
  setFormData((prev) => ({
```

```

...prev,
[name]: type === "file" ? files[0] : value,
});
};

const handleRoleChange = (newRole) => {
setRole(newRole);
setErrors({});
};

const handleSubmit = async (e) => {
e.preventDefault();
const validationErrors = await registerValidation(formData, role);
setErrors(validationErrors);
if (Object.keys(validationErrors).length > 0) return;

try {
 setLoading(true);
const data = new FormData();
Object.keys(formData).forEach((key) => {
 if (formData[key] !== null) data.append(key, formData[key]);
});
data.append("role", role);

const res = await api.post("/register", data, {
 headers: { "Content-Type": "multipart/form-data" },
});

localStorage.setItem("user", JSON.stringify(res.data.user));
localStorage.setItem("role", res.data.user.role);
localStorage.setItem("token", res.data.token);
setUser(res.data.user);

toast.success("Registration successful!", { autoClose: 2000 });

setTimeout(() => {
 const userRole = res.data.user.role.toLowerCase();

```

```

        if (userRole === "admin") navigate("/admin-dashboard");
        if (userRole === "instructor") navigate("/instructor-dashboard");
        if (userRole === "student") navigate("/");
    }, 2000);
} catch (err) {
    if (err.response?.status === 409)
        toast.error("Email already registered!");
    else toast.error("Server error, please try again later.");
} finally {
    setLoading(false);
}
};

const handleReset = (e) => {
    e.preventDefault();
    setFormData({
        name: "",
        email: "",
        password: "",
        con_password: "",
        education: "",
        interests: "",
        profilePic: null,
        bio: "",
        expertise: "",
        qualifications: "",
        experience: ""
    });
    if (fileInputRef.current) fileInputRef.current.value = "";
    setErrors({});
};

return (
    <div className="register-container" style={{ marginTop: "5%" }}>
        <ToastContainer position="top-right" />
        <div className="register-card">

```

```

<h2>Create Your {role === "Student" ? "Student" : "Instructor"} Account</h2>

<div className="role-buttons">
  <FormButton
    type="button"
    onClick={() => handleRoleChange("Student")}
    text="Student"
    className={`student ${role === "Student" ? "active" : ""}`}
  />
  <FormButton
    type="button"
    onClick={() => handleRoleChange("Instructor")}
    text="Instructor"
    className={`instructor ${role === "Instructor" ? "active" : ""}`}
  />
</div>

<form onSubmit={handleSubmit} onReset={handleReset}
autoComplete="off">
  <FormInput label="Full Name" type="text" name="name"
  onChange={handleChange} value={formData.name} error={errors.name}
  autoComplete="name" />
  <FormInput label="Email" type="email" name="email"
  placeholder="abc@mail.com" onChange={handleChange} value={formData.email}
  error={errors.email} autoComplete="email" />

  {role === "Student" && (
    <>
      <FormInput label="Education" type="text" name="education"
      placeholder="e.g. B.Sc in Computer Science" onChange={handleChange}
      value={formData.education} error={errors.education} autoComplete="education"
    />
      <FormInput label="Interests" type="text" name="interests"
      placeholder="e.g. Machine Learning, UI/UX, Teaching" onChange={handleChange}
      value={formData.interests} error={errors.interests} autoComplete="off" />
    </>
  )}
</form>

```

```

        )}

{role === "Instructor" && (
  <>
    <FormInput label="Profile Picture" type="file" name="profilePic"
    onChange={handleChange} inputRef={fileInputRef} error={errors.profilePic} />
    <FormInput label="Bio" type="text" name="bio" placeholder="e.g.
    Passionate React Developer" onChange={handleChange} value={formData.bio}
    error={errors.bio} />
    <FormInput label="Expertise" type="text" name="expertise"
    placeholder="e.g. Web Development, AI, Cloud" onChange={handleChange}
    value={formData.expertise} error={errors.expertise} />
    <FormInput label="Qualifications" type="text" name="qualifications"
    placeholder="e.g. M.Tech in Computer Science" onChange={handleChange}
    value={formData.qualifications} error={errors.qualifications} />
    <FormInput label="Experience (yrs)" type="number" name="experience"
    onChange={handleChange} value={formData.experience}
    error={errors.experience} />
  </>
)
}

<FormInput label="Password" type="password" name="password"
onChange={handleChange} value={formData.password} error={errors.password}
autoComplete="new-password" />
<FormInput label="Confirm Password" type="password"
name="con_password" onChange={handleChange}
value={formData.con_password} error={errors.con_password}
autoComplete="new-password" />

<div className="d-flex">
  <FormButton type="reset" text="Clear" />
  <FormButton type="submit" text={loading ? "Registering..." : "Register"}
  disabled={loading} />
</div>

<p className="link">
  Already have an account? <Link to="/login">Login Here</Link>

```

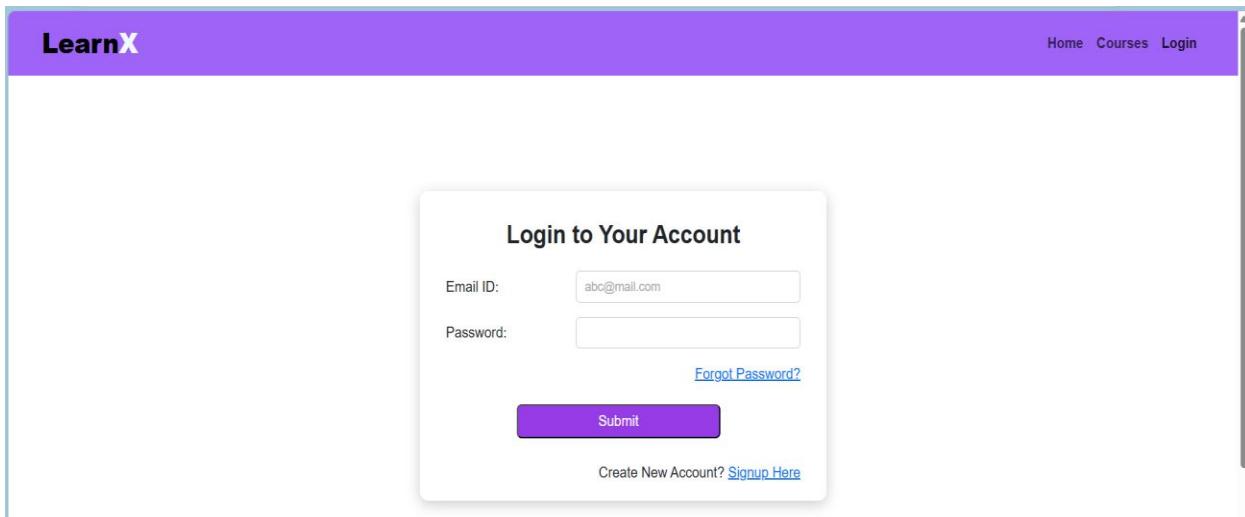
```

        </p>
    </form>
</div>
</div>
);
}

export default Register;

```

Login Page:



Login.jsx

```

import { useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import FormButton from "../../components/formButtons";
import FormInput from "../../components/formInputs";
import "../../styles/form.css";
import { loginValidation } from "./validation";
import api from "../../api/api";

```

```
function Login({ setUser }) {
  const navigate = useNavigate();
  const [formData, setFormData] = useState({ email: "", password: "" });
  const [errors, setErrors] = useState({});
  const [loading, setLoading] = useState(false);

  const handleChange = (e) => {
    setFormData((prev) => ({ ...prev, [e.target.name]: e.target.value }));
  };

  const handleSubmit = async (e) => {
    e.preventDefault();

    const validationErrors = await loginValidation(formData);
    setErrors(validationErrors);
    if (Object.keys(validationErrors).length > 0) return;

    try {
      setLoading(true);
      const res = await api.post("/login", formData);

      const role = res.data.user.role.toLowerCase();
      localStorage.setItem("user", JSON.stringify(res.data.user));
      localStorage.setItem("role", role);
      localStorage.setItem("token", res.data.token);
      setUser(res.data.user);

      toast.success("Login successful!", { autoClose: 2500 });

      setTimeout(() => {
        if (role === "admin") navigate("/admin-dashboard");
        if (role === "instructor") navigate("/instructor-dashboard");
        if (role === "student") navigate("/");
      }, 1500);
    } catch (err) {
      if (err.response?.status === 500) {
        toast.error("Server error..!", { autoClose: 2500 });
      }
    }
  };
}
```

```
    } else {
      toast.error(
        err.response?.data?.error || err.message || "Login failed!",
        { autoClose: 2500 }
      );
    }
  } finally {
  setLoading(false);
}
};

return (
  <div className="login-container">
    <ToastContainer position="bottom-center" autoClose={2500} />
    <div className="login-card">
      <h1>Login to Your Account</h1>
      <form onSubmit={handleSubmit}>
        <FormInput
          label="Email ID"
          type="email"
          name="email"
          id="email"
          placeholder="abc@mail.com"
          value={formData.email}
          onChange={handleChange}
          error={errors.email}
          autoComplete="email"
        />
        <FormInput
          label="Password"
          type="password"
          name="password"
          id="password"
          placeholder="Enter your password"
          value={formData.password}
          onChange={handleChange}
          error={errors.password}
        />
      
    
```

```
    autoComplete="current-password"
  />

  <div style={{ display: "flex", justifyContent: "flex-end" }}>
    <Link to="/forgot-password">Forgot Password?</Link>
  </div>

  <div
    style={{
      display: "flex",
      justifyContent: "center",
      width: "250px",
      marginLeft: "20%",
      marginTop: "5%",
    }}
  >
    <FormButton
      type="submit"
      text={loading ? "Logging in.." : "Submit"}
      disabled={loading}
    />
  </div>

  <p className="link">
    Create New Account? <Link to="/register">Signup Here</Link>
  </p>
</form>
</div>
</div>
);

}

export default Login;
```

forgot Password Page:

Forgot Password

Enter your email

Please enter your email

Send OTP

Forgot Password

OTP sent to hmsonline3111@gmail.com

Enter OTP

Verify OTP

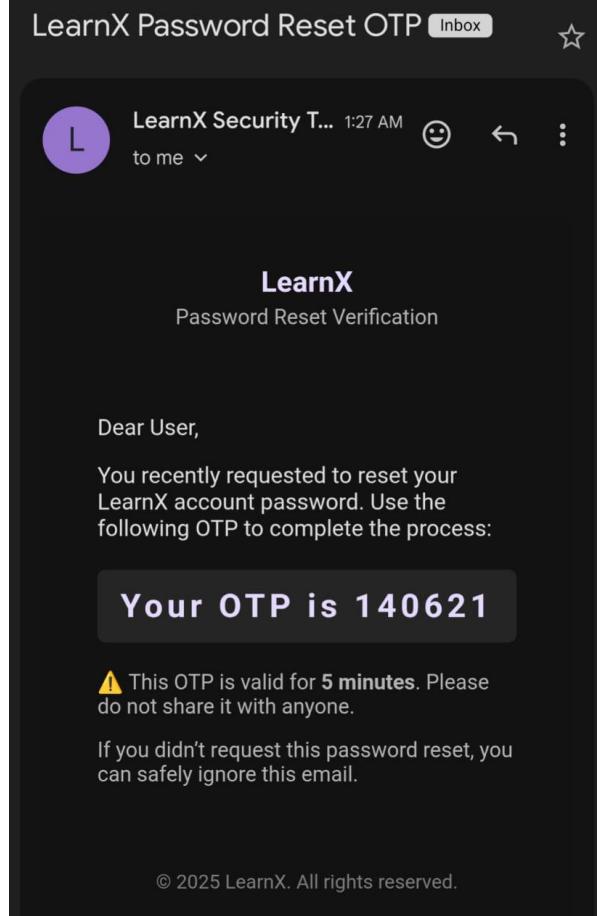
Resend OTP in 40s

Forgot Password

.....

.....

Reset Password



forgotPassword.jsx

```
import { useState, useEffect } from "react";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import api from "../../api/api";
import { useNavigate } from "react-router-dom";

function ForgotPassword() {
  const [email, setEmail] = useState("");
  const [otp, setOtp] = useState("");
  const [newPassword, setNewPassword] = useState("");
  const [confirmPassword, setConfirmPassword] = useState("");
  const [step, setStep] = useState(1);
  const [loading, setLoading] = useState(false);

  const [token, setToken] = useState("");
  const [resetToken, setResetToken] = useState("");

  const [error, setError] = useState("");
  const [timer, setTimer] = useState(0);

  const navigate = useNavigate();

  // ⏳ Timer logic
  useEffect(() => {
    if (timer > 0) {
      const interval = setInterval(() => setTimer((t) => t - 1), 1000);
      return () => clearInterval(interval);
    }
  }, [timer]);

  // ✅ Step 1 — Send OTP
  const handleSendOtp = async () => {
    if (!email) return setError("Please enter your email");
    setError("");
    setLoading(true);
    try {
      const res = await api.post("/auth/send-otp", { email });
      setToken(res.data.token);
      toast.success("OTP sent to your email");
      setStep(2);
      setTimer(60);
    } catch (err) {
      setError("An error occurred while sending OTP");
    }
  };
}
```

```

} catch (err) {
    setError(err.response?.data?.message || "Failed to send OTP");
} finally {
    setLoading(false);
}
};

// ✅ Step 2 — Verify OTP
const handleVerifyOtp = async () => {
    if (!otp) return setError("Please enter OTP");
    setError("");
    setLoading(true);
    try {
        const res = await api.post("/auth/verify-otp", { email, otp, token });
        setResetToken(res.data.resetToken);
        toast.success("OTP verified successfully!");
        setStep(3);
    } catch (err) {
        setError(err.response?.data?.message || "Invalid OTP");
    } finally {
        setLoading(false);
    }
};

// ✅ Step 3 — Reset Password
const handleResetPassword = async () => {
    if (!newPassword || !confirmPassword)
        return setError("Please fill all fields");
    if (newPassword !== confirmPassword)
        return setError("Passwords do not match");

    setError("");
    setLoading(true);
    try {
        const res = await api.post("/auth/reset-password", {
            resetToken,
            newPassword,
        });
        toast.success(res.data.message || "Password reset successful!");
        setStep(1);
        setEmail("");
        setOtp("");
    }
};

```

```

setNewPassword("");
setConfirmPassword("");
setToken("");
setResetToken("");

setTimeout(() => {
  navigate("/login");
}, 1500);
} catch (err) {
  setError(err.response?.data?.message || "Failed to reset password");
} finally {
  setLoading(false);
}
};

// ✅ Resend OTP
const handleResendOtp = () => {
  if (timer > 0) return;
  handleSendOtp();
};

return (
  <div style={styles.container}>
    <ToastContainer position="bottom-center" autoClose={2500} />
    <div style={styles.card}>
      <h2 style={styles.title}>Forgot Password</h2>

      {/* Step 1 : Enter Email */}
      {step === 1 && (
        <>
          <input
            style={styles.input}
            type="email"
            placeholder="Enter your email"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
          />
          {error && <p style={styles.error}>{error}</p>}
          <button
            style={styles.button}
            onClick={handleSendOtp}
            disabled={loading}
          >
            {loading ? "Sending..." : "Send OTP"}
          </button>
        </>
      )}
    </div>
  </div>
);

```

```

        </button>
      </>
    )}

/* Step 2 : Verify OTP */
{step === 2 && (
  <>
    <p style={{ marginBottom: 10 }}>
      OTP sent to <b>{email}</b>
    </p>

    <input
      style={styles.input}
      type="text"
      placeholder="Enter OTP"
      value={otp}
      onChange={(e) => setOtp(e.target.value)}
    />{error && <p style={styles.error}>{error}</p>}
    <button
      style={styles.button}
      onClick={handleVerifyOtp}
      disabled={loading}
    >
      {loading ? "Verifying..." : "Verify OTP"}
    </button>

    <button
      style={{
        ...styles.linkBtn,
        color: timer > 0 ? "#888" : "#2E2B5F",
        cursor: timer > 0 ? "not-allowed" : "pointer",
      }}
      onClick={handleResendOtp}
      disabled={timer > 0}
    >
      {timer > 0 ? `Resend OTP in ${timer}s` : "Resend OTP"}
    </button>
  </>
}

/* Step 3 : Reset Password */
{step === 3 && (
  <>
    <input

```

```

        style={styles.input}
        type="password"
        placeholder="Enter new password"
        value={newPassword}
        onChange={(e) => setNewPassword(e.target.value)}
    />
    <input
        style={styles.input}
        type="password"
        placeholder="Confirm new password"
        value={confirmPassword}
        onChange={(e) => setConfirmPassword(e.target.value)}
    />
    {error && <p style={styles.error}>{error}</p>}
    <button
        style={styles.button}
        onClick={handleResetPassword}
        disabled={loading}
    >
        {loading ? "Saving..." : "Reset Password"}
    </button>
    </>
    )}
</div>
</div>
);
}

export default ForgotPassword;

const styles = {
  container: {
    minHeight: "100vh",
    display: "flex",
    alignItems: "center",
    justifyContent: "center",
    backgroundColor: "#f3f4f6",
  },
  card: {
    width: 360,
    padding: 25,
    borderRadius: 12,
    backgroundColor: "#fff",
    boxShadow: "0 4px 14px rgba(0,0,0,0.1)",
  }
};

```

```
    textAlign: "center",
},
title: {
  marginBottom: 20,
  fontWeight:"600",
  fontSize:"30px"
},
input: {
  width: "100%",
  padding: "7px 15px",
  marginBottom: "10px",
  border: "1px solid #ccc",
  borderRadius: "8px",
},
button: {
  width: "60%",
  padding: "6px",
  backgroundColor: "#953ce7",
  border: "1px solid black",
  color: "#fff",
  borderRadius: "8px",
  cursor: "pointer",
  fontWeight: "400",
  marginTop: "10px",
},
linkBtn: {
  background: "none",
  border: "none",
  color: "#2E2B5F",
  marginTop: "10px",
  fontSize: "14px",
},
error: {
  color: "red",
  fontSize: "14px",
  marginBottom: "8px",
},
};
```

Courses Page:

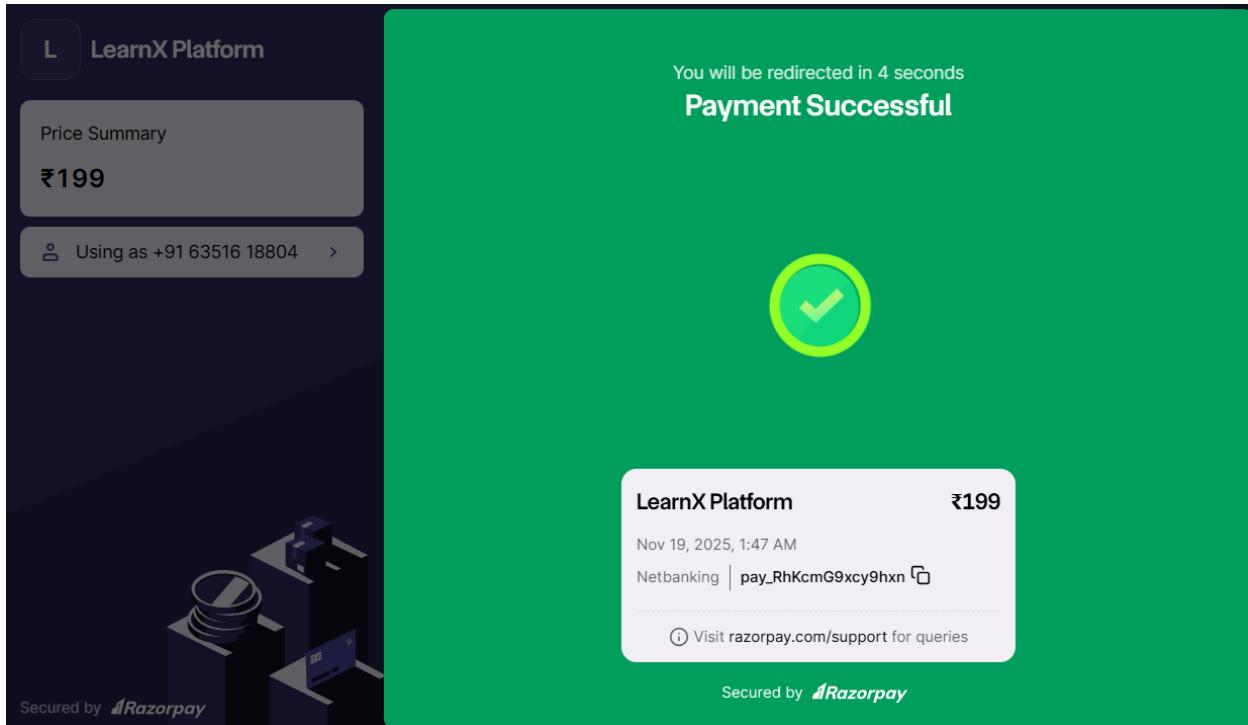
The screenshot shows the LearnX platform's Courses page. At the top, there is a search bar with the placeholder "Search courses...". Below the search bar, there are navigation links for "Home" and "Courses", and a user profile icon. On the left, there is a sidebar titled "Categories" with a list of course types: Next JS, Data Science, Frontend Development, Fullstack Development, MERN Stack Development, Backend Development, Javascript, and Python. There is also a "Clear Filters" button. The main content area displays 11 course cards arranged in two rows. Each card includes the course title, a brief description, the instructor, the number of enrolled students, the difficulty level, the price, and an "Enroll" button.

Course Title	Description	Instructor	Enrolled Students	Difficulty	Price	Action
Mastering Next.js	Learn Next.js step by step with projects	abc	1	Intermediate	Free	Go to Course
Advanced MERN Project	To Add in Resume	abc	0	Intermediate	₹299	Go to Course Unenroll
Mastering Next.js: Part 1	NEXT.js	abc	0	Beginner	₹199	Enroll
Data Science with Python	DATA SCIENCE USING PYTHON	instructor2	0	Intermediate	₹299	Enroll
Frontend Development Bootcamp	Master HTML, CSS, JavaScript, and modern frontend frameworks.	instructor2	0	Beginner	₹149	Enroll
Fullstack Development Project	Build fullstack applications using MERN stack and deploy them.	instructor2	0	Advanced	₹399	Enroll
Mastering Next.js: Part 1	NEXT.js	instructor3	0	Beginner	₹99	Enroll
JavaScript Basics	Learn the fundamentals of JavaScript, including variables, functions, loops, and DOM manipulation.	instructor3	0	Beginner	₹99	Enroll
Advanced Python for Data Science	Master Python libraries like Pandas, NumPy, and Matplotlib for data analysis and visualization.	instructor3	0	Advanced	₹399	Enroll

At the bottom of the page, there are navigation buttons for "Prev", "Page 1 of 2", and "Next".

Enrollment process:

The screenshot shows the LearnX Platform's enrollment process. On the left, there is a sidebar with the "LearnX Platform" logo and a "Price Summary" section showing "₹199". Below this, there is a button for "Using as +91 63516 18804" with a "Next" button. The main content area is titled "Payment Options". It shows four payment methods: "Cards" (with icons for VISA, Mastercard, American Express, and others), "Netbanking" (with icons for SBI, ICICI, Axis, and others), "Wallet" (with icons for Paytm, PhonePe, Google Pay, and others), and "Pay Later" (with icons for Citi, ICICI, and others). To the right of these options is a "Continue" button. At the bottom left, it says "Secured by Razorpay".



Courses.jsx:

```

import { useState, useEffect } from "react";
import api from "../../api/api";
import { Link, useLocation, useNavigate } from "react-router-dom";
import { toast } from "react-toastify";

function Courses() {
  const [courses, setCourses] = useState([]);
  const [categories, setCategories] = useState([]);
  const [selectedCategories, setSelectedCategories] = useState([]);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState("");
  const [currentPage, setCurrentPage] = useState(1);
  const coursesPerPage = 8;

  const location = useLocation();
  const navigate = useNavigate();

  const loggedInUser = JSON.parse(localStorage.getItem("user"));
  const studentId = loggedInUser?._id;

  const [enrollLoadingIds, setEnrollLoadingIds] = useState([]);

```

```

const [enrolledCoursesIds, setEnrolledCoursesIds] = useState({});
const [enrollmentsLoading, setEnrollmentsLoading] = useState(true);

const BASE_URL = import.meta.env.VITE_BASE_URL || "";

useEffect(() => {
  if (!loggedInUser) {
    navigate("/login", { replace: true });
  }
}, [loggedInUser, navigate]);

const fetchCategories = async () => {
  try {
    const res = await api.get("/courses/categories");
    const catArray = Array.isArray(res.data) ? res.data : res.data.categories;
    setCategories(catArray || []);
  } catch (err) {
    console.error(err);
    setCategories([]);
  }
};

const fetchCourses = async (filters = {}) => {
  try {
    setLoading(true);
    setError("");
    const params = { limit: 100, ...filters };
    const res = await api.get("/courses", { params });
    const data = Array.isArray(res.data) ? res.data : res.data.courses || [];

    // Now data already has totalEnrolled
    setCourses(data);
    setCurrentPage(1);
  } catch (err) {
    console.error(err);
    setError("Failed to load courses.");
  } finally {
    setLoading(false);
  }
};

const fetchEnrollments = async () => {
  try {
    if (!studentId) return;
  
```

```

setEnrollmentsLoading(true);
const token = localStorage.getItem("token");
const res = await api.get("/enrollments", {
  headers: { Authorization: `Bearer ${token}` },
});
};

if (res.data.success) {
  const enrollmentMap = {};
  res.data.enrollments.forEach((e) => {
    if (e.course?._id) {
      enrollmentMap[e.course._id] = {
        status: e.status,
        expiryDate: e.expiryDate,
      };
    }
  });
  setEnrolledCoursesIds(enrollmentMap);
}
} catch (err) {
  console.error(err);
} finally {
  setEnrollmentsLoading(false);
}
};

useEffect(() => {
  fetchCategories();
  fetchEnrollments();
}, []);

useEffect(() => {
  if (!Array.isArray(categories) || categories.length === 0) return;
  const queryParams = new URLSearchParams(location.search);
  const categoryNames = queryParams.get("category");
  setSelectedCategories(categoryNames ? categoryNames.split(",") : []);
}, [categories, location.search]);

useEffect(() => {
  const queryParams = new URLSearchParams(location.search);
  const searchQuery = queryParams.get("search") || "";
  const filters = {};
  if (searchQuery.trim() !== "") filters.search = searchQuery;
  if (selectedCategories.length > 0)

```

```

filters.categories = selectedCategories.join(",");
fetchCourses(filters);
}, [selectedCategories, location.search]);

const handleCategoryChange = (catName) => {
  setSelectedCategories((prev) => {
    const newSelected = prev.includes(catName)
      ? prev.filter((c) => c !== catName)
      : [...prev, catName];
    const params = new URLSearchParams(location.search);
    if (newSelected.length > 0) params.set("category", newSelected.join(","));
    else params.delete("category");
    navigate(`?${params.toString()}`, { replace: true });
    return newSelected;
  });
};

const loadRazorpayScript = () => {
  return new Promise((resolve) => {
    const script = document.createElement("script");
    script.src = "https://checkout.razorpay.com/v1/checkout.js";
    script.onload = () => resolve(true);
    script.onerror = () => resolve(false);
    document.body.appendChild(script);
  });
};

const handleEnroll = async (course) => {
  try {
    setEnrollLoadingIds([...prev, course._id]);
    const user = JSON.parse(localStorage.getItem("user"));
    const token = localStorage.getItem("token");

    if (!user || !token) {
      toast.error("Please log in to enroll");
      return;
    }

    // Free course direct enrollment
    if (!course.price || course.price === 0) {
      const { data } = await api.post(
        "/enrollments",

```

```

    { courseId: course._id, amount: 0 },
    { headers: { Authorization: `Bearer ${token}` } }
);
if (data.success) {
  toast.success("Enrolled successfully!");
  fetchEnrollments();
} else toast.error(data.message || "Enrollment failed.");
return;
}

// Paid course via Razorpay
const scriptLoaded = await loadRazorpayScript();
if (!scriptLoaded) {
  toast.error("Razorpay SDK failed to load.");
  return;
}

const { data } = await api.post(
  "/payment/create-order",
  { courseId: course._id, studentId: user._id },
  { headers: { Authorization: `Bearer ${token}` } }
);

if (!data.success) return toast.error(data.message);

const { key, orderId, amount, currency } = data;
const options = {
  key,
  amount: amount * 100,
  currency,
  name: "LearnX Platform",
  description: course.title,
  order_id: orderId,
  handler: async function (response) {
    const verify = await api.post(
      "/payment/verify-payment",
      {
        ...response,
        studentId: user._id,
        courseId: course._id,
      },
      { headers: { Authorization: `Bearer ${token}` } }
    );
  }
};

```

```

        if (verify.data.success) {
            toast.success("Enrollment successful!");
            fetchEnrollments();
        } else toast.error("Payment verification failed!");
    },
    prefill: { name: user.name, email: user.email, contact: user.phone || "" },
    theme: { color: "#2E2B5F" },
};

const razorpay = new window.Razorpay(options);
razorpay.open();
} catch (error) {
    console.error(error);
    toast.error("Enrollment failed");
} finally {
    setEnrollLoadingIds((prev) => prev.filter((id) => id !== course._id));
}
};

const handleUnenroll = async (courseld) => {
    try {
        const token = localStorage.getItem("token");
        if (!token) return toast.error("Please log in first");
        if (!window.confirm("Are you sure you want to unenroll?")) return;
        await api.put(
            `/enrollments/unenroll/${courseld}`,
            {},
            { headers: { Authorization: `Bearer ${token}` } }
        );
        toast.success("Unenrolled successfully");
        fetchEnrollments();
    } catch (err) {
        console.error(err);
        toast.error("Failed to unenroll");
    }
};

const handleReenroll = async (course) => {
    try {
        const user = JSON.parse(localStorage.getItem("user"));
        const token = localStorage.getItem("token");
        if (!user || !token) return toast.error("Please log in first");
        if (!window.confirm("Do you want to re-enroll in this course?")) return;

        if (!course.price || course.price === 0) {

```

```

const { data } = await api.post(
  "/enrollments",
  { courseld: course._id, amount: 0 },
  { headers: { Authorization: `Bearer ${token}` } }
);
if (data.success) {
  toast.success("Re-enrolled successfully!");
  fetchEnrollments();
} else toast.error(data.message);
return;
}

const scriptLoaded = await loadRazorpayScript();
if (!scriptLoaded) return toast.error("Razorpay SDK failed to load.");

const { data } = await api.post(
  "/payment/create-order",
  { courseld: course._id, studentId: user._id },
  { headers: { Authorization: `Bearer ${token}` } }
);
if (!data.success) return toast.error(data.message);

const { key, orderId, amount, currency } = data;
const options = {
  key,
  amount: amount * 100,
  currency,
  name: "LearnX Platform",
  description: `Re-enrollment for ${course.title}`,
  order_id: orderId,
  handler: async function (response) {
    const verify = await api.post(
      "/payment/verify-payment",
      { ...response, studentId: user._id, courseld: course._id },
      { headers: { Authorization: `Bearer ${token}` } }
    );
    if (verify.data.success) {
      toast.success("Re-enrollment successful!");
      fetchEnrollments();
    } else toast.error("Payment verification failed!");
  },
  prefill: { name: user.name, email: user.email, contact: user.phone || "" },
  theme: { color: "#2E2B5F" },
};

```

```

const razorpay = new window.Razorpay(options);
razorpay.open();
} catch (err) {
  console.error(err);
  toast.error("Re-enrollment failed");
}
};

if (enrollmentsLoading)
  return (
    <div className="course-loading">
      <div className="spinner" />
      <p>Loading enrollments...</p>
    </div>
  );
}

const approvedCourses = courses.filter((c) => c.status === "approved");
const indexOfLastCourse = currentPage * coursesPerPage;
const indexOfFirstCourse = indexOfLastCourse - coursesPerPage;
const currentCourses = approvedCourses.slice(indexOfFirstCourse, indexOfLastCourse);
const totalPages = Math.ceil(approvedCourses.length / coursesPerPage);

const handlePageChange = (page) => {
  if (page >= 1 && page <= totalPages) setCurrentPage(page);
};

return (
<div
  style={{
    fontFamily: "Poppins, sans-serif",
    padding: "20px",
    background: "#f8f7ff",
    minHeight: "100vh",
    marginTop: "67px",
  }}
>
  <div style={{ display: "flex", gap: "20px" }}>
    /* Sidebar */
    <div
      style={{
        width: "25%",
        background: "#fff",
        padding: "20px",
        borderRadius: "12px",
      }}
    >

```

```
        boxShadow: "0 4px 10px rgba(46, 43, 95, 0.1)",
    )}
>
<h3 style={{ fontSize: "1.1rem", marginBottom: "12px", color: "#2E2B5F" }}>
    Categories
</h3>
<div style={{ display: "flex", flexDirection: "column", gap: "8px" }}>
    {Array.isArray(categories) && categories.length > 0 ? (
        categories.map((cat) => (
            <label
                key={cat._id}
                style={{
                    display: "flex",
                    alignItems: "center",
                    gap: "8px",
                    cursor: "pointer",
                    color: "#333",
                    fontSize: "0.9rem",
                }}
            >
                <input
                    type="checkbox"
                    checked={selectedCategories.includes(cat.name)}
                    onChange={() => handleCategoryChange(cat.name)}
                    style={{ accentColor: "#2E2B5F" }}
                />
                {cat.name}
            </label>
        ))
    ) : (
        <p style={{ color: "#888", fontSize: "0.9rem" }}>No categories available</p>
    )
    <button
        onClick={() => {
            setSelectedCategories([]);
            const params = new URLSearchParams(location.search);
            params.delete("category");
            navigate(`?${params.toString()}`, { replace: true });
        }}
        style={{
            marginTop: "12px",
            fontSize: "0.9rem",
            color: "#2E2B5F",
            background: "none",
        }}
    >
```

```

        border: "none",
        cursor: "pointer",
        fontWeight: "500",
    )}
>
    Clear Filters
</button>
</div>
</div>

/* Main Content */
<div style={{ flex: 1 }}>
    {loading && <p style={{ color: "#666" }}>Loading courses...</p>}
    {error && <p style={{ color: "red" }}>{error}</p>}

    {!loading && !error && (
        <>
            <h3 style={{ marginBottom: "20px", color: "#2E2B5F" }}>
                {approvedCourses.length} courses found
            </h3>
            {approvedCourses.length === 0 ? (
                <p style={{ color: "#888" }}>No courses found.</p>
            ) : (
                <>
                    <div
                        style={{
                            display: "grid",
                            gridTemplateColumns: "repeat(auto-fill, minmax(240px, 1fr))",
                            gap: "18px",
                        }}
                    >
                        {currentCourses.map((course) => {
                            const enrollment = enrolledCoursesIds[course._id];
                            const isExpired =
                                enrollment?.expiryDate &&
                                new Date(enrollment.expiryDate) < new Date();
                            const isEnrolled =
                                enrollment?.status === "active" ||
                                enrollment?.status === "completed";
                            const isCompleted = enrollment?.status === "completed";
                            const isExpiredOrCancelled =
                                enrollment?.status === "cancelled" || isExpired;

                            return (

```

```
<div
key={course._id}
style={{
  background: "#fff",
  borderRadius: "12px",
  overflow: "hidden",
  boxShadow: "0 4px 8px rgba(46, 43, 95, 0.1)",
  display: "flex",
  flexDirection: "column",
  fontSize: "0.85rem",
  position: "relative",
}}
>
/* Enrolled / Completed / Expired badges */
{isEnrolled && (
<span
style={{
  position: "absolute",
  top: "8px",
  right: "8px",
  background: isCompleted ? "#007bff" : "#28a745",
  color: "fff",
  padding: "2px 6px",
  borderRadius: "6px",
  fontSize: "0.7rem",
  fontWeight: "bold",
}}
>
{isCompleted ? "Completed" : "Enrolled"}
</span>
)}
{isExpiredOrCancelled && (
<span
style={{
  position: "absolute",
  top: "8px",
  right: "8px",
  background: "#FFA500",
  color: "fff",
  padding: "2px 6px",
  borderRadius: "6px",
  fontSize: "0.7rem",
  fontWeight: "bold",
}}
>
```

```

>
  Expired
</span>
)}

<img
  src={
    course.thumbnail
    ? course.thumbnail.startsWith("http")
    ? course.thumbnail
    : `${BASE_URL}${course.thumbnail}`
    : "https://via.placeholder.com/300x140"
  }
  alt={course.title}
  style={{
    width: "100%",
    height: "140px",
    objectFit: "fill",
  }}
/>

<div
  style={{
    padding: "10px",
    flex: 1,
    display: "flex",
    flexDirection: "column",
    justifyContent: "space-between",
  }}
>
  <div>
    <h3
      style={{
        fontSize: "1rem",
        fontWeight: "600",
        marginBottom: "5px",
        color: "#2E2B5F",
      }}
    >
      <Link
        to={`/courses/${course._id}`}
        style={{
          textDecoration: "none",
          color: "#2E2B5F",
        }}
      >
        {course.title}
      </Link>
    </h3>
    <p>${course.description}</p>
  </div>
</div>

```

```
        }
      >
      {course.title}
    </Link>
  </h3>
  <p style={{ fontSize: "0.8rem", color: "#555" }}>
    {course.description}
  </p>
  <p style={{ fontSize: "0.75rem", color: "#777" }}>
    Instructor: {course.instructor?.name || "N/A"}
  </p>
  <p style={{ fontSize: "0.75rem", color: "#777" }}>
    Enrolled Students: {course.totalEnrolled || 0}
  </p>
  <span
    style={{
      display: "inline-block",
      padding: "2px 8px",
      background: "#eee",
      borderRadius: "6px",
      fontSize: "0.75rem",
      color: "#2E2B5F",
      fontWeight: "500",
    }}
  >
    {course.level}
  </span>
</div>

/* Enrollment Button Logic */
<div
  style={{
    marginTop: "8px",
    display: "flex",
    justifyContent: "space-between",
    alignItems: "center",
  }}
>
  <p
    style={{
      fontWeight: "bold",
      color: "#28a745",
      fontSize: "0.85rem",
    }}
  >
```

```

>
{course.price ? `₹${course.price}` : "Free"}
</p>

{isEnrolled ? (
  <div style={{ display: "flex", gap: "6px" }}>
    <button
      onClick={() => navigate(`/courses/${course._id}`)}
      style={{
        background: "#2E2B5F",
        color: "#fff",
        border: "none",
        padding: "6px 10px",
        borderRadius: "6px",
        fontSize: "0.8rem",
        cursor: "pointer",
      }}
    >
      Go to Course
    </button>

    {/* Hide Unenroll button for completed courses */}
    {!isCompleted && (
      <button
        onClick={() => handleUnenroll(course._id)}
        style={{
          background: "#dc3545",
          color: "#fff",
          border: "none",
          padding: "6px 10px",
          borderRadius: "6px",
          fontSize: "0.8rem",
          cursor: "pointer",
        }}
      >
        Unenroll
      </button>
    )}
  </div>
) : isExpiredOrCancelled ? (
  <button
    onClick={() => handleReenroll(course)}
    disabled={enrollLoadingIds.includes(course._id)}
    style={{

```

```

        background: "#007bff",
        color: "#fff",
        border: "none",
        padding: "6px 10px",
        borderRadius: "6px",
        fontSize: "0.8rem",
        cursor: enrollLoadingIds.includes(course._id)
          ? "not-allowed"
          : "pointer",
      }
    >
      {enrollLoadingIds.includes(course._id)
        ? "Re-Enrolling..."
        : "Re-Enroll"}
    </button>
  ) : (
    <button
      onClick={() => handleEnroll(course)}
      disabled={enrollLoadingIds.includes(course._id)}
      style={{
        background: "#28a745",
        color: "#fff",
        border: "none",
        padding: "6px 10px",
        borderRadius: "6px",
        fontSize: "0.8rem",
        cursor: enrollLoadingIds.includes(course._id)
          ? "not-allowed"
          : "pointer",
      }}
    >
      {enrollLoadingIds.includes(course._id)
        ? "Enrolling..."
        : "Enroll"}
    </button>
  )
);
</div>
</div>
</div>
);
})
}
</div>

```

/* Pagination */

```
{totalPages > 1 && (
  <div
    style={{
      marginTop: "25px",
      display: "flex",
      justifyContent: "center",
      gap: "10px",
    }}
  >
    <button
      onClick={() => handlePageChange(currentPage - 1)}
      disabled={currentPage === 1}
      style={{
        padding: "6px 10px",
        background: "#2E2B5F",
        color: "#fff",
        border: "none",
        borderRadius: "6px",
        cursor: currentPage === 1 ? "not-allowed" : "pointer",
      }}
    >
      Prev
    </button>
    <span style={{ fontWeight: "bold", color: "#2E2B5F" }}>
      Page {currentPage} of {totalPages}
    </span>
    <button
      onClick={() => handlePageChange(currentPage + 1)}
      disabled={currentPage === totalPages}
      style={{
        padding: "6px 10px",
        background: "#2E2B5F",
        color: "#fff",
        border: "none",
        borderRadius: "6px",
        cursor:
          currentPage === totalPages ? "not-allowed" : "pointer",
      }}
    >
      Next
    </button>
  </div>
)}
```

```

        )}
      </>
    )}
</div>
</div>
</div>
);
}

export default Courses;

```

Course Details Page:

The screenshot shows the LearnX platform interface. At the top, there's a purple header bar with the 'LearnX' logo, a search bar containing 'Search courses...', and navigation links for 'Home', 'Courses', and a user profile icon.

The main content area displays a course card for 'Advanced MERN Project'. The card includes the course title, a brief description ('Build a full MERN stack application with React, Node, Express, and MongoDB.'), category ('MERN Stack Development'), level ('Intermediate'), price ('\$299'), instructor ('abc'), and progress ('40%'). It also features a progress bar, a 'Continue Lessons' button, and an 'Unenroll' button. To the right of the card is a sidebar with a purple banner titled '6 Best MERN Projects' and an 'Add in Resume' button, along with icons for a laptop and a resume.

Below the course card is a video player showing a preview of a lesson titled 'introduction'. The video frame displays the text 'NEXT.js' with a play button and a timestamp of '0:00 / 0:05'. To the right of the video player is a sidebar with tabs for 'Lessons' and 'Exams'. The 'Lessons' tab is active, showing a list of three lessons: '1. introduction' (marked with a green checkmark), '2. lesson2', and '3. lesson 3'. The 'Exams' tab is shown but not active.

At the bottom of the page, there's a large callout box containing two exam entries. The first entry is 'Exam 1' (marked with a green checkmark) with a duration of '15 mins', 'Questions: 2', and a 'Best: 100%' status. The second entry is 'Exam 2' with a duration of '20 mins', 'Questions: 3', and a '#2' icon.

courseDetails.jsx

```
import { useEffect, useState, useRef } from "react";
import { useParams, useNavigate } from "react-router-dom";
import { toast, ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import api from "../../api/api";
import "../../styles/courseDetail.css";
import { Eye, Download } from "lucide-react";

function CourseDetail() {
  const { id } = useParams();
  const navigate = useNavigate();

  const [course, setCourse] = useState(null);
  const [lessons, setLessons] = useState([]);
  const [exams, setExams] = useState([]);
  const [completedLessons, setCompletedLessons] = useState([]);
  const [selectedLesson, setSelectedLesson] = useState(null);
  const [selectedTab, setSelectedTab] = useState("lessons");
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState("");
  const [enrollLoading, setEnrollLoading] = useState(false);
  const [isEnrolled, setIsEnrolled] = useState(false);
  const [examProgress, setExamProgress] = useState([]);
  const [progress, setProgress] = useState(0);
  const [certificate, setCertificateUrl] = useState(null);
  const [isExpired, setIsExpired] = useState(false);
  const [videoProgress, setVideoProgress] = useState(0);

  const loggedInUser = JSON.parse(localStorage.getItem("user"));
  const studentId = loggedInUser?._id;
  const videoRef = useRef(null);
  const BASE_URL = import.meta.env.VITE_BASE_URL || "";

  useEffect(() => {
    if (!loggedInUser) {
      navigate("/login", { replace: true });
    }
  }, [loggedInUser, navigate]);

  // ----- Fetch Course & Enrollment -----
  useEffect(() => {
```

```

const fetchCourseData = async () => {
  try {
    setLoading(true);
    const res = await api.get(`/courses/${id}`);
    const courseData = res.data || {};
    setCourse(courseData);

    const courseLessons = Array.isArray(courseData.lessons) ? courseData.lessons : [];
    setLessons(courseLessons);
    setExams(Array.isArray(courseData.exams) ? courseData.exams : []);

    // ✅ Fetch enrollment
    if (studentId && courseData?._id) {
      try {
        const enrollRes = await
api.get(`/enrollments/student/${studentId}/course/${courseData._id}`);
        const enrollData = enrollRes.data;

        if (enrollData) {
          setIsEnrolled(enrollData.status === "active" || enrollData.status === "completed");
          setProgress(enrollData.progress || 0);
          setCompletedLessons(enrollData.completedLessons || []);
          setExamProgress(enrollData.examProgress || []);
          setCertificateUrl(enrollData.certificate || null);

          if (
            enrollData.status === "cancelled" ||
            (enrollData.expiryDate && new Date() > new Date(enrollData.expiryDate))
          ) {
            setIsExpired(true);
            setIsEnrolled(false);
          }
        }
      } catch (err) {
        console.error("Enrollment fetch error:", err);
      }
    }

    const firstPreview = courseLessons.find(l => l.isPreviewFree) || courseLessons[0];
    setSelectedLesson(firstPreview || null);
  } catch (err) {
    console.error("Course fetch error:", err);
    setError("Failed to load course details.");
  } finally {

```

```

        setLoading(false);
    }
};

fetchCourseData();
}, [id, studentId]);

// ----- Enroll -----
const handleEnroll = async () => {
    if (!course) return;

    if (course.status !== "approved") {
        toast.warning("Course is not approved yet!");
        return;
    }

    try {
        setEnrollLoading(true);
        const res = await api.post("/enrollments", {
            courseId: id,
            studentId,
            amount: course.price || 0,
        });

        if (res.data.success) {
            toast.success("Successfully enrolled!");
            setIsEnrolled(true);
            setIsExpired(false);
            navigate(`/course/${id}/lessons/${lessons[0]?._id}`);
        }
    } catch (err) {
        console.error("Enroll error:", err);
        toast.error(err.response?.data?.message || "Failed to enroll. Try again.");
    } finally {
        setEnrollLoading(false);
    }
};

// ----- Unenroll -----
const handleUnenroll = async () => {
    if (!isEnrolled) return toast.info("You're not enrolled in this course.");

    if (!window.confirm("Are you sure you want to unenroll from this course?")) return;

```

```

try {
  setEnrollLoading(true);
  const res = await api.put('/enrollments/unenroll/${id}', { studentId });
  if (res.data.success) {
    toast.info("You have been unenrolled successfully!");
    setIsEnrolled(false);
    setIsExpired(true);
    setProgress(0);
    setCompletedLessons([]);
    setCertificateUrl(null);
  }
} catch (err) {
  console.error("Unenroll error:", err);
  toast.error(err.response?.data?.message || "Failed to unenroll.");
} finally {
  setEnrollLoading(false);
}
};

// ----- Re-Enroll -----
const handleReEnroll = async () => {
  try {
    setEnrollLoading(true);
    const res = await api.post("/enrollments", {
      courseId: id,
      studentId,
      amount: course.price || 0,
    });
    if (res.data.success) {
      toast.success("Re-enrolled successfully!");
      setIsEnrolled(true);
      setIsExpired(false);
      navigate(`/course/${id}/lessons/${lessons[0]?._id}`);
    }
  } catch (err) {
    console.error("Re-enroll error:", err);
    toast.error(err.response?.data?.message || "Failed to re-enroll. Try again.");
  } finally {
    setEnrollLoading(false);
  }
};

// ----- Track Video Progress -----

```

```

useEffect(() => {
  const video = videoRef.current;
  if (!video) return;

  const handleTimeUpdate = () => {
    const percent = (video.currentTime / video.duration) * 100;
    setVideoProgress(percent);
  };

  video.addEventListener("timeupdate", handleTimeUpdate);
  return () => video.removeEventListener("timeupdate", handleTimeUpdate);
}, [selectedLesson]);

// ----- Mark Lesson Completed -----
useEffect(() => {
  if (!selectedLesson || !studentId) return;

  const markCompleted = async () => {
    if (completedLessons.includes(selectedLesson._id)) return;

    try {
      await api.post(`/lessons/${selectedLesson._id}/markWatched`, {
        studentId,
        courseid: course?._id,
      });
      setCompletedLessons((prev) => [...prev, selectedLesson._id]);
    }
  }

  const updatedEnroll = await api.get(
    `/enrollments/student/${studentId}/course/${course._id}`
  );
  setProgress(updatedEnroll.data.progress || 0);
  setCertificateUrl(updatedEnroll.data.certificateUrl || null);
} catch (err) {
  console.error("Mark watched error:", err);
}
};

if (selectedLesson.contentType === "video") {
  if (videoProgress >= 90) markCompleted();
} else {
  markCompleted();
}
}, [selectedLesson, videoProgress, completedLessons, studentId, course]);

```

```

const getLessonProgress = (lessonId) => {
  if (completedLessons.includes(lessonId)) return 100;
  if (selectedLesson?._id === lessonId && selectedLesson.contentType === "video") {
    return videoProgress;
  }
  return 0;
};

// ----- Certificate Actions -----
const handleViewCertificate = () => {
  if (!certificate) return toast.info("No certificate available yet.");
  const fullUrl = certificate.startsWith("http") ? certificate : `${BASE_URL}${certificate}`;
  window.open(fullUrl, "_blank");
};

const handleDownloadCertificate = async () => {
  if (!certificate) return toast.info("No certificate available yet.");

  try {
    const fileUrl = certificate.startsWith("http") ? certificate : `${BASE_URL}${certificate}`;
    const response = await fetch(fileUrl, {
      method: "GET",
      headers: { Authorization: `Bearer ${localStorage.getItem("token") || ""}` },
    });

    if (!response.ok) throw new Error("Failed to download file.");

    const blob = await response.blob();
    const downloadUrl = window.URL.createObjectURL(blob);
    const link = document.createElement("a");
    link.href = downloadUrl;
    link.download = `${course.title?.replace(/\s+/g, "_")}_Certificate.pdf`;
    document.body.appendChild(link);
    link.click();
    document.body.removeChild(link);
    window.URL.revokeObjectURL(downloadUrl);
    toast.success("Certificate downloaded successfully!");
  } catch (err) {
    console.error("Error downloading certificate:", err);
    toast.error("Failed to download certificate.");
  }
};

// ----- JSX -----

```

```

if (loading)
  return (
    <div className="course-loading">
      <div className="spinner" />
      <p>Loading course details...</p>
    </div>
  );
}

if (error) return <p>{error}</p>;
if (!course) return <p>Course not found.</p>

return (
  <div className="course-detail-container">
    <ToastContainer position="bottom-center" autoClose={2500} />
    {/* ----- HEADER ----- */}
    <div className="course-header">
      <div className="course-header-left">
        <h1>{course.title || "Untitled Course"}</h1>
        <p>{course.description || "No description available."}</p>
        <p>
          Category: <span>{course.category?.name ?? "N/A"}</span> | Level:{" "}
          <span>{course.level ?? "N/A"}</span>
        </p>
        <p>
          Price: <span>{course.price > 0 ? `₹${course.price}` : "Free"}</span>
        </p>
        <p>
          Instructor: <span>{course.instructor?.name ?? "N/A"}</span>
        </p>
      </div>
    <div className="course-progress-container">
      <p style={{ marginBottom: "5px", color: "#6f42c1" }}>
        Progress: <strong>{progress}%</strong>{" "}
        {progress === 100 && <span>✓</span>}
      </p>
      <div className="course-progress-bar">
        <div
          className="course-progress-fill"
          style={{ width: `${progress}%` }}
        ></div>
      </div>
    <div>
      {progress === 100 && (

```

```

<div className="course-completed-actions">
  {certificate ? (
    <div className="certificate-buttons">
      <button
        className="certificate-icon-btn"
        onClick={handleViewCertificate}
        title="View Certificate"
      >
        <Eye size={20} />
      </button>
      <button
        className="certificate-icon-btn"
        onClick={handleDownloadCertificate}
        title="Download Certificate"
      >
        <Download size={20} />
      </button>
    </div>
  ) : (
    <p>No certificate available yet.</p>
  )}
</div>
)}
{isEnrolled ? (
  <div className="enroll-actions">
    <button
      className="enroll-button"
      onClick={() => {
        const firstIncomplete =
          lessons.find((l) => !completedLessons.includes(l._id)) || lessons[0];
        navigate(`/course/${id}/lessons/${firstIncomplete?._id}`);
      }}
      disabled={enrollLoading}
    >
      Continue Lessons
    </button>
    <button
      className="unenroll-button"
      onClick={handleUnenroll}
      disabled={enrollLoading}
    >

```

```

        >
      Unenroll
    </button>
  </div>
) : (
  <button
    className="enroll-button"
    onClick={isExpired ? handleReEnroll : handleEnroll}
    disabled={enrollLoading}
  >
    {enrollLoading
      ? "Processing..."
      : isExpired
      ? `Re-Enroll ${course.price > 0 ? ` for ₹${course.price}` : ""}`
      : `Enroll ${course.price > 0 ? ` for ₹${course.price}` : ""}`}
    </button>
  )
</div>

<div className="course-header-right">
  {course.thumbnail ? (
    <img
      src={
        course.thumbnail.startsWith("http")
          ? course.thumbnail
          : `${BASE_URL}${course.thumbnail}`
      }
      alt={course.title}
      style={{ width: "100%", height: "230px", objectFit: "cover" }}
    />
  ) : (
    <div className="no-image">No Image</div>
  )}
</div>
</div>
/* ----- Lessons & Exams ----- */
<div className="course-body">
  <div className="lesson-player">
    {selectedTab === "lessons" ? (
      selectedLesson ? (
        <>
        <h3>
          {selectedLesson.title || "Untitled Lesson"} {" "}
          {selectedLesson.isPreviewFree && !isEnrolled && (

```

```

        <span className="free-preview">Free Preview</span>
    )}
</h3>
{selectedLesson.contentType === "video" &&
 selectedLesson.fileUrl ? (
<video ref={videoRef} controls>
    <source
        src={
            selectedLesson.fileUrl.startsWith("http")
                ? selectedLesson.fileUrl
                : `${BASE_URL}${selectedLesson.fileUrl}`
        }
        type="video/mp4"
    />
</video>
) : selectedLesson.contentType === "pdf" &&
 selectedLesson.fileUrl ? (
<iframe
    src={
        selectedLesson.fileUrl.startsWith("http")
            ? selectedLesson.fileUrl
            : `${BASE_URL}${selectedLesson.fileUrl}`
    }
    title={selectedLesson.title}
/>
) : (
<p>
    {selectedLesson.description || 
        "No content available for this lesson."}
</p>
)}
</>
) : (
<p>Select a lesson to view.</p>
)
) : (
<div>
    <h3>Exam Section</h3>
    <p>Click an exam to start.</p>
</div>
)}
</div>

/* Lesson List */

```

```

<div className="lesson-list">
  <div className="tab-toggle">
    <button
      className={selectedTab === "lessons" ? "active-tab" : ""}
      onClick={() => setSelectedTab("lessons")}
    >
      Lessons
    </button>
    <button
      className={selectedTab === "exams" ? "active-tab" : ""}
      onClick={() => setSelectedTab("exams")}
    >
      Exams
    </button>
  </div>

  {selectedTab === "lessons" ? (
    <ul>
      {lessons.length > 0 ? (
        lessons.map((lesson, idx) => {
          const canAccess = isEnrolled || lesson.isPreviewFree;
          const isActive = selectedLesson?._id === lesson._id;
          const progress = getLessonProgress(lesson._id);
          const isCompleted = completedLessons.includes(lesson._id);

          return (
            <li
              key={lesson._id || idx}
              onClick={() => canAccess && setSelectedLesson(lesson)}
              className={`${isActive ? "active" : ""} ${!canAccess ? "disabled" : ""} ${lesson.isPreviewFree && !isEnrolled
                ? "free-preview"
                : ""}`}
            >
              <div className="lesson-title">
                {idx + 1}. {lesson.title || "Untitled Lesson"}
                {lesson.isPreviewFree && !isEnrolled ? "(Free)" : ""}
                <div className="lesson-progress-bar">
                  <div
                    className="lesson-progress"
                    style={{ width: `${progress}%` }}
                  >
                    {isCompleted && <span className="tick-mark">✓</span>}
                  </div>
                </div>
              </div>
            </li>
          );
        })
      )
    )
  )
)

```

```

        </div>
      </div>
    </div>
  </li>
);
})
) : (
  <p>No lessons available.</p>
)
</ul>
) : (
<div className="exam-list">
{exams.length > 0 ? (
  exams.map((exam, idx) => {
    const title = exam.title || `Exam ${idx + 1}`;
    const duration = exam.duration ?? 0;
    const questionsCount = Array.isArray(exam.questions)
      ? exam.questions.length
      : 0;

    const progress = examProgress.find((p) => p.examId === exam._id);
    const isCompleted = progress?.isCompleted;
    const bestScore = progress?.bestScore ?? null;

    const handleExamClick = () => {
      if (!isEnrolled) {
        alert("Please enroll in the course to attempt exams.");
        return;
      }
      navigate(`/course/${id}/exam/${exam._id}`);
    };

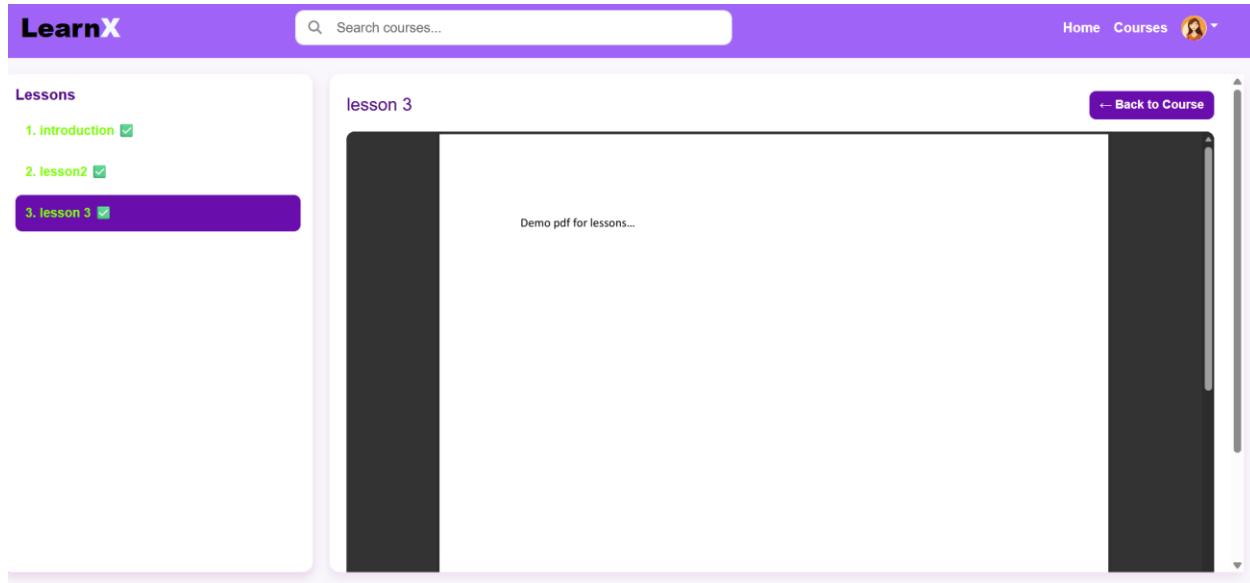
    return (
      <div
        key={exam._id || idx}
        className={`exam-card ${isCompleted ? "completed-exam" : ""} ${!isEnrolled ?
"disabled-exam" : ""}
      `}
      onClick={handleExamClick}
      style={{
        cursor: isEnrolled ? "pointer" : "not-allowed",
        opacity: isEnrolled ? 1 : 0.6,
      }}
    >

```

```
<div className="exam-card-header">
  <h4>
    {title} {isCompleted && <span className="exam-tick"> ✓ </span>}
  </h4>
  <span className="exam-number">#{idx + 1}</span>
</div>
<div className="exam-card-details">
  <p>{duration} mins</p>
  <p>
    <strong>Questions:</strong> {questionsCount}
  </p>
  {bestScore !== null && (
    <p className="exam-score">
      <strong>Best:</strong> {bestScore}%
    </p>
  )}
</div>
</div>
);
})
) : (
  <p className="no-exams">No exams available.</p>
)
</div>
)
</div>
</div>
</div>
);
}

export default CourseDetail;
```

Lessons Page:



Lessons.jsx:

```
import { useEffect, useState, useRef } from "react";
import { useParams, useNavigate } from "react-router-dom";
import api from "../../api/api";
import "../../styles/lessons.css";

function Lesson() {
  const { courseId, lessonId } = useParams();
  const navigate = useNavigate();

  const [course, setCourse] = useState(null);
  const [lessons, setLessons] = useState([]);
  const [currentLesson, setCurrentLesson] = useState(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState("");
  const [completedLessons, setCompletedLessons] = useState([]);
  const [isEnrolled, setIsEnrolled] = useState(false);
  const [videoProgress, setVideoProgress] = useState(0);

  const videoRef = useRef(null);
  const loggedInUser = JSON.parse(localStorage.getItem("user"));
  const studentId = loggedInUser?._id;
```

```

// ✅ Ensure consistent base URL
const BASE_URL = import.meta.env.VITE_BASE_URL?.replace(/\//g, "");

// Fetch data
useEffect(() => {
  const fetchLessonData = async () => {
    try {
      setLoading(true);
      const res = await api.get(`/courses/${courseld}`);
      const courseData = res.data;
      setCourse(courseData);
      setLessons(courseData.lessons || []);
    }
    const enrollmentRes = await api.get("/enrollments");
    const enrollments = Array.isArray(enrollmentRes.data.enrollments)
      ? enrollmentRes.data.enrollments
      : [];
    const enrolled = enrollments.some(e => e.course?._id === courseld);
    setIsEnrolled(enrolled);

    const progressRes = await api.get(`/${studentId}/completedLessons`);
    setCompletedLessons(Array.isArray(progressRes.data) ? progressRes.data : []);

    const lesson =
      courseData.lessons.find(l => l._id === lessonId) || courseData.lessons[0];
    if (lesson) {
      setCurrentLesson(lesson);
      localStorage.setItem(`lastLesson_${courseld}`, lesson._id);
    } else {
      setError("Lesson not found in this course.");
    }
  } catch (err) {
    console.error("Lesson fetch error:", err);
    setError("Failed to load lesson details.");
  } finally {
    setLoading(false);
  }
};

fetchLessonData();
}, [courseld, lessonId, studentId]);

// Track video progress
useEffect(() => {

```

```

const video = videoRef.current;
if (!video) return;

const handleTimeUpdate = () => {
  const percent = (video.currentTime / video.duration) * 100;
  setVideoProgress(percent);
};

video.addEventListener("timeupdate", handleTimeUpdate);
return () => video.removeEventListener("timeupdate", handleTimeUpdate);
}, [currentLesson]);

// Auto mark lesson as watched
useEffect(() => {
  if (!currentLesson) return;

  const markAsCompleted = async () => {
    if (completedLessons.includes(currentLesson._id)) return;
    try {
      await api.post(`/lessons/${currentLesson._id}/markWatched`, {
        studentId,
        courseId: course._id,
      });
      setCompletedLessons(prev => [...prev, currentLesson._id]);
    } catch (err) {
      console.error("Mark watched error:", err);
    }
  };
  if (currentLesson.contentType?.toLowerCase() === "video") {
    if (videoProgress >= 90 && isEnrolled) markAsCompleted();
  } else {
    if (isEnrolled) markAsCompleted();
  }
}, [currentLesson, videoProgress, completedLessons, studentId, course, isEnrolled]);

const handleNextLesson = () => {
  const index = lessons.findIndex(l => l._id === currentLesson._id);
  if (index >= 0 && index < lessons.length - 1) {
    const nextLessonId = lessons[index + 1]._id;
    navigate(`course/${courseId}/lessons/${nextLessonId}`);
  } else alert("You have reached the last lesson!");
};

```

```

const handlePrevLesson = () => {
  const index = lessons.findIndex(l => l._id === currentLesson._id);
  if (index > 0) {
    const prevLessonId = lessons[index - 1]._id;
    navigate(`/course/${courseId}/lessons/${prevLessonId}`);
  } else alert("This is the first lesson!");
};

const handleBackToCourse = () => {
  navigate(`/courses/${courseId}`);
};

if (loading)
  return (
    <div className="course-loading">
      <div className="spinner" />
      <p>Loading lessons...</p>
    </div>
  );
}

if (error) return <p style={{ color: "red" }}>{error}</p>;
if (!currentLesson) return <p>Lesson not found.</p>

const type = currentLesson.contentType?.toLowerCase().trim();
const canPlay = currentLesson.isPreviewFree || isEnrolled;

// ✅ Construct correct file URL
const fileUrl = currentLesson.fileUrl
  ? currentLesson.fileUrl.startsWith("http")
    ? currentLesson.fileUrl
    : `${BASE_URL}${currentLesson.fileUrl.startsWith("/") ? "" : "/"}${currentLesson.fileUrl}`
  : null;

// Disable right-click for media
const disableRightClick = (e) => e.preventDefault();

return (
  <div className="lesson-page">
    {/* Sidebar */}
    <div className="lesson-list-sidebar">
      <h3>Lessons</h3>
      <ul>
        {lessons.map((lesson, idx) => {
          const isActive = lesson._id === currentLesson._id;

```

```

const isCompleted = completedLessons.includes(lesson._id);
const locked = !lesson.isPreviewFree && !isEnrolled;

return (
  <li
    key={lesson._id}
    className={`${$isActive ? "active" : ""} ${$isCompleted ? "completed" : ""} ${$locked ? "locked" : ""}`}
    onClick={() => !locked && setCurrentLesson(lesson)}
  >
    {idx + 1}. {lesson.title}{" "}
    {lesson.isPreviewFree && !isEnrolled && (
      <span className="preview-locked">Preview</span>
    )}
    {isCompleted && <span> ✓ </span>}
  </li>
);
})
</ul>
</div>

/* Main Content */
<div className="lesson-content">
  <div className="lesson-header">
    <h2>{currentLesson.title}</h2>
    <button className="back-button" onClick={handleBackToCourse}>
      ← Back to Course
    </button>
  </div>

  /* Lesson Display */
  {canPlay ? (
    type === "video" ? (
      <div onContextMenu={disableRightClick}>
        <video
          ref={videoRef}
          className="lesson-video"
          controls
          controlsList="nodownload noreMOTEPLAYBACK"
          disablePictureInPicture
          onContextMenu={disableRightClick}
        >
          <source src={fileUrl} type="video/mp4" />
        Your browser does not support the video tag.
    
```

```

        </video>
    </div>
) : type === "pdf" ? (
    fileUrl ? (
        <iframe
            src={`${fileUrl}#toolbar=0`}
            title={currentLesson.title}
            className="lesson-pdf"
            onContextMenu={disableRightClick}
            allow="fullscreen"
            style={{
                width: "100%",
                height: "80vh",
                border: "none",
            }}
        />
    ) : (
        <p style={{ color: "#888" }}>PDF not uploaded yet.</p>
    )
) : type === "text" ? (
    <div className="lesson-text">
        {currentLesson.description || "Description not available."}
    </div>
) : (
    <p>Unsupported lesson type.</p>
)
) : (
    <p style={{ color: "#888" }}>This lesson is locked. Enroll to access.</p>
)

<div className="lesson-navigation">
    <button
        onClick={handlePrevLesson}
        disabled={lessons.findIndex(l => l._id === currentLesson._id) === 0}
    >
        ← Previous
    </button>
    <button
        onClick={handleNextLesson}
        disabled={lessons.findIndex(l => l._id === currentLesson._id) === lessons.length - 1}
    >
        Next →
    </button>
</div>

```

```

        </div>
        </div>
    );
}

export default Lesson;

```

Exams Page:

The screenshot shows the LearnX platform's exam interface. At the top, there's a purple header bar with the 'LearnX' logo, a search bar, and navigation links for 'Home', 'Courses', and a user profile icon. Below the header, on the left, is a sidebar for 'Advanced MERN Project' with a 'Exam 2' tab selected. This tab has a green checkmark next to 'Exam 1'. The main content area displays 'Exam 2' details: 'Duration: 20 minutes' and 'Time Left: 19:56'. A progress bar indicates 'Attempt 1 of 3'. The first question, '1. Que 1', is shown with four options: 'op1', 'op2', 'op3 - correct', and 'op4'. The second question, '2. Que 2', also has four options: 'op1 - correct', 'op2', 'op3', and 'op4'. The third question, '3. Que 3', has four options: 'op1', 'op2', 'op3', and 'op4 - correct'. At the bottom of the page is a purple 'Submit Exam' button.

Exams.jsx

```

import React, { useEffect, useState } from "react";
import { useParams, useNavigate } from "react-router-dom";
import api from "../../api/api";
import "../../styles/lessons.css";

```

```

export default function Exams() {
  const { courseid, examId } = useParams();
  const navigate = useNavigate();

  const [course, setCourse] = useState(null);
  const [exams, setExams] = useState([]);
  const [exam, setExam] = useState(null);
  const [answers, setAnswers] = useState({});
  const [timeLeft, setTimeLeft] = useState(0);
  const [submitted, setSubmitted] = useState(false);
  const [result, setResult] = useState(null);
  const [attemptNumber, setAttemptNumber] = useState(0);
  const [examStatuses, setExamStatuses] = useState({});
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState("");

  const loggedInUser = JSON.parse(localStorage.getItem("user"));
  const studentId = loggedInUser?._id;

  // ✅ Fetch exams and progress
  useEffect(() => {
    const fetchExams = async () => {
      try {
        setLoading(true);
        const [courseRes, examsRes] = await Promise.all([
          api.get(`/courses/${courseid}`),
          api.get(`/exams/course/${courseid}`),
        ]);

        setCourse(courseRes.data);
        setExams(examsRes.data);

        // Fetch completion status
        const progressStatuses = {};
        await Promise.all(
          examsRes.data.map(async (ex) => {
            try {
              const res = await api.get(`exams/${ex._id}/result/${studentId}`);
              if (res.data?.isCompleted) progressStatuses[ex._id] = "completed";
            } catch {
              progressStatuses[ex._id] = "pending";
            }
          })
        );
      };
    };
  });
}

```

```

setExamStatuses(progressStatuses);

if (examId) {
  const singleExamRes = await api.get(`/exams/${examId}`);
  setExam(singleExamRes.data);
  setTimeLeft(singleExamRes.data.duration * 60);

  try {
    const attemptRes = await api.get(`/exams/${examId}/result/${studentId}`);
    if (attemptRes.data) {
      setAttemptNumber(attemptRes.data.attemptNumber || 0);
      setResult(attemptRes.data);
    }
  } catch (err) {
    if (err.response?.status === 404) setAttemptNumber(0);
  }
}

} catch (err) {
  console.error("Error loading exams:", err);
  setError("Failed to load exams.");
} finally {
  setLoading(false);
}
};

fetchExams();
}, [courseId, examId]);

// Timer countdown
useEffect(() => {
  if (!timeLeft || submitted) return;
  const timer = setInterval(() => {
    setTimeLeft((prev) => {
      if (prev <= 1) {
        clearInterval(timer);
        handleSubmit();
        return 0;
      }
      return prev - 1;
    });
  }, 1000);
  return () => clearInterval(timer);
}, [timeLeft, submitted]);

const handleAnswerChange = (questionId, option) => {

```

```

setAnswers((prev) => ({ ...prev, [questionId]: option }));
};

const formatTime = (sec) => {
  const m = Math.floor(sec / 60);
  const s = sec % 60;
  return `${m}:${s < 10 ? "0" : ""}${s}`;
};

const handleSubmit = async () => {
  if (submitted || attemptNumber >= 3) return;
  setSubmitted(true);

  try {
    // ✅ Calculate score
    let correct = 0;
    exam.questions.forEach((q) => {
      if (answers[q._id] === q.correctAnswer) correct++;
    });
    const score = Math.round((correct / exam.questions.length) * 100);

    // ✅ Prepare data for backend
    const payload = {
      studentId,
      courseId,
      examId,
      score,
    };
    console.log("Submitting exam:", payload);

    // ✅ Submit to backend
    const res = await api.post('/exams/submit', payload);

    setResult(res.data);
    setAttemptNumber(res.data.attemptNumber || attemptNumber + 1);

    if (res.data.isCompleted) {
      setExamStatuses((prev) => ({ ...prev, [examId]: "completed" }));
    }
  }

  // ✅ Refresh result
  const refreshed = await api.get(`/exams/${examId}/result/${studentId}`);

```

```

if (refreshed.data) {
  setResult(refreshed.data);
  setAttemptNumber(refreshed.data.attemptNumber || attemptNumber + 1);
  if (refreshed.data.isCompleted) {
    setExamStatuses((prev) => ({ ...prev, [examId]: "completed" }));
  }
}
} catch (err) {
  console.error("✖ Error submitting exam:", err);
  alert(err.response?.data?.message || "Error submitting exam. Try again.");
  setSubmitted(false);
}
};

const handleRetry = async () => {
  if (attemptNumber >= 3 || result?.isCompleted) return;
  setSubmitted(false);
  setResult(null);
  setAnswers({});
  setTimeLeft(exam.duration * 60);
};

const handleBackToCourse = () => navigate(`/courses/${courseId}`);
const handleExamSelect = (id) => navigate(`/course/${courseId}/exam/${id}`);

if (loading)
  return (
    <div className="course-loading">
      <div className="spinner" />
      <p>Loading exams...</p>
    </div>
  );
}

if (error) return <p style={{ color: "red" }}>{error}</p>

// --- Sidebar progress indicator ---
const ProgressBar = ({ current }) => {
  const width = (current / 3) * 100;
  return (
    <div style={{ marginTop: "10px" }}>
      <div
        style={{
          background: "#e0e0e0",
          borderRadius: "6px",

```

```

        height: "10px",
        width: "100%",
    })
>
<div
    style={{
        background: "#6f42c1",
        width: `${width}`,
        height: "10px",
        borderRadius: "6px",
        transition: "width 0.3s ease",
    }}
/>
</div>
<small style={{ color: "#6f42c1" }}>
    Attempt {current} of 3
</small>
</div>
);
};

// --- Exam List Page ---
if (!examId || !exam) {
    return (
        <div className="lesson-page">
            <div className="lesson-list-sidebar">
                <h3>{course?.title || "Course"}</h3>
                <ul>
                    {exams.length > 0 ? (
                        exams.map((ex, i) => (
                            <li
                                key={ex._id}
                                onClick={() => handleExamSelect(ex._id)}
                                className="sidebar-item"
                            >
                                {examStatuses[ex._id] === "completed" ? "✓" : ""}
                                Exam {i + 1}: {ex.title}
                            </li>
                        ))
                    ) : (
                        <li>No exams available</li>
                    )}
                </ul>
            </div>
        </div>
    );
}


```

```

<div className="lesson-content">
  <h2>Course Exams</h2>
  <button className="back-button" onClick={handleBackToCourse}>
    ← Back to Course
  </button>
  <p>Select an exam from the left sidebar to start.</p>
</div>
</div>
);
}

// --- Max Attempts / Completed View ---
if ((attemptNumber >= 3 || result?.isCompleted) && !submitted) {
  return (
    <div className="lesson-page">
      <div className="lesson-list-sidebar">
        <h3>{course?.title}</h3>
        <ul>
          {exams.map((ex) => (
            <li
              key={ex._id}
              onClick={() => handleExamSelect(ex._id)}
              className={`${'sidebar-item '}{ex._id === examId ? 'active' : ''}`}
            >
              {examStatuses[ex._id] === "completed" ? "✓ " : " "}
              {ex.title}
            </li>
          )));
        </ul>
      </div>
    </div>

    <div className="lesson-content">
      <div className="lesson-header">
        <h2>{exam.title}</h2>
        <button className="back-button" onClick={handleBackToCourse}>
          ← Back to Course
        </button>
      </div>

      <div className="result-card" style={{ marginTop: "30px" }}>
        <h3>
          {result?.isCompleted ? "✓ Exam Completed" : "✗ Max Attempts Reached"}
        </h3>
        <ProgressBar current={attemptNumber} />
      </div>
    </div>
  );
}

```

```

<p>
  <b>Last Score:</b> {result?.score ?? "N/A"}%
</p>
<p>
  <b>Best Score:</b> {result?.bestScore ?? "N/A"}%
</p>
</div>
</div>
</div>
);
}

// --- Result View ---
if (submitted && result) {
  return (
    <div className="lesson-page">
      <div className="lesson-list-sidebar">
        <h3>{course?.title}</h3>
        <ul>
          {exams.map((ex) => (
            <li
              key={ex._id}
              onClick={() => handleExamSelect(ex._id)}
              className={`sidebar-item ${ex._id === examId ? "active" : ""}`}
            >
              {examStatuses[ex._id] === "completed" ? "✓" : " "}
              {ex.title}
            </li>
          )))
        </ul>
      </div>

      <div className="lesson-content">
        <div className="lesson-header">
          <h2>{exam.title}</h2>
          <button className="back-button" onClick={handleBackToCourse}>
            ← Back to Course
          </button>
        </div>

        <div className="result-card" style={{ marginTop: "30px" }}>
          <h3>🎯 Exam Result</h3>
          <ProgressBar current={attemptNumber} />
          <p>

```

```

<b>Score:</b> {result.score}%
</p>
<p>
  <b>Attempt:</b> {result.attemptNumber}/3
</p>
<p>
  <b>Best Score:</b> {result.bestScore}%
</p>

{result.score >= 60 ? (
  <p style={{ color: "green", fontWeight: "bold" }}>  Passed!</p>
) : (
  <p style={{ color: "red", fontWeight: "bold" }}>  Not Passed</p>
)}

{!result.isCompleted && attemptNumber < 3 && (
  <button
    onClick={handleRetry}
    style={{
      background: "#6f42c1",
      color: "#fff",
      border: "none",
      padding: "10px 20px",
      borderRadius: "6px",
      cursor: "pointer",
      marginTop: "15px",
    }}
  >
     Retry Exam
  </button>
)
}

</div>
</div>
</div>
);

}

// --- Exam Attempt View ---
return (
  <div className="lesson-page">
    <div className="lesson-list-sidebar">
      <h3>{course?.title}</h3>
      <ul>

```

```

{exams.map((ex) => (
  <li
    key={ex._id}
    onClick={() => handleExamSelect(ex._id)}
    className={`sidebar-item ${ex._id === examId ? "active" : ""}`}
  >
    {examStatuses[ex._id] === "completed" ? "✓ " : " "}
    {ex.title}
  </li>
))
</ul>
</div>

<div className="lesson-content">
  <div className="lesson-header">
    <h2>{exam.title}</h2>
    <button className="back-button" onClick={handleBackToCourse}>
      ← Back to Course
    </button>
  </div>

  <div className="exam-info">
    <p>
      <b>Duration:</b> {exam.duration} minutes
    </p>
    <p>
      <b>Time Left:</b>{" "}
      <span style={{ color: timeLeft < 60 ? "red" : "green" }}>
        {formatTime(timeLeft)}
      </span>
    </p>
    <ProgressBar current={attemptNumber + 1} />
  </div>

  <div className="exam-questions">
    {exam.questions.map((q, index) => (
      <div
        key={q._id || index}
        className="question-card"
        style={{
          border: "1px solid #ddd",
          padding: "15px",
          borderRadius: "10px",
          marginBottom: "15px",
        }}
      >
        {q.question}
      </div>
    ))
  </div>
</div>

```

```

        backgroundColor: "#fafafa",
    )}
>
<h4>
  {index + 1}. {q.questionText}
</h4>
{q.options.map((opt, i) => (
  <label key={i} style={{ display: "block", margin: "6px 0" }}>
    <input
      type="radio"
      name={`q-${index}`}
      value={opt}
      checked={answers[q._id] === opt}
      onChange={() => handleAnswerChange(q._id, opt)}
    />" "}
    {opt}
  </label>
))
)
</div>
))
</div>

<div className="lesson-navigation">
  <button
    onClick={handleSubmit}
    disabled={submitted}
    style={{
      background: "#6f42c1",
      color: "#fff",
      border: "none",
      padding: "10px 20px",
      borderRadius: "6px",
      cursor: "pointer",
    }}
  >
    Submit Exam
  </button>
</div>
</div>
</div>
);
}

```

Student Profile Page:

The screenshot shows a student profile page with a purple header bar. The header contains the logo "LearnX", a search bar with the placeholder "Search courses...", and navigation links for "Home", "Courses", and a user icon. The main content area has a white background with a rounded rectangle containing the profile information. At the top of this section is a title "Student Profile" with three buttons: "View" (highlighted in purple), "Edit", and "Change Password". Below this is a heading "Profile Details" followed by a circular profile picture of a woman with brown hair. Underneath the picture, the following details are listed: Name: student1, Email: hmsonline3111@gmail.com, Role: student, education: bca,mca, interests: ai,ml. A horizontal line separates this from the next section, "Enrolled Courses", which lists three items: Mastering Next.js, Next JS Fundamentals, and Advanced MERN Project.

The image displays two versions of the student profile page. The left version is titled "Student Profile" and includes "Edit" and "Change Password" buttons. It features a "Edit Profile" section with a circular profile picture of a woman, a "Profile Picture" input field with a "Choose File" button and a "No file chosen" message, and four input fields for "Name" (student1), "education" (bca,mca), and "interests" (ai,ml). A "Save" button is located at the bottom. The right version is also titled "Student Profile" and includes "View", "Edit", and "Change Password" buttons. It features a "Change Password" section with three input fields for "Old Password", "New Password", and "Confirm Password", and a "Update Password" button at the bottom.

Profile.jsx

```
import React, { useEffect, useState } from "react";
```

```

import { useNavigate } from "react-router-dom";
import api from "../../api/api";
import "../../styles/profile.css";

export default function Profile() {
  const navigate = useNavigate();
  const [user, setUser] = useState(null);
  const [profile, setProfile] = useState(null);
  const [loading, setLoading] = useState(true);
  const [mode, setMode] = useState("view");
  const [formData, setFormData] = useState({});
  const [userData, setUserData] = useState({});
  const [passwords, setPasswords] = useState({
    oldPassword: "",
    newPassword: "",
    confirmPassword: ""
  });
  const [profilePic, setProfilePic] = useState(null);

  const BASE_URL = import.meta.env.VITE_BASE_URL || "";
  const DEFAULT_PROFILE = `${BASE_URL}/uploads/default.png`;

  // ----- Fetch Profile -----
  useEffect(() => {
    const fetchProfile = async () => {
      try {
        const res = await api.get("/profile");
        const userDataFromApi = res.data.user;
        const profileDataFromApi = res.data.profile;

        // ✅ Build full URL only for displaying
        const fullPicUrl = userDataFromApi.profilePic
          ? userDataFromApi.profilePic.startsWith("http")
            ? userDataFromApi.profilePic
            : `${BASE_URL}${userDataFromApi.profilePic}`
          : DEFAULT_PROFILE;

        setUser(userDataFromApi);
        setProfile(profileDataFromApi);
        setUserData({
          name: userDataFromApi.name,
          email: userDataFromApi.email
        });
        setProfilePic(fullPicUrl);
      } catch (error) {
        console.error("Error fetching profile:", error);
      }
    };
    fetchProfile();
  }, []);
}

```

```

    setFormData(profileDataFromApi || {});
} catch (err) {
  console.error("Fetch profile error:", err.response?.data || err);
} finally {
  setLoading(false);
}
};

fetchProfile();
}, [BASE_URL]);

// ----- Handlers -----
const handleUserChange = (e) =>
 (userData{ ...userData, [e.target.name]: e.target.value });

const handleFormChange = (e) =>
  setFormData({ ...formData, [e.target.name]: e.target.value });

const handleProfilePicChange = (e) => {
  if (e.target.files && e.target.files[0]) setProfilePic(e.target.files[0]);
};

const handleSave = async () => {
try {
  const data = new FormData();
  Object.entries(userData).forEach(([key, value]) => data.append(key, value));
  Object.entries(formData).forEach(([key, value]) => {
    if (!["_id", "user", "__v", "createdAt", "updatedAt", "coursesCreated",
"enrolledCourses"].includes(key)) {
      data.append(key, Array.isArray(value) ? value.join(",") : value);
    }
  });
}

if (profilePic instanceof File) data.append("profilePic", profilePic);

const res = await api.put("/profile", data, {
  headers: { "Content-Type": "multipart/form-data" },
});

const updatedPic = res.data.user.profilePic
? res.data.user.profilePic.startsWith("http")
? res.data.user.profilePic
: `${BASE_URL}${res.data.user.profilePic}`
: DEFAULT_PROFILE;

```

```

// ✅ Update local state
setUser((prev) => ({
  ...prev,
  ...userData,
  profilePic: updatedPic,
}));
setProfile((prev) => ({ ...prev, ...formData }));
setProfilePic(updatedPic);

// ✅ Update App-level state and localStorage
if (setUser) {
  const updatedUser = { ...res.data.user, profilePic: updatedPic };
  localStorage.setItem("user", JSON.stringify(updatedUser));
  setUser(updatedUser);
}

alert("Profile updated successfully!");
setMode("view");
} catch (err) {
  console.error("Update profile error:", err.response?.data || err);
  alert(err.response?.data?.message || "Failed to update profile");
}
};

const handlePasswordChange = async () => {
  if (!passwords.oldPassword || !passwords.newPassword) {
    alert("Please fill both fields");
    return;
  }
  try {
    await api.put("/profile/password", passwords);
    alert("Password changed successfully!");
    setPasswords({ oldPassword: "", newPassword: "", confirmPassword: "" });
    setMode("view");
  } catch (err) {
    console.error("Password change error:", err.response?.data || err);
    alert(err.response?.data?.message || "Failed to change password");
  }
};

// ----- Loading / Errors -----
if (loading)
  return (
    <div className="ml-course-loading" style={{marginTop:"200px"}}>

```

```

<div className="ml-spinner" />
<p className="ml-loading-text">Loading your courses...</p>
</div>
);
if (!user) return <p>Please log in again.</p>

const excludeFields = [
  "_id",
  "user",
  "__v",
  "createdAt",
  "updatedAt",
  "coursesCreated",
  "enrolledCourses",
];
const courses = user.role === "student" ? profile?.enrolledCourses : null;

// ----- JSX -----
return (
  <div className="profile-simple-container">
    <h2 className="profile-title">
      {user.role.charAt(0).toUpperCase() + user.role.slice(1)} Profile
    </h2>

    <div className="profile-buttons">
      <button
        className={mode === "view" ? "active" : ""}
        onClick={() => setMode("view")}
      >
        View
      </button>
      <button
        className={mode === "edit" ? "active" : ""}
        onClick={() => setMode("edit")}
      >
        Edit
      </button>
      <button
        className={mode === "password" ? "active" : ""}
        onClick={() => setMode("password")}
      >
        Change Password
      </button>
    </div>
  </div>
);

```

```

</div>

<div className="profile-section">
  {/* ----- View Mode ----- */}
  {mode === "view" && (
    <div>
      <h4>Profile Details</h4>
      <img
        src={profilePic || DEFAULT_PROFILE}
        alt="Profile"
        className="profile-pic"
      />
      <p><strong>Name:</strong> {user.name}</p>
      <p><strong>Email:</strong> {user.email}</p>
      <p><strong>Role:</strong> {user.role}</p>

      {profile &&
        Object.entries(profile)
          .filter(([key]) => !excludeFields.includes(key))
          .map(([key, value]) => (
            <p key={key}>
              <strong>{key}:</strong>{" "}
              {Array.isArray(value) ? value.join(", ") : value || "None"}
            </p>
          )))
      }

      {user.role === "student" && courses && courses.length > 0 && (
        <div>
          <h4>Enrolled Courses</h4>
          <ul>
            {courses.map((course) =>
              <li key={course._id}>
                <a
                  onClick={() => navigate(`/courses/${course._id}`)}
                  style={{ cursor: "pointer", color: "#6D28D9" }}
                >
                  {course.title}
                </a>
              </li>
            ))}
          </ul>
        </div>
      )}
    </div>
  )
</div>

```

```

    )}

/* ----- Edit Mode ----- */
{mode === "edit" && (
  <div>
    <h4>Edit Profile</h4>
    <img
      src={
        profilePic instanceof File
        ? URL.createObjectURL(profilePic)
        : profilePic || DEFAULT_PROFILE
      }
      alt="Profile"
      className="profile-pic"
    />

    <div className="input-group-simple">
      <label>Profile Picture</label>
      <input type="file" accept="image/*" onChange={handleProfilePicChange} />
    </div>

    <div className="input-group-simple">
      <label>Name</label>
      <input
        type="text"
        name="name"
        value={userData.name || ""}
        onChange={handleUserChange}
      />
    </div>

{Object.entries(formData)
  .filter(([key]) => !excludeFields.includes(key))
  .map(([key, value]) => (
    <div key={key} className="input-group-simple">
      <label>{key}</label>
      <input
        type="text"
        name={key}
        value={Array.isArray(value) ? value.join(", ") : value || ""}
        onChange={(e) =>
          setFormData({
            ...formData,
            [key]: Array.isArray(value)
          })
        }
      />
    </div>
  ))
}

```

```

        ? e.target.value.split(",").map((v) => v.trim())
        : e.target.value,
    })
}
/>
</div>
))}

<button className="save-btn-simple" onClick={handleSave}>
  Save
</button>
</div>
)}
}

/* ----- Password Mode ----- */
{mode === "password" && (
<div>
  <h4>Change Password</h4>
  <div className="input-group-simple">
    <label>Old Password</label>
    <input
      type="password"
      value={passwords.oldPassword}
      onChange={(e) =>
        setPasswords({ ...passwords, oldPassword: e.target.value })
      }
    />
  </div>
  <div className="input-group-simple">
    <label>New Password</label>
    <input
      type="password"
      value={passwords.newPassword}
      onChange={(e) =>
        setPasswords({ ...passwords, newPassword: e.target.value })
      }
    />
  </div>
  <div className="input-group-simple">
    <label>Confirm Password</label>
    <input
      type="password"
      value={passwords.confirmPassword}
      onChange={(e) =>

```

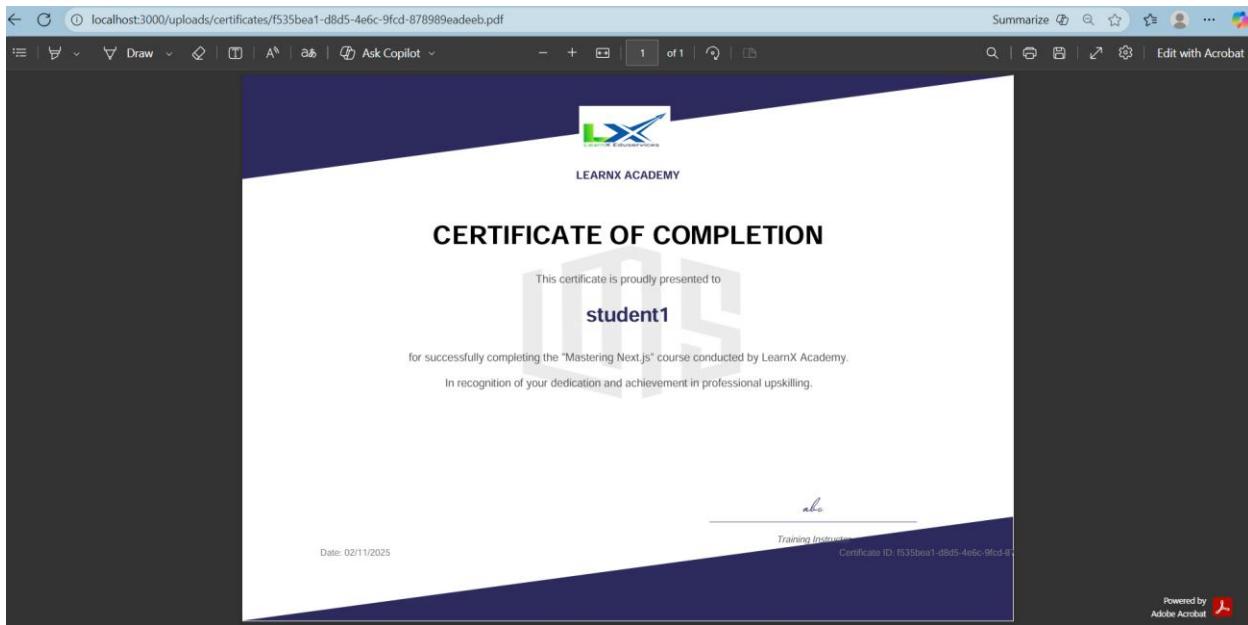
```
        setPasswords({ ...passwords, confirmPassword: e.target.value })
    }
/>
</div>
<button
    className="save-btn-simple"
    onClick={() => {
        if (
            !passwords.oldPassword ||
            !passwords.newPassword ||
            !passwords.confirmPassword
        ) {
            alert("Please fill all fields");
            return;
        }
        if (passwords.newPassword !== passwords.confirmPassword) {
            alert("New password and confirm password do not match");
            return;
        }
        handlePasswordChange();
    }}
    >
    Update Password
</button>
</div>
)}
</div>
</div>
);
}
```

Student Learning Page:

The screenshot shows the 'My Learnings' section of the LearnX platform. It displays three completed courses with their respective completion percentages and download certificate buttons.

- Mastering Next.js: Part 1**: Completed 100%. Includes a green progress bar and a purple 'Download Certificate' button.
- Advanced MERN Project**: Completed 80%. Includes a grey progress bar and a purple 'Download Certificate' button.
- Next.js Fundamentals**: Completed 0%. Includes a grey progress bar and a purple 'Continue Course' button.

Download Certificate option after course completion:



myLearnings.jsx

```
import { useEffect, useState } from "react";
```

```
import { useNavigate } from "react-router-dom";
import api from "../../api/api";
import "../../styles/myLearnings.css";

function MyLearnings() {
  const [courses, setCourses] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState("");
  const navigate = useNavigate();

  const loggedInUser = JSON.parse(localStorage.getItem("user"));
  const studentId = loggedInUser?._id;
  const BASE_URL = import.meta.env.VITE_BASE_URL || "http://localhost:3000";

  useEffect(() => {
    if (!studentId) {
      setError("User not logged in.");
      setLoading(false);
      return;
    }

    const fetchEnrollments = async () => {
      try {
        setLoading(true);
        const res = await api.get('/enrollments');
        console.log("Enrollments API response:", res.data);

        // ✅ Normalize various backend formats
        let enrollmentsArray = [];
        if (Array.isArray(res.data)) {
          enrollmentsArray = res.data;
        } else if (Array.isArray(res.data.enrollments)) {
          enrollmentsArray = res.data.enrollments;
        } else if (Array.isArray(res.data.data)) {
          enrollmentsArray = res.data.data;
        } else {
          console.warn("⚠️ Unexpected response format:", res.data);
        }
      } catch (err) {
        setError(`Error fetching enrollments: ${err.message}`);
        setLoading(false);
      }
    };

    fetchEnrollments();
  }, [studentId]);
}
```

```
    enrollmentsArray = [];
}

const formattedCourses = enrollmentsArray.map((enrollment) => ({
  ...enrollment.course,
  progress: enrollment.progress || 0,
  certificate: enrollment.certificate || null,
}));
```

```
  setCourses(formattedCourses);
} catch (err) {
  console.error("✖ Error fetching enrollments:", err);
  setError("Failed to load your courses.");
} finally {
  setLoading(false);
}
};

fetchEnrollments();
}, [studentId]);
```

```
const handleDownloadCertificate = async (certificateUrl) => {
  if (!certificateUrl) return alert("No certificate available yet.");
```

```
  try {
    const link = document.createElement("a");
    link.href = `${BASE_URL}${certificateUrl}`;
    link.setAttribute("download", "certificate.pdf");
    document.body.appendChild(link);
    link.click();
    link.remove();
  } catch (err) {
    console.error("✖ Error downloading certificate:", err);
    alert("Failed to download certificate.");
  }
};
```

```

if (loading)
  return (
    <div className="ml-course-loading" style={{ marginTop: "200px" }}>
      <div className="ml-spinner" />
      <p className="ml-loading-text">Loading your courses...</p>
    </div>
  );
}

if (error) return <p className="ml-error-text">{error}</p>;
if (courses.length === 0)
  return <p className="ml-info-text">You have not enrolled in any courses
yet.</p>

return (
  <div className="ml-container" style={{ marginTop: "60px" }}>
    <h2>My Learnings</h2>
    <div className="ml-courses-list">
      {courses.map((course, idx) => {
        const progressPercent =
          course.progress ??
          (course.completedLessons && course.lessons
            ? Math.round(
                (course.completedLessons.length / course.lessons.length) * 100
              )
            : 0);
      });
    
```

return (

```

      <div key={course._id || idx} className="ml-course-card">
        <div
          className="ml-course-thumbnail"
          style={{
            backgroundImage: `url(${{
              course.thumbnail
              ? course.thumbnail.startsWith("http")
              ? course.thumbnail
              : `${BASE_URL}${course.thumbnail}`
              : "/placeholder.png"
            }}`
          }}
        </div>
      </div>
    
```

```
        `),
        backgroundPosition: "center",
        backgroundSize: "100% 100%",
    `}
/>
</div>

<div className="ml-course-details">
    <h3>{course.title || "Untitled Course"}</h3>
    <p>
        {course.description?.substring(0, 120) ||
        "No description available."}
        ...
    </p>

    <div className="ml-progress-wrapper">
        <div className="ml-progress-bg">
            <div
                className="ml-progress-fill"
                style={{
                    width: `${progressPercent}%`,
                    background: `
                        linear-gradient(
                            90deg,
                            ${{
                                progressPercent < 40
                                    ? "#ff4d4d"
                                    : progressPercent < 75
                                    ? "#ffcc00"
                                    : "#4caf50"
                            },
                            ${{
                                progressPercent < 40
                                    ? "#ff8080"
                                    : progressPercent < 75
                                    ? "#ffe680"
                                    : "#81c784"
                            }
                        )
                    `
                }}
            >
            </div>
        </div>
    </div>
</div>
```

```

        )
      ,
    })
  />
</div>
<span className="ml-progress-text">
  {progressPercent}% Completed
</span>
</div>

<div className="ml-course-actions" style={{ gap: "15px" }}>
  <button onClick={() => navigate(`/courses/${course._id}`)} style={{
width:"200px", marginRight:"10px"}}>
    Continue Course
  </button>

  {progressPercent === 100 && course.certificate && (
    <button
      className="ml-download-btn"
      onClick={() =>
        handleDownloadCertificate(course.certificate)
      }
      style={{padding:"5px 30px"}}
    >
      Download Certificate
    </button>
  )}
</div>
</div>
</div>
);
})}
</div>
</div>
);
}

```

```
export default MyLearnings;
```

Common Sidebars for Instructor and Admin

dashboardLayout.jsx

```
import React, { useState } from "react";
import { Link, useLocation, useNavigate } from "react-router-dom";
import {
  FaBell,
  FaUser,
  FaBook,
  FaChartLine,
  FaClipboardList,
  FaChevronDown,
  FaChevronUp,
  FaSignOutAlt,
  FaUsers,
  FaRupeeSign
} from "react-icons/fa";

/**
 * DashboardLayout component for Admin and Instructor dashboards
 * @param {Array} sidebarLinks - Links to render in sidebar
 * @param {ReactNode} children - Main content
 */
export default function DashboardLayout({ sidebarLinks, children }) {
  const [openSections, setOpenSections] = useState({});
  const location = useLocation();
  const navigate = useNavigate();

  // Get role from localStorage, default to instructor
  const userRole = localStorage.getItem("role") || "instructor";

  // Toggle sidebar section
  const toggleSection = (label) => {
    setOpenSections((prev) => ({
      ...prev,
      [label]: !prev[label],
    }));
  };
}
```

```
// Logout function
const handleLogout = () => {
  localStorage.removeItem("token");
  localStorage.removeItem("role");
  localStorage.removeItem("user");
  navigate("/login");
  window.location.reload();
};

// Styling
const styles = {
  sidebar: {
    backgroundColor: "#6f42c1",
    minHeight: "100vh",
    width: "269px",
    display: "flex",
    flexDirection: "column",
    padding: "2rem 1rem",
  },
  sidebarLink: {
    color: "white",
    display: "flex",
    alignItems: "center",
    justifyContent: "space-between",
    gap: "10px",
    padding: "10px 15px",
    borderRadius: "8px",
    marginBottom: "8px",
    textDecoration: "none",
    fontWeight: 500,
    cursor: "pointer",
    transition: "background 0.3s",
  },
  nestedLink: {
    paddingLeft: "35px",
    fontSize: "0.9rem",
    display: "flex",
    alignItems: "center",
    gap: "10px",
    marginBottom: "6px",
    textDecoration: "none",
    color: "white",
    borderRadius: "6px",
  }
};
```

```

padding: "8px 12px",
transition: "background 0.3s",
},
main: {
flexGrow: 1,
padding: "30px",
backgroundColor: "#f9f7fc",
minHeight: "100vh",
},
};

// Check if a path is active
const isActive = (path) => location.pathname === path;

return (
<div style={{ display: "flex" }}>
/* Sidebar */
<div style={styles.sidebar}>
<h3 className="text-center fw-bold mb-5 text-white">
<Link
to={userRole === "admin" ? "/admin-dashboard" : "/instructor-dashboard"}
style={{ textDecoration: "none", color: "white" }}
>
Dashboard
</Link>
</h3>

/* Sidebar Links */
{Array.isArray(sidebarLinks) &&
sidebarLinks.map((link, idx) => (
<div key={idx}>
<div
style={{
...styles.sidebarLink,
backgroundColor: openSections[link.label] ? "#5931a0" : "transparent",
}}
onClick={() => toggleSection(link.label)}
>
<span style={{ display: "flex", alignItems: "center", gap: "10px" }}>
{link.icon} {link.label}
</span>
{Array.isArray(link.children) &&
(openSections[link.label] ? <FaChevronUp /> : <FaChevronDown />)}
</div>

```

```

/* Nested Links */
{Array.isArray(link.children) &&
openSections[link.label] &&
link.children.map((child, cidx) =>
  child?.path && child?.label ? (
    <Link
      key={cidx}
      to={child.path}
      style={{
        ...styles.nestedLink,
        backgroundColor: isActive(child.path) ? "#5931a0" : "transparent",
      }}
    >
      - {child.label}
    </Link>
  ) : null
)
</div>
))}

<hr style={{ borderColor: "rgba(255,255,255,0.3)" }} />

/* Logout */
<div
  style={{
    ...styles.sidebarLink,
    backgroundColor: "#5931a0",
    justifyContent: "center",
  }}
  onClick={handleLogout}
>
  <FaSignOutAlt style={{ marginRight: "10px" }} />
  Logout
</div>
</div>

/* Main Content */
<div style={styles.main}>{children}</div>
</div>
);

}

/**

```

```

* Instructor sidebar links (can be imported in instructor dashboard)
*/
export const instructorSidebarLinks = [
{
  label: "Courses",
  icon: <FaBook />,
  children: [
    { label: "My Courses", path: "/instructor-dashboard/instructor_courses" },
    { label: "Add New Course", path: "/instructor-dashboard/add_courses" },
    { label: "Pending Approvals", path: "/instructor-dashboard/pending_approvals" },
  ],
},
{
  label: "Categories",
  icon: <FaBook />,
  children: [
    { label: "Request Category", path: "/instructor-dashboard/request-category" },
  ],
},
{
  label: "Lessons",
  icon: <FaBook />,
  children: [
    { label: "Manage Lessons", path: "/instructor-dashboard/manage_lessons" },
  ],
},
{
  label: "Exams",
  icon: <FaClipboardList />,
  children: [
    { label: "Manage Exams", path: "/instructor-dashboard/manage_exams" },
    /* { label: "Exam Results", path: "/instructor-dashboard/exam_results" }, */
  ],
},
{
  label: "Students",
  icon: <FaUser />,
  children: [
    { label: "Enrolled Students", path: "/instructor-dashboard/enrolled_students" },
  ],
},
{
  label: "Earnings",
  icon: <FaRupeeSign />,

```

```

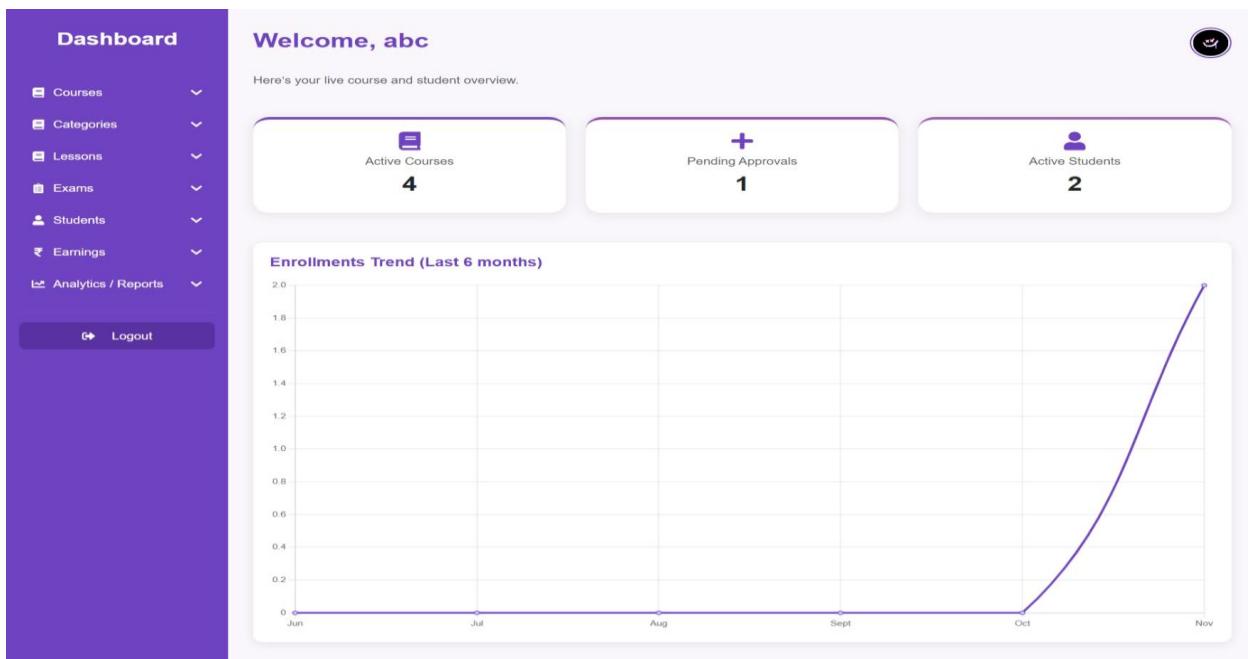
children: [
  { label: "My Earnings", path: "/instructor-dashboard/earnings" },
  { label: "Payout History", path: "/instructor-dashboard/payout-history" },
],
},
{
  label: "Analytics / Reports",
  icon: <FaChartLine />,
  children: [
    { label: "Course Analytics", path: "/instructor-dashboard/course_analytics" },
    { label: "Student Progress", path: "/instructor-dashboard/student_progress" },
  ],
},
];
export const adminSidebarLinks = [
{
  label: "Users",
  icon: <FaUsers />,
  children: [
    { label: "All Users", path: "/admin-dashboard/users" },
    /* { label: "Instructors", path: "/admin-dashboard/instructors" },
    { label: "Students", path: "/admin-dashboard/students" }, */
  ],
},
{
  label: "Courses",
  icon: <FaBook />,
  children: [
    { label: "All Courses", path: "/admin-dashboard/courses" },
    { label: "Pending Approval", path: "/admin-dashboard/pending-courses" },
    { label: "Rejected Courses", path: "/admin-dashboard/rejected-courses" },
  ],
},
{
  label: "Categories",
  icon: <FaBook />,
  children: [
    { label: "Manage Categories", path: "/admin-dashboard/categories" },
    { label: "Suggestions", path: "/admin-dashboard/category-suggestions" },
  ],
},
{
  label: "Payments & Revenue",
  icon: <FaRupeeSign />,
}

```

```
children: [
  { label: "Earnings Overview", path: "/admin-dashboard/revenue" },
  { label: "Instructor Payouts", path: "/admin-dashboard/payouts" },
  { label: "Transactions", path: "/admin-dashboard/transactions" },
],
},
{
  label: "Reports & Analytics",
  icon: <FaClipboardList />,
  children: [
    { label: "Enrollment Stats", path: "/admin-dashboard/reports/enrollments" },
    { label: "Course Performance", path: "/admin-dashboard/reports/courses" },
  ],
},
{
  label: "Support",
  icon: <FaBell />,
  children: [
    { label: "Contact Messages", path: "/admin-dashboard/contact-messages" },
  ],
},
];
};
```

INSTRUCTOR SIDE

Instructor dashboard:



instructorDashboard.jsx

```
import React, { useEffect, useState } from "react";
import "bootstrap/dist/css/bootstrap.min.css";
import "bootstrap/dist/js/bootstrap.bundle.min.js";
import { Outlet, useLocation, useNavigate } from "react-router-dom";
import { Line } from "react-chartjs-2";
import {
  Chart as ChartJS,
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
  Title,
  Tooltip,
  Legend,
} from "chart.js";

import { FaBell, FaUser, FaBook, FaPlus } from "react-icons/fa";
```

```

import DashboardLayout, { instructorSidebarLinks } from "../dashboardLayout";
import api from "../../api/api";

ChartJS.register(CategoryScale, LinearScale, PointElement, LineElement, Title, Tooltip, Legend);

export default function InstructorDashboard() {
  const location = useLocation();
  const navigate = useNavigate();

  const [activeCourses, setActiveCourses] = useState(0);
  const [pendingApprovals, setPendingApprovals] = useState(0);
  const [newStudents, setNewStudents] = useState(0);
  const [profile, setProfile] = useState({ name: "", image: "/uploads/default.png" });
  const [chartData, setChartData] = useState({ labels: [], datasets: [] });
  const [loading, setLoading] = useState(true);

  const BASE_URL = import.meta.env.VITE_BASE_URL || ""; // optional for full image URL

  useEffect(() => {
    const fetchDashboardData = async () => {
      try {
        const token = localStorage.getItem("token");
        const res = await api.get("/instructor/dashboard", {
          headers: { Authorization: `Bearer ${token}` },
        });

        // Set stats
        setActiveCourses(res.data.myCourses.length || 0);
        setPendingApprovals(res.data.pendingApprovals.length || 0);
        setNewStudents(res.data.newStudents || 0);

        // Set profile
        const profileImage = res.data.profile?.image
          ? res.data.profile.image.startsWith("http")
          ? res.data.profile.image
          : `${BASE_URL}${res.data.profile.image}`
          : `${BASE_URL}/uploads/default.png`;

        setProfile({
          name: res.data.profile?.name || "Instructor",
          image: profileImage,
        });

        // Set chart data
      }
    };
  });
}

```

```

setChartData({
  labels: res.data.chartData.labels,
  datasets: [
    {
      label: "Enrollments",
      data: res.data.chartData.data,
      borderColor: "#6f42c1",
      backgroundColor: "#d1c4ff",
      tension: 0.4,
    },
  ],
});
} catch (error) {
  console.error("Dashboard fetch failed:", error);
} finally {
  setLoading(false);
}
};

fetchDashboardData();
}, []));

const chartOptions = {
  responsive: true,
  plugins: { legend: { display: false } },
  scales: { y: { beginAtZero: true } },
};

const styles = {
  textPurple: { color: "#6f42c1" },
  card: {
    borderRadius: "20px",
    textAlign: "center",
    padding: "20px",
    boxShadow: "0 4px 20px rgba(0,0,0,0.08)",
    backgroundColor: "white",
    transition: "transform 0.3s, box-shadow 0.3s",
  },
  cardBorderTop: (color) => ({ borderTop: `4px solid ${color}` }),
  chartCard: {
    borderRadius: "12px",
    padding: "20px",
    boxShadow: "0 4px 15px rgba(0,0,0,0.08)",
    backgroundColor: "white",
  }
};

```

```

        marginTop: "20px",
    },
    profileBorder: {
        border: "2px solid #6f42c1",
        padding: "2px",
        cursor: "pointer",
        width: "50px",
        height: "50px",
        objectFit: "cover",
    },
};

if (loading) {
    return (
        <DashboardLayout sidebarLinks={instructorSidebarLinks}>
            <div style={{ display: "flex", flexDirection: "column", justifyContent: "center", alignItems: "center", height: "70vh", color: "#6f42c1", }}>
                <div style={{ border: "4px solid #f3f3f3", borderTop: "4px solid #6f42c1", borderRadius: "50%", width: "50px", height: "50px", animation: "spin 1s linear infinite", marginBottom: "20px", }} />
                    <p>Loading your data...</p>
                    <style>{@keyframes spin { 0% { transform: rotate(0deg); } 100% { transform: rotate(360deg); } }}</style>
                </div>
            </DashboardLayout>
        );
    }
}

return (
    <DashboardLayout sidebarLinks={instructorSidebarLinks}>
        {location.pathname === "/instructor-dashboard" ? (
            <>
                {/* Top Header */}
                <div className="d-flex justify-content-between align-items-center mb-4">
                    <h2 style={styles.textPurple} className="fw-bold">
                        Welcome, {profile.name}
                    </h2>

                    <div className="d-flex align-items-center gap-3">

                        <img
                            src={profile.image}
                            alt="profile"

```

```

      className="rounded-circle"
      style={styles.profileBorder}
      onClick={() => navigate("/instructor-dashboard/profile")}
    />
  </div>
</div>

<p className="text-muted mb-5">
  Here's your live course and student overview.
</p>

/* Stats Cards */
<div className="row mb-5 g-4">
  {[{"icon": "FaBook", "title": "Active Courses", "value": activeCourses, "color": "#6f42c1"}, {"icon": "FaPlus", "title": "Pending Approvals", "value": pendingApprovals, "color": "#8e44ad"}, {"icon": "FaUser", "title": "Active Students", "value": newStudents, "color": "#9b59b6"}].map((card, idx) => (
    <div key={idx} className="col-md-4">
      <div style={{ ...styles.card, ...styles.cardBorderTop(card.color) }}>
        <card.icon size={30} className="mb-2" style={styles.textPurple} />
        <h6 className="text-muted">{card.title}</h6>
        <h2 className="fw-bold">{card.value}</h2>
      </div>
    </div>
  )))
</div>

/* Chart Section */
<div style={styles.chartCard}>
  <h5 style={styles.textPurple} className="fw-bold mb-3">
    Enrollments Trend (Last 6 months)
  </h5>
  <Line data={chartData} options={chartOptions} />
</div>
</>
):(
  <Outlet />
)}
</DashboardLayout>
);
}

```

Manage Courses Page:

The screenshot shows a user interface for managing courses. On the left, a purple sidebar titled "Dashboard" contains navigation links for Courses, Categories, Lessons, Exams, Students, Earnings, and Analytics / Reports, along with a "Logout" button. The main area is titled "My Courses" and displays a grid of course cards. Each card includes a thumbnail image, the course title, category, level, price, status, and edit/delete buttons.

Course Title	Category	Level	Price	Status
Mastering Next JS: Part 1	Next JS	Intermediate	Free	approved
Advanced MERN Project	MERN Stack Development	Intermediate	₹299	approved
Next JS Fundamentals	Next JS	Beginner	₹199	approved
Backend Development with Node.js	Backend Development	Intermediate	₹290	draft
Frontend Frameworks with React	Frontend Development	Intermediate	₹149	approved
abc	Next JS	Intermediate	₹99	pendingApproval
xyz	Python	Intermediate	₹99	rejected

instructorCourses.jsx

```
import React, { useEffect, useState } from "react";
import api from "../../api/api";

export default function InstructorCourses() {
  const [courses, setCourses] = useState([]);
  const [categories, setCategories] = useState([]);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState("");

  const [modalOpen, setModalOpen] = useState(false);
  const [editingCourse, setEditingCourse] = useState(null);

  const [submitting, setSubmitting] = useState(false);
  const [uploading, setUploading] = useState(false);

  // Delete confirmation state
  const [confirmOpen, setConfirmOpen] = useState(false);
  const [selectedCourse, setSelectedCourse] = useState(null);
```

```

const [deleting, setDeleting] = useState(false);

const [form, setForm] = useState({
  title: "",
  description: "",
  category: "",
  level: "",
  price: "",
  thumbnail: ""
});

const BASE_URL = import.meta.env.VITE_BASE_URL;

// ----- FETCH COURSES + CATEGORIES -----
useEffect(() => {
  const token = localStorage.getItem("token");

  if (!token) {
    setError("Unauthorized");
    return;
  }

  const fetchCourses = async () => {
    try {
      setLoading(true);

      const res = await api.get("/instructor/courses", {
        headers: { Authorization: `Bearer ${token}` },
      });

      setCourses(res.data.courses || []);
    } catch (err) {
      console.error(err);
      setError("Failed to fetch courses.");
    } finally {
      setLoading(false);
    }
  };

  const fetchCategories = async () => {
    try {
      const res = await api.get("/courses/categories");
      setCategories(res.data.categories || []);
    } catch (err) {

```

```

        console.error(err);
    }
};

fetchCourses();
fetchCategories();
}, []);

// ----- EDIT MODAL -----
const openEditModal = (course) => {
    setEditingCourse(course);
    setForm({
        title: course.title || "",
        description: course.description || "",
        category:
            typeof course.category === "object"
                ? course.category._id
                : course.category || "",
        level: course.level || "",
        price: course.price || "",
        thumbnail: course.thumbnail || ""
    });
    setModalOpen(true);
};

const handleChange = (e) => {
    const { name, value } = e.target;
    setForm((prev) => ({ ...prev, [name]: value }));
};

// ----- THUMBNAIL UPLOAD -----
const handleThumbnailUpload = async (e) => {
    const file = e.target.files[0];
    if (!file) return;

    const token = localStorage.getItem("token");

    try {
        setUploading(true);
        setError("");
        const formData = new FormData();
        formData.append("thumbnail", file);
    }
}

```

```

const res = await api.post(
  "/instructor/course/upload-thumbnail",
  formData,
  {
    headers: {
      "Content-Type": "multipart/form-data",
      Authorization: `Bearer ${token}`,
    },
  }
);

setForm((prev) => ({ ...prev, thumbnail: res.data.fileUrl }));
} catch (err) {
  console.error(err);
  setError("Thumbnail upload failed");
} finally {
  setUploading(false);
}
};

// ----- SAVE COURSE -----
const handleSubmit = async (e) => {
  e.preventDefault();

  try {
    setSubmitting(true);
    const token = localStorage.getItem("token");

    const res = await api.put(
      `/instructor/course/${editingCourse._id}`,
      form,
      {
        headers: { Authorization: `Bearer ${token}` },
      }
    );

    setCourses((prev) =>
      prev.map((c) =>
        c._id === editingCourse._id ? res.data.course : c
      )
    );
  } finally {
    setModalOpen(false);
  }
};

```

```

} catch (err) {
  console.error(err);
  setError("Failed to update course");
} finally {
  setSubmitting(false);
}
};

// ----- STATUS UPDATE -----
const handleStatusUpdate = async (course, newStatus) => {
  const token = localStorage.getItem("token");

  try {
    const res = await api.put(
      `/instructor/course/${course._id}/status`,
      { status: newStatus },
      { headers: { Authorization: `Bearer ${token}` } }
    );

    setCourses((prev) =>
      prev.map((c) => (c._id === course._id ? res.data.course : c))
    );
  } catch (err) {
    console.error(err);
    setError("Failed to update status");
  }
};

// ----- DELETE CONFIRMATION -----
const confirmDelete = (course) => {
  setSelectedCourse(course);
  setConfirmOpen(true);
};

const handleDelete = async () => {
  if (!selectedCourse) return;

  try {
    setDeleting(true);
    const token = localStorage.getItem("token");

    await api.delete(`/instructor/course/${selectedCourse._id}`, {
      headers: { Authorization: `Bearer ${token}` },
    });
  }
};

```

```

setCourses((prev) =>
  prev.filter((c) => c._id !== selectedCourse._id)
);

setConfirmOpen(false);
setSelectedCourse(null);
} catch (err) {
  console.error(err);
  setError("Failed to delete course");
} finally {
  setDeleting(false);
}
};

// ----- UI -----
if (loading) {
  return (
    <div
      style={{
        display: "flex",
        flexDirection: "column",
        justifyContent: "center",
        alignItems: "center",
        height: "70vh",
        color: "#6f42c1",
      }}
    >
    <div
      style={{
        border: "4px solid #f3f3f3",
        borderTop: "4px solid #6f42c1",
        borderRadius: "50%",
        width: "50px",
        height: "50px",
        animation: "spin 1s linear infinite",
        marginBottom: "20px",
      }}
    />
    <p>Loading courses...</p>
    <style>{
      @keyframes spin {
        0% { transform: rotate(0deg); }
        100% { transform: rotate(360deg); }
      }
    </style>
  
```

```

        }
      `}</style>
    </div>
  );
}

return (
  <div style={{ padding: "20px", minHeight: "100vh" }}>
    <h2 style={{ marginBottom: "20px", color: "#6f42c1" }}>
      My Courses
    </h2>

    {error && <p style={{ color: "red" }}>{error}</p>

    {courses.length === 0 ? (
      <p>No courses created yet.</p>
    ) : (
      <div
        style={{
          display: "grid",
          gridTemplateColumns:
            "repeat(auto-fill, minmax(260px,1fr))",
          gap: "15px",
        }}
      >
        {courses.map((course) => (
          <div
            key={course._id}
            style={{
              background: "#fff",
              padding: "12px",
              borderRadius: "10px",
              boxShadow: "0 2px 6px rgba(0,0,0,0.1)",
              border: "1px solid #e4d7f5",
            }}
          >
            <img
              src={
                course.thumbnail
                ? `${BASE_URL}${course.thumbnail}`
                : "https://via.placeholder.com/300x150"
              }
              alt={course.title}
              style={{

```

```
        width: "100%",
        height: "150px",
        objectFit: "cover",
        borderRadius: "8px",
        marginBottom: "8px",
    }}
/>

<h3 style={{ fontSize: "1.1rem" }}>{course.title}</h3>

<p style={{ margin: "3px 0" }}>
    <strong>Category:</strong>{" "}
    {course.category?.name || "N/A"}
</p>

<p style={{ margin: "3px 0" }}>
    <strong>Level:</strong> {course.level}
</p>

<p style={{ margin: "3px 0" }}>
    <strong>Price:</strong>{" "}
    {course.price > 0 ? `₹${course.price}` : "Free"}
</p>

<p style={{ margin: "3px 0" }}>
    <strong>Status:</strong> {course.status}
</p>

<button
    onClick={() => openEditModal(course)}
    style={{
        marginTop: "8px",
        width: "35%",
        padding: "6px 0",
        borderRadius: "5px",
        border: "none",
        background: "#7c3aed",
        color: "#fff",
        cursor: "pointer",
        marginRight: "5px",
    }}
>
    Edit
</button>
```

```
{course.status === "draft" && (
  <button
    onClick={() =>
      handleStatusUpdate(course, "pendingApproval")
    }
    style={{
      marginTop: "8px",
      width: "60%",
      padding: "6px 0",
      borderRadius: "5px",
      border: "none",
      background: "#28a745",
      color: "#fff",
      cursor: "pointer",
    }}
  >
  Submit for Approval
</button>
)}

{course.status === "pendingApproval" && (
  <button
    onClick={() =>
      handleStatusUpdate(course, "draft")
    }
    style={{
      marginTop: "8px",
      width: "60%",
      padding: "6px 0",
      borderRadius: "5px",
      border: "none",
      background: "#ffc107",
      color: "#000",
      cursor: "pointer",
    }}
  >
  Revert to Draft
</button>
)

{!(course.status === "draft" ||
  course.status === "pendingApproval") && (
  <button
```

```

        onClick={() => confirmDelete(course)}
        disabled={deleting}
        style={{
          marginTop: "10px",
          width: "100%",
          padding: "6px 0",
          borderRadius: "5px",
          border: "none",
          background: "#dc3545",
          color: "#fff",
          cursor: deleting ? "not-allowed" : "pointer",
        }}
      >
    {deleting &&
      selectedCourse?._id === course._id
      ? "Deleting..."
      : "Delete"
    }
  </button>
)
</div>
))}
</div>
}

/* ----- EDIT MODAL ----- */
{modalOpen && (
  <div
    style={{
      position: "fixed",
      inset: 0,
      background: "rgba(0,0,0,0.5)",
      display: "flex",
      justifyContent: "center",
      alignItems: "center",
      zIndex: 999,
      padding: "20px",
    }}
  >
    <form
      onSubmit={handleSubmit}
      style={{
        background: "#fff",
        padding: "20px",
        borderRadius: "10px",
      }}
    >

```

```
        width: "100%",
        maxWidth: "450px",
    }}
>
<h3 style={{ marginBottom: "10px" }}>Edit Course</h3>

<input
    name="title"
    placeholder="Title"
    value={form.title}
    onChange={handleChange}
    required
    style={{
        width: "100%",
        marginBottom: "10px",
        padding: "6px",
    }}
/>

<textarea
    name="description"
    placeholder="Description"
    value={form.description}
    onChange={handleChange}
    style={{
        width: "100%",
        marginBottom: "10px",
        padding: "6px",
    }}
/>

<select
    name="category"
    value={form.category}
    onChange={handleChange}
    required
    style={{
        width: "100%",
        marginBottom: "10px",
        padding: "6px",
    }}
>
<option value="">Select Category</option>
{categories.map((cat) => (
```

```
<option value={cat._id} key={cat._id}>
  {cat.name}
</option>
))}
</select>
```

```
<input
  name="level"
  placeholder="Level"
  value={form.level}
  onChange={handleChange}
  style={{
    width: "100%",
    marginBottom: "10px",
    padding: "6px",
  }}
/>
```

```
<input
  name="price"
  type="number"
  placeholder="Price"
  value={form.price}
  onChange={handleChange}
  style={{
    width: "100%",
    marginBottom: "10px",
    padding: "6px",
  }}
/>
```

```
<input
  type="file"
  onChange={handleThumbnailUpload}
  style={{ marginBottom: "10px" }}
/>
```

```
{uploading && <p>Uploading...</p>}
```

```
{form.thumbnail && !uploading && (
  <img
    src={'${BASE_URL}${form.thumbnail}'}
    alt="thumbnail"
    style={{
```

```
        width: "120px",
        height: "70px",
        marginBottom: "10px",
        borderRadius: "5px",
        objectFit: "cover",
    }}
/>
)}
```

```
{editingCourse?.status === "draft" && (
    <button
        type="button"
        onClick={() =>
            handleStatusUpdate(editingCourse, "pendingApproval")
        }
        style={{
            marginBottom: "10px",
            width: "100%",
            padding: "6px 0",
            borderRadius: "5px",
            border: "none",
            background: "#28a745",
            color: "#fff",
            cursor: "pointer",
        }}
    >
        Submit for Approval
    </button>
)}
```

```
{editingCourse?.status === "pendingApproval" && (
    <button
        type="button"
        onClick={() =>
            handleStatusUpdate(editingCourse, "draft")
        }
        style={{
            marginBottom: "10px",
            width: "100%",
            padding: "6px 0",
            borderRadius: "5px",
            border: "none",
            background: "#ffc107",
            color: "#000",
        }}
    >
```

```

        cursor: "pointer",
    }}
    >
    Revert to Draft
    </button>
)}

<div style={{ textAlign: "right" }}>
    <button
        type="button"
        onClick={() => setModalOpen(false)}
        style={{
            padding: "6px 12px",
            marginRight: "10px",
            background: "#ccc",
            border: "none",
            borderRadius: "5px",
        }}
    >
        Cancel
    </button>

    <button
        type="submit"
        disabled={submitting || uploading}
        style={{
            padding: "6px 12px",
            background: "#7c3aed",
            border: "none",
            borderRadius: "5px",
            color: "#fff",
            cursor: "pointer",
        }}
    >
        {submitting ? "Saving..." : "Save"}
    </button>
</div>
</form>
</div>
)}

/* ----- DELETE MODAL ----- */
{confirmOpen && selectedCourse && (
    <div

```

```
style={{
  position: "fixed",
  inset: 0,
  background: "rgba(0,0,0,0.5)",
  display: "flex",
  justifyContent: "center",
  alignItems: "center",
  zIndex: 999,
}}
>
<div
  style={{
    background: "#fff",
    width: "90%",
    maxWidth: "400px",
    padding: "20px",
    borderRadius: "10px",
    textAlign: "center",
  }}
>
  <h3>Delete Course?</h3>
  <p>
    Are you sure you want to permanently delete this course?
  </p>

  <div
    style={{
      display: "flex",
      justifyContent: "center",
      gap: "10px",
      marginTop: "20px",
    }}
  >
    <button
      onClick={() => setConfirmOpen(false)}
      style={{
        padding: "6px 12px",
        background: "#ccc",
        border: "none",
        borderRadius: "5px",
      }}
    >
      Cancel
    </button>
```

```

<button
  onClick={handleDelete}
  disabled={deleting}
  style={{
    padding: "6px 12px",
    background: "#dc3545",
    border: "none",
    borderRadius: "5px",
    color: "#fff",
    cursor: deleting ? "not-allowed" : "pointer",
  }}
>
  {deleting ? "Deleting..." : "Delete"}
</button>
</div>
</div>
</div>
)}
</div>
);
}

```

Add Courses Page

The screenshot shows the 'Add New Course' form overlaid on the 'Dashboard' sidebar.

Dashboard Sidebar:

- Courses: - My Courses, - Add New Course (highlighted in purple), - Pending Approvals
- Categories
- Lessons
- Exams
- Students
- Earnings
- Analytics / Reports
- Logout

Add New Course Form:

Title: Enter course title

Description: Enter course description

Category: - Select Category --

Level: Beginner

Price (₹): 0

Create Course button

addCourses.jsx

```
import { useState, useEffect } from "react";
import api from "../../api/api";
import { useNavigate } from "react-router-dom";
import "../styles/AddCourse.css";

function AddCourse() {
  const [title, setTitle] = useState("");
  const [description, setDescription] = useState("");
  const [category, setCategory] = useState("");
  const [categories, setCategories] = useState([]);
  const [level, setLevel] = useState("Beginner");
  const [price, setPrice] = useState(0);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState("");
  const navigate = useNavigate();

  const token = localStorage.getItem("token");
  const instructorId = localStorage.getItem("userId");

  useEffect(() => {
    const fetchCategories = async () => {
      try {
        const res = await api.get("/courses/categories", {
          headers: { Authorization: `Bearer ${token}` },
        });
        const cats = Array.isArray(res.data) ? res.data : res.data.categories || [];
        setCategories(cats);
      } catch (err) {
        console.error("Fetch categories error:", err);
        setError("Failed to fetch categories.");
        setCategories([]);
      }
    };
    fetchCategories();
  }, [token]);

  const handleSubmit = async (e) => {
    e.preventDefault();
    setError("");

    if (!token) return setError("You must be logged in.");
  }
}
```

```

if (!category) return setError("Please select a category.");

try {
  setLoading(true);
  const res = await api.post(
    "/instructor/create-course",
    { title, description, category, level, instructor: instructorId, price },
    { headers: { Authorization: `Bearer ${token}` } }
  );

  if (res.status === 201 && res.data.course?._id) {
    alert("Course created successfully!");
    navigate("/instructor-dashboard/instructor_courses");
  } else {
    console.error("Unexpected response:", res.data);
    setError("Unexpected server response. Check console for details.");
  }
} catch (err) {
  console.error("Create course error:", err);
  setError(
    err.response?.data?.error ||
    err.response?.data?.message ||
    err.message ||
    "Failed to create course."
  );
} finally {
  setLoading(false);
}
};

return (
  <div className="add-course-page">
    <div className="add-course-card">
      <h2 className="add-course-title">Add New Course</h2>
      {error && <p className="error-message">{error}</p>}

      <form onSubmit={handleSubmit} className="add-course-form">
        <label>Title</label>
        <input
          value={title}
          onChange={(e) => setTitle(e.target.value)}
          required
          className="input-field"
          placeholder="Enter course title"
        >
      </form>
    </div>
  </div>
);

```

```
/>

<label>Description</label>
<textarea
  value={description}
  onChange={(e) => setDescription(e.target.value)}
  className="input-field"
  placeholder="Enter course description"
/>

<label>Category</label>
<select
  value={category}
  onChange={(e) => setCategory(e.target.value)}
  required
  className="input-field"
>
  <option value="">-- Select Category --</option>
{Array.isArray(categories) &&
  categories.map((cat) => (
    <option key={cat._id} value={cat._id}>
      {cat.name}
    </option>
  )))
</select>

<label>Level</label>
<select
  value={level}
  onChange={(e) => setLevel(e.target.value)}
  className="input-field"
>
  <option>Beginner</option>
  <option>Intermediate</option>
  <option>Advanced</option>
</select>

<label>Price (₹)</label>
<input
  type="number"
  value={price}
  onChange={(e) => setPrice(Number(e.target.value))}
  className="input-field"
  placeholder="0"
```

```

        />

        <button type="submit" className="submit-btn" disabled={loading}>
          {loading ? "Saving..." : "Create Course"}
        </button>
      </form>
    </div>
  </div>
);
}

export default AddCourse;

```

Pending Approvals Page:

pendingApprovals.jsx

```

import { useEffect, useState } from "react";
import api from "../../api/api";

function PendingApprovals() {
  const [courses, setCourses] = useState([]);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState("");
  const [modalOpen, setModalOpen] = useState(false);
  const [editingCourse, setEditingCourse] = useState(null);
  const [form, setForm] = useState({
    title: "",
    category: "Next JS",
    level: "Intermediate",
    price: "₹99",
    lessons: 0,
    exams: 0,
    status: "pendingApproval"
  });
}

```

```

title: "",
description: "",
category: "",
level: "",
price: "",
thumbnail: "",
});
const [categories, setCategories] = useState([]);
const [submitting, setSubmitting] = useState(false);
const [uploading, setUploading] = useState(false);

const BASE_URL = import.meta.env.VITE_BASE_URL;

// Fetch pending courses
const fetchPending = async () => {
  try {
    setLoading(true);
    const token = localStorage.getItem("token");
    if (!token) throw new Error("No token found");

    const res = await api.get("/instructor/courses?status=pendingApproval", {
      headers: { Authorization: `Bearer ${token}` },
    });
    setCourses(res.data.courses || []);
  } catch (err) {
    console.error(err);
    setError("Failed to fetch pending courses");
  } finally {
    setLoading(false);
  }
};

useEffect(() => {
  fetchPending();

  const fetchCategories = async () => {
    try {
      const res = await api.get("/courses/categories");
      setCategories(Array.isArray(res.data) ? res.data : res.data.categories || []);
    } catch (err) {
      console.error(err);
    }
  };
  fetchCategories();
};

```

```

}, []);
```

```

// Open edit modal
const openEditModal = (course) => {
  setEditingCourse(course);
  setForm({
    title: course.title || "",
    description: course.description || "",
    category:
      course.category
      ? typeof course.category === "object"
        ? course.category._id
          : course.category
        : "",
    level: course.level || "",
    price: course.price || "",
    thumbnail: course.thumbnail || "",
  });
  setModalOpen(true);
  setError("");
};
```

```

const handleChange = (e) => {
  const { name, value } = e.target;
  setForm((prev) => ({ ...prev, [name]: value }));
};
```

```

const handleThumbnailUpload = async (e) => {
  const file = e.target.files[0];
  if (!file) return;

  const token = localStorage.getItem("token");
  const formData = new FormData();
  formData.append("thumbnail", file);

  try {
    setUploading(true);
    const res = await api.post("/instructor/course/upload-thumbnail", formData, {
      headers: {
        "Content-Type": "multipart/form-data",
        Authorization: `Bearer ${token}`,
      },
    });
    setForm((prev) => ({ ...prev, thumbnail: res.data.fileUrl }));
  }
```

```

} catch (err) {
  console.error(err);
  setError("Thumbnail upload failed");
} finally {
  setUploading(false);
}
};

const handleSubmit = async (e) => {
  e.preventDefault();
  if (!editingCourse) return;

  try {
    setSubmitting(true);
    const token = localStorage.getItem("token");
    const res = await api.put(`'/instructor/course/${editingCourse._id}`, form, {
      headers: { Authorization: `Bearer ${token}` },
    });

    setCourses((prev) =>
      prev.map((c) => (c._id === editingCourse._id ? res.data.course : c))
    );
    setModalOpen(false);
  } catch (err) {
    console.error(err);
    setError(err.response?.data?.message || "Failed to update course");
  } finally {
    setSubmitting(false);
  }
};

//if (loading) return <p style={{ padding: "20px" }}>Loading pending approvals...</p>;
if (loading) {
  return (
    <div style={{ display: "flex", flexDirection: "column", justifyContent: "center", alignItems: "center", height: "70vh", color: "#6f42c1", }}>
      <div style={{ border: "4px solid #f3f3f3", borderTop: "4px solid #6f42c1", borderRadius: "50%", width: "50px", height: "50px", animation: "spin 1s linear infinite", marginBottom: "20px", }} />
      <p>Loading pending approvals...</p>
      <style>{@keyframes spin { 0% { transform: rotate(0deg); } 100% { transform: rotate(360deg); } }}</style>
    </div>
  );
}

```

```
}

return (
  <div style={{ padding: "20px", background: "#f4f4f9", minHeight: "100vh" }}>
    <h2 style={{ marginBottom: "20px" }}>Pending Approvals</h2>
    {error && <p style={{ color: "red" }}>{error}</p>

    {courses.length === 0 ? (
      <p>No courses awaiting approval.</p>
    ) : (
      <div
        style={{
          display: "grid",
          gridTemplateColumns: "repeat(auto-fill, minmax(250px, 1fr))",
          gap: "15px",
        }}
      >
        {courses.map((course) => (
          <div
            key={course._id}
            style={{
              background: "#fff",
              padding: "12px",
              borderRadius: "10px",
              boxShadow: "0 2px 6px rgba(0,0,0,0.1)",
              border: course.status === "pendingApproval" ? "2px solid #ffc107" : "1px solid #ccc",
            }}
          >
            <img
              src={course.thumbnail ? `${BASE_URL}${course.thumbnail}` :
              "https://via.placeholder.com/250x120"}
              alt={course.title}
              style={{
                width: "100%",
                height: "120px",
                objectFit: "cover",
                borderRadius: "8px",
                marginBottom: "8px",
              }}
            />
            <h3 style={{ fontSize: "1.1rem", marginBottom: "5px" }}>{course.title}</h3>
            <p style={{ margin: "2px 0", fontSize: "0.85rem" }}>
              <strong>Category:</strong> {course.category?.name || "N/A"}
            </p>
        
```

```

<p style={{ margin: "2px 0", fontSize: "0.85rem" }}>
  <strong>Level:</strong> {course.level}
</p>
<p style={{ margin: "2px 0", fontSize: "0.85rem" }}>
  <strong>Price:</strong> {course.price > 0 ? `₹${course.price}` : "Free"}
</p>
<p style={{ margin: "2px 0", fontSize: "0.85rem" }}>
  <strong>Lessons:</strong> {course.lessons?.length || 0} |{" "}
  <strong>Exams:</strong> {course.exams?.length || 0}
</p>
<p style={{ margin: "2px 0", fontSize: "0.85rem" }}>
  <strong>Status:</strong> {course.status}
</p>
<button
  onClick={() => openEditModal(course)}
  style={{
    marginTop: "10px",
    width: "30%",
    padding: "4px 0",
    borderRadius: "5px",
    border: "none",
    background: "#9858deff",
    color: "#fff",
    cursor: "pointer",
  }}
>
  Edit
</button>
</div>
))}>
</div>
}

/* Edit Modal */
{modalOpen && (
<div
  style={{
    position: "fixed",
    top: 0,
    left: 0,
    width: "100%",
    height: "100%",
    background: "rgba(0,0,0,0.5)",
    display: "flex",
  }}>

```

```

justifyContent: "center",
alignItems: "center",
zIndex: 999,
padding: "10px",
})
>
<form
  onSubmit={handleSubmit}
  style={{
    background: "#fff",
    padding: "20px",
    borderRadius: "10px",
    width: "100%",
    maxWidth: "400px",
    boxShadow: "0 4px 12px rgba(0,0,0,0.2)",
  }}
>
  <h3 style={{ marginBottom: "15px", textAlign: "center", color: "#6f42c1" }}>Edit
Course</h3>

<input
  name="title"
  placeholder="Title"
  value={form.title}
  onChange={handleChange}
  required
  style={{ width: "100%", marginBottom: "10px", padding: "8px", borderRadius: "6px",
border: "1px solid #ccc" }}
/>
<textarea
  name="description"
  placeholder="Description"
  value={form.description}
  onChange={handleChange}
  style={{ width: "100%", marginBottom: "10px", padding: "8px", borderRadius: "6px",
border: "1px solid #ccc" }}
/>
<select
  name="category"
  value={form.category}
  onChange={handleChange}
  required
  style={{ width: "100%", marginBottom: "10px", padding: "8px", borderRadius: "6px",
border: "1px solid #ccc" }}

```

```

>
<option value="">Select Category</option>
{categories.map((cat) => (
  <option key={cat._id} value={cat._id}>{cat.name}</option>
))
}
</select>
<input
  name="level"
  placeholder="Level"
  value={form.level}
  onChange={handleChange}
  style={{ width: "100%", marginBottom: "10px", padding: "8px", borderRadius: "6px",
border: "1px solid #ccc" }}
/>
<input
  name="price"
  type="number"
  placeholder="Price"
  value={form.price}
  onChange={handleChange}
  style={{ width: "100%", marginBottom: "10px", padding: "8px", borderRadius: "6px",
border: "1px solid #ccc" }}
/>

<input type="file" onChange={handleThumbnailUpload} style={{ width: "100%",
marginBottom: "10px" }} />
{uploading && <p>Uploading thumbnail...</p>}
{form.thumbnail && !uploading && (
  <img src={`${BASE_URL}${form.thumbnail}`} alt="preview" style={{ width: "120px",
height: "70px", objectFit: "cover", borderRadius: "5px", marginBottom: "10px", border: "1px
solid #ccc" }} />
)}
<div style={{ display: "flex", justifyContent: "flex-end" }}>
<button
  type="button"
  onClick={() => setModalOpen(false)}
  style={{ marginRight: "10px", padding: "6px 12px", borderRadius: "5px", border: "1px
solid #ccc", background: "#f0f0f0", cursor: "pointer" }}
>
  Cancel
</button>
<button
  type="submit"

```

```

        disabled={submitting || uploading}
        style={{ padding: "6px 12px", borderRadius: "5px", border: "none", background:
      "#955defff", color: "#fff", cursor: "pointer" }}
      >
        {submitting ? "Saving..." : "Save"}
      </button>
    </div>
  </form>
</div>
)
</div>
);
}

export default PendingApprovals;

```

Category Requests Page

Category	Status	Requested At	Action
inst11	pending	15/11/2025, 10:21:23 pm	
inst123	rejected	15/11/2025, 9:53:28 pm	Re-Request

requestCategory.jsx

```

import React, { useEffect, useState } from "react";
import api from "../../api/api";

export default function RequestCategory() {
  const [name, setName] = useState("");
  const [message, setMessage] = useState("");
  const [loading, setLoading] = useState(false);

```

```

const [myRequests, setMyRequests] = useState([]);
const [checking, setChecking] = useState(false);
const [existsMessage, setExistsMessage] = useState("");
const [search, setSearch] = useState("");

const fetchMyRequests = async () => {
  try {
    const res = await api.get("/categories/my-requests");
    setMyRequests(res.data);
  } catch (err) {
    console.log(err);
  }
};

const checkCategoryExists = async (value) => {
  if (!value.trim()) {
    setExistsMessage("");
    return;
  }

  setChecking(true);
  try {
    const res = await api.get(`/categories/check?name=${value}`);
    setExistsMessage(
      res.data.exists ? "Category already exists!" : "Category is available"
    );
  } catch {
    setExistsMessage("");
  }
  setChecking(false);
};

const submitRequest = async () => {
  if (!name.trim()) return setMessage("Please enter a category");

  setLoading(true);

  try {
    await api.post("/categories/request", { name });
    setMessage("Category request submitted!");
    setName("");
    fetchMyRequests();
  } catch (error) {
    setMessage(error.response?.data?.message || "Something went wrong");
  }
};

```

```

    }

    setLoading(false);
};

const reRequest = async (catName) => {
    setLoading(true);
    try {
        await api.post("/categories/request", { name: catName });
        setMessage("Category request re-submitted!");
        fetchMyRequests();
    } catch (error) {
        setMessage(error.response?.data?.message || "Something went wrong");
    }
    setLoading(false);
};

useEffect(() => {
    fetchMyRequests();
}, []);

return (
    <div
        style={{
            background: "#faf7ff",
            padding: "25px",
            borderRadius: "12px",
            boxShadow: "0 0 18px rgba(0,0,0,0.08)",
        }}
    >
    <h2 style={{ color: "#6a0dad", fontWeight: "700", marginBottom: "20px" }}>
        Request New Category
    </h2>

    {/* INPUT BOX */}
    <div style={{ marginBottom: "10px" }}>
        <input
            type="text"
            style={{
                width: "100%",
                padding: "12px",
                borderRadius: "8px",
                border: "2px solid #ddd",
                outline: "none",
            }}
        />
    </div>
);

```

```

        transition: "0.3s",
        boxShadow: name ? "0 0 10px rgba(138, 43, 226, 0.3)" : "none",
    )}
placeholder="Enter category name"
value={name}
onChange={(e) => {
    setName(e.target.value);
    checkCategoryExists(e.target.value);
}}
/>

```

{checking ? (
 <p style={{ color: "#8a2be2", marginTop: "5px" }}>Checking...</p>
) : (
 <p
 style={{
 marginTop: "5px",
 fontWeight: "600",
 color: existsMessage.includes("available") ? "green" : "red",
 }}
 >
 {existsMessage}
 </p>
)
}
</div>

```

/* BUTTON */
<button
    onClick={submitRequest}
    disabled={loading}
    style={{
        padding: "6px 18px",
        backgroundColor: "#9d24f4ff",
        border: "none",
        borderRadius: "8px",
        color: "white",
        fontWeight: "600",
        cursor: "pointer",
        transition: "0.3s",
    }}
    >
    {loading ? "Submitting..." : "Submit Request"}
</button>

```

```
{message && (
  <p
    style={{
      marginTop: "15px",
      fontSize: "1.1rem",
      color: "red",
      fontWeight: "700",
    }}
  >
  {message}
</p>
)}

<hr style={{ margin: "25px 0" }} />

/* SEARCH */
<div style={{ marginBottom: "15px" }}>
  <input
    type="text"
    placeholder="Search categories..."
    style={{
      width: "100%",
      padding: "10px",
      borderRadius: "8px",
      border: "1px solid #ccc",
    }}
    value={search}
    onChange={(e) => setSearch(e.target.value)}
  />
</div>

<h4 style={{ color: "#6a0dad", fontWeight: "700" }}>My Category Requests</h4>

{myRequests.length === 0 ? (
  <p>No requests yet.</p>
) : (
  <table
    style={{
      width: "100%",
      borderCollapse: "separate",
      borderSpacing: "0 8px",
    }}
  >
  <thead>
```

```

<tr style={{ background: "#eee", borderRadius: "8px" }}>
  <th style={{ padding: "10px" }}>Category</th>
  <th>Status</th>
  <th>Requested At</th>
  <th>Action</th>
</tr>
</thead>

<tbody>
  {myRequests
    .filter((cat) =>
      cat.name.toLowerCase().includes(search.toLowerCase())
    )
    .map((cat) => (
      <tr
        key={cat._id}
        style={{
          background: "white",
          boxShadow: "0 0 6px rgba(0,0,0,0.05)",
          borderRadius: "8px",
        }}
      >
        <td style={{ padding: "10px" }}>{cat.name}</td>
        <td>
          <span
            style={{
              padding: "5px 12px",
              borderRadius: "6px",
              color: "white",
              background:
                cat.status === "approved"
                  ? "green"
                  : cat.status === "pending"
                  ? "orange"
                  : "red",
            }}
          >
            {cat.status}
          </span>
        </td>
        <td>{new Date(cat.createdAt).toLocaleString()}</td>
        <td>
          {cat.status === "rejected" && (

```

```

        <button
          onClick={() => reRequest(cat.name)}
          style={{
            padding: "4px 11px",
            background: "#9921efff",
            color: "white",
            borderRadius: "6px",
            border: "none",
            cursor: "pointer",
          }}
        >
          Re-Request
        </button>
      )
    </td>
  </tr>
)
)
</tbody>
</table>
)
</div>
);
}

```

Manage Lessons page

The screenshot shows the 'Manage Lessons' page of a learning management system. On the left, there's a purple sidebar with a 'Dashboard' header and several navigation items: 'Courses', 'Categories', 'Lessons' (which is expanded to show '- Manage Lessons'), 'Exams', 'Students', 'Earnings', and 'Analytics / Reports'. Below these is a 'Logout' button. The main content area has a light gray background. At the top, it says 'Manage Lessons'. Below that, there's a section for 'Mastering Next.js' which is marked as 'Status: approved'. It contains a 'Add Lesson' button and a note: 'Lessons cannot be modified because this course is approved.' A table lists three lessons: 'Getting Started with Mastering Next.js' (video, Yes, with Edit and Delete buttons), 'Next.js Pages and Routing' (pdf, No, with Edit and Delete buttons), and 'API Routes in Next.js' (text, No, with Edit and Delete buttons). Further down, there are sections for 'Advanced MERN Project' (Status: approved) and 'Next JS Fundamentals' (Status: approved). Under 'Advanced MERN Project', there are entries for 'Backend Development with Node.js' (Status: draft) and 'Frontend Frameworks with React' (Status: approved). At the bottom, there are two more entries: 'abc' (Status: pendingApproval) and 'xyz' (Status: rejected).

Title	Content Type	Preview	Actions
Getting Started with Mastering Next.js	video	Yes	Edit Delete
Next.js Pages and Routing	pdf	No	Edit Delete
API Routes in Next.js	text	No	Edit Delete

ManageLessons.jsx

```
import React, { useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import api from "../../api/api";

export default function ManageLessons() {
  const { courseId: selectedCourseId } = useParams();
  const token = localStorage.getItem("token");

  const EDITABLE_STATUSES = ["draft", "pendingApproval"];

  const [courses, setCourses] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState("");
  const [warning, setWarning] = useState("");
  const [modalOpen, setModalOpen] = useState(false);
  const [editingLesson, setEditingLesson] = useState(null);
  const [currentCourseId, setCurrentCourseId] = useState(null);
  const [submitting, setSubmitting] = useState(false);
  const [uploading, setUploading] = useState(false);

  const [form, setForm] = useState({
    title: "",
    contentType: "video",
    fileUrl: "",
    description: "",
    isPreviewFree: false,
  });

  useEffect(() => {
    const fetchCourses = async () => {
      try {
        setLoading(true);
        const res = await api.get("/instructor/courses", {
          headers: { Authorization: `Bearer ${token}` },
        });
      
```

```

const coursesData = res.data.courses.map((course) => ({
  ...course,
  lessons: [],
  expanded: false,
  lessonsLoading: false,
}));
```

```

  setCourses(coursesData);
} catch (err) {
  console.error(err);
  setError("Failed to fetch courses");
} finally {
  setLoading(false);
}
};

fetchCourses();
}, [token]);
```

```

useEffect(() => {
  if (selectedCourseId && courses.length > 0) {
    setCourses((prev) =>
      prev.map((course) =>
        course._id === selectedCourseId
          ? { ...course, expanded: true }
          : course
      )
    );
  }
}, [selectedCourseId, courses]);
```

```

const toggleCourse = async (courseId) => {
  setCourses((prev) =>
    prev.map((course) =>
      course._id === courseId ? { ...course, expanded: !course.expanded } : course
    )
  );
}
```

```

const course = courses.find((c) => c._id === courseId);

if (course && course.lessons.length === 0) {
  try {
    setCourses((prev) =>
      prev.map((c) =>
        c._id === courseId ? { ...c, lessonsLoading: true } : c
      )
    );
  }

  const res = await api.get(`/instructor/course/${courseId}/lessons`, {
    headers: { Authorization: `Bearer ${token}` },
  });

  setCourses((prev) =>
    prev.map((c) =>
      c._id === courseId
        ? { ...c, lessons: res.data.lessons || [], lessonsLoading: false }
        : c
    )
  );
}

} catch (err) {
  console.error(err);
  setError("Failed to fetch lessons");
}
}

};

const isCourseEditable = (status) => EDITABLE_STATUSES.includes(status);

const handleChange = (e) => {
  const { name, value, type, checked } = e.target;

  setForm((prev) => {
    let newForm = { ...prev, [name]: type === "checkbox" ? checked : value };
    if (name === "contentType" && value === "text") {

```

```

    newForm.fileUrl = "";
}
return newForm;
});
};

const handleFileUpload = async (e) => {
const file = e.target.files[0];
if (!file) return;

const formData = new FormData();
formData.append("file", file);

try {
  setUploading(true);
  const res = await api.post("/instructor/lesson/upload", formData, {
    headers: {
      "Content-Type": "multipart/form-data",
      Authorization: `Bearer ${token}`,
    },
  });
}

let uploadedPath = res.data.fileUrl;
if (uploadedPath.startsWith("http")) {
  const filename = uploadedPath.split("/").pop();
  uploadedPath = `/uploads/lessons/${filename}`;
}

setForm((prev) => ({ ...prev, fileUrl: uploadedPath }));
} catch (err) {
  console.error(err);
  setError("File upload failed");
} finally {
  setUploading(false);
}
};

```

```
const openModal = (courseld, lesson = null) => {
  const course = courses.find((c) => c._id === courseld);

  if (!isCourseEditable(course.status)) {
    setWarning("⚠ You cannot add/edit lessons for an approved course!");
    return; // Prevent opening modal
  }

  setCurrentCourseld(courseld);

  if (lesson) {
    setEditingLesson(lesson);
    setForm({
      title: lesson.title,
      contentType: lesson.contentType,
      fileUrl: lesson.fileUrl,
      description: lesson.description,
      isPreviewFree: lesson.isPreviewFree,
    });
  } else {
    setEditingLesson(null);
    setForm({
      title: "",
      contentType: "video",
      fileUrl: "",
      description: "",
      isPreviewFree: false,
    });
  }

  setModalOpen(true);
  setWarning("");
  setError("");
};

const handleSubmit = async (e) => {
  e.preventDefault();
```

```

const course = courses.find((c) => c._id === currentCourseId);
if (!isCourseEditable(course.status)) {
  setWarning("⚠ You cannot modify lessons in an approved course!");
  return;
}

if (form.contentType !== "text" && !form.fileUrl) {
  return setError("Please upload a file first");
}

setSubmitting(true);
try {
  const payload = {
    title: form.title,
    contentType: form.contentType,
    description: form.description,
    isPreviewFree: form.isPreviewFree,
    ...(form.contentType !== "text" && { fileUrl: form.fileUrl }),
  };
}

let res;
if (editingLesson) {
  res = await api.put(`/instructor/lesson/${editingLesson._id}`, payload, {
    headers: { Authorization: `Bearer ${token}` },
  });
}

setCourses((prev) =>
  prev.map((course) =>
    course._id !== currentCourseId
      ? course
      : {
        ...course,
        lessons: course.lessons.map((l) =>
          l._id === editingLesson._id ? res.data.lesson : l
        ),
      }
)

```

```

    )
);
} else {
  res = await api.post(`/instructor/course/${currentCourseId}/add-lesson`,
payload, {
  headers: { Authorization: `Bearer ${token}` },
});
}

setCourses((prev) =>
  prev.map((course) =>
    course._id !== currentCourseId
      ? course
      : { ...course, lessons: [...course.lessons, res.data.lesson] }
  )
);
}

setModalOpen(false);
setWarning("");
} catch (err) {
  console.error(err);
  setError(err.response?.data?.message || "Failed to save lesson");
} finally {
  setSubmitting(false);
}
};

const handleDelete = async (courseId, lessonId) => {
  const course = courses.find((c) => c._id === courseId);

  if (!isCourseEditable(course.status)) {
    setWarning("⚠ You cannot delete lessons from an approved course!");
    return;
  }

  if (!window.confirm("Are you sure you want to delete this lesson?")) return;
}

```

```

try {
  await api.delete(`/instructor/lesson/${lessonId}`, {
    headers: { Authorization: `Bearer ${token}` },
  });
}

setCourses((prev) =>
  prev.map((course) =>
    course._id !== courseId
      ? course
      : {
        ...course,
        lessons: course.lessons.filter((l) => l._id !== lessonId),
      }
  )
);
} catch (err) {
  console.error(err);
  setError("Failed to delete lesson");
}
};

if (loading) {
  return (
    <div
      style={{
        display: "flex",
        flexDirection: "column",
        justifyContent: "center",
        alignItems: "center",
        height: "70vh",
        color: "#6f42c1",
      }}
    >
      <div
        style={{
          border: "4px solid #f3f3f3",
          borderTop: "4px solid #6f42c1",

```

```

        borderRadius: "50%",
        width: "50px",
        height: "50px",
        animation: "spin 1s linear infinite",
        marginBottom: "20px",
    )}
/>
<p>Loading lessons...</p>
<style>
`@keyframes spin { 0% { transform: rotate(0deg); } 100% { transform:
rotate(360deg); } }`
</style>
</div>
);
}

return (
<div style={{ padding: "20px" }}>
<h2 style={{ marginBottom: "30px" }}>Manage Lessons</h2>
{error && <p style={{ color: "red" }}>{error}</p>}
{warning && <p style={{ color: "#d97706", fontWeight: "500"
}}>{warning}</p>}

{courses.map((course) => (
<div key={course._id} style={{ marginBottom: "20px" }}>
<h6
    onClick={() => toggleCourse(course._id)}
    style={{
        cursor: "pointer",
        userSelect: "none",
        display: "flex",
        justifyContent: "space-between",
        alignItems: "center",
        background: course._id === selectedCourseId ? "#e0d4f7" : "#f0f0f0",
        padding: "10px",
        borderRadius: "5px",
        fontWeight: "600",
    }}

```

```

        }}
      >
        {course.title} {course.expanded ? "▲" : "▼"}
        <span style={{ fontSize: "12px", opacity: 0.7 }}>
          Status: {course.status}
        </span>
      </h6>

      <div
        style={{
          maxHeight: course.expanded ? "1000px" : "0px",
          overflow: "hidden",
          transition: "max-height 0.3s ease",
        }}
      >
        <div style={{ padding: course.expanded ? "10px 0" : "0 0" }}>
          <button
            onClick={() => openModal(course._id)}
            disabled={submitting || !isCourseEditable(course.status)}
            style={{
              marginBottom: "10px",
              padding: "6px 12px",
              background: isCourseEditable(course.status) ? "#784bd9ff" : "#ccc",
              color: "#fff",
              border: "none",
              borderRadius: "5px",
              cursor: isCourseEditable(course.status) ? "pointer" : "not-allowed",
            }}
          >
            Add Lesson
          </button>
        </div>
        {(!isCourseEditable(course.status) && course.lessons.length > 0) && (
          <p style={{ color: "#d97706", fontWeight: "500" }}>
            Lessons cannot be modified because this course is approved.
          </p>
        )}
      </div>
    
```

```

{course.lessonsLoading ? (
  <p>Loading lessons...</p>
) : course.lessons.length === 0 ? (
  <p>No lessons for this course.</p>
) : (
  <table style={{ width: "100%", borderCollapse: "collapse" }}>
    <thead>
      <tr>
        <th style={thStyle}>Title</th>
        <th style={thStyle}>Content Type</th>
        <th style={thStyle}>Preview</th>
        <th style={thStyle}>Actions</th>
      </tr>
    </thead>
    <tbody>
      {course.lessons.map((lesson) => (
        <tr key={lesson._id}>
          <td style={tdStyle}>{lesson.title}</td>
          <td style={tdStyle}>{lesson.contentType}</td>
          <td style={tdStyle}>{lesson.isPreviewFree ? "Yes" : "No"}</td>
          <td style={tdStyle}>
            <button
              onClick={() => openModal(course._id, lesson)}
              disabled={submitting || !isCourseEditable(course.status)}
              style={actionButtonStyle("#0fb425ff")}
            >
              Edit
            </button>
            <button
              onClick={() => handleDelete(course._id, lesson._id)}
              disabled={submitting || !isCourseEditable(course.status)}
              style={actionButtonStyle("#dc3545")}
            >
              Delete
            </button>
          </td>
        </tr>
      ))
    </tbody>
  </table>
)

```

```
        </tr>
    )}
</tbody>
</table>
)}
</div>
</div>
</div>
)})

{modalOpen && (
<div style={modalOverlayStyle}>
<form onSubmit={handleSubmit} style={modalContentStyle}>
<h3>{editingLesson ? "Edit Lesson" : "Add Lesson"}</h3>

{warning && (
<p style={{ color: "#d97706", marginBottom: "10px", fontWeight: "500"
}}>
    {warning}
</p>
)}

<label>Title:</label>
<input
  name="title"
  value={form.title}
  onChange={handleChange}
  required
  style={inputStyle}
/>

<label>Content Type:</label>
<select
  name="contentType"
  value={form.contentType}
  onChange={handleChange}
  style={inputStyle}
```

```

>
<option value="video">Video</option>
<option value="pdf">Pdf</option>
<option value="text">Text</option>
</select>

{form.contentType !== "text" && (
  <>
    <label>Upload File:</label>
    <input type="file" onChange={handleFileUpload} style={inputStyle} />
    {uploading && <p>Uploading file...</p>}
    {form.fileUrl && !uploading && (
      <p>
        Uploaded: <a href={form.fileUrl} target="_blank">{form.fileUrl}</a>
      </p>
    )}
  </>
)})

<label>Description:</label>
<textarea
  name="description"
  value={form.description}
  onChange={handleChange}
  style={inputStyle}
/>

<label>
  <input
    type="checkbox"
    name="isPreviewFree"
    checked={form.isPreviewFree}
    onChange={handleChange}
  />{" "}
  Free Preview
</label>

```

```

<div style={{ marginTop: "20px", textAlign: "right" }}>
  <button
    type="button"
    onClick={() => { setModalOpen(false); setWarning(""); }}
    disabled={submitting}
    style={actionButtonStyle("#6c757d")}>
  </button>

  <button
    type="submit"
    disabled={submitting || uploading}
    style={actionButtonStyle("#198754", true)}>
    {submitting ? "Saving..." : editingLesson ? "Update" : "Add"}
  </button>
</div>
</form>
</div>
)}
</div>
);
}

// Styles
const thStyle = { border: "1px solid #ccc", padding: "8px", textAlign: "left" };
const tdStyle = { border: "1px solid #ccc", padding: "8px" };
const inputStyle = { width: "100%", marginBottom: "10px", padding: "6px",
borderRadius: "5px", border: "1px solid #ccc" };

const modalOverlayStyle = {
  position: "fixed",
  top: 0,
  left: 0,
  width: "100%",
  height: "100%",

```

```
background: "rgba(0,0,0,0.5)",
display: "flex",
justifyContent: "center",
alignItems: "center",
zIndex: 999,
overflowY: "auto",
padding: "20px",
};

const modalContentStyle = {
  background: "#fff",
  padding: "20px",
  borderRadius: "10px",
  width: "100%",
  maxWidth: "500px",
};

const actionButtonStyle = (bgColor, isPrimary = false) => ({
  marginLeft: "5px",
  padding: "6px 12px",
  background: bgColor,
  color: "#fff",
  border: "none",
  borderRadius: "5px",
  cursor: "pointer",
  fontWeight: isPrimary ? "600" : "500",
});
```

Manage Exams Page

The screenshot shows a dashboard interface with a purple sidebar on the left containing a navigation menu. The 'Exams' option is currently selected. The main content area is titled 'Manage Exams' and displays a list of exams with their details and status. A table at the bottom allows for editing or deleting specific exams.

Title	Duration	Questions	Actions
exam 1	20 mins	2	<button>Edit</button> <button>Delete</button>

manageExams.jsx

```
import React, { useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import api from "../../api/api";

export default function ManageExams() {
  const { courseId: selectedCourseId } = useParams();
  const token = localStorage.getItem("token");

  const EDITABLE_STATUSES = ["draft", "pendingApproval"]; // Courses with these statuses can be edited

  const [courses, setCourses] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState("");
  const [warning, setWarning] = useState(""); // Warning for non-editable courses
  const [modalOpen, setModalOpen] = useState(false);
  const [editingExam, setEditingExam] = useState(null);
  const [currentCourseId, setCurrentCourseId] = useState(null);
  const [submitting, setSubmitting] = useState(false);

  const [form, setForm] = useState({
    title: "",
```

```

duration: "",
questions: [],
});

const isCourseEditable = (status) => EDITABLE_STATUSES.includes(status);

// Fetch courses
useEffect(() => {
  const fetchCourses = async () => {
    try {
      setLoading(true);
      const res = await api.get("/instructor/courses", {
        headers: { Authorization: `Bearer ${token}` },
      });
      const coursesData = res.data.courses.map((course) => ({
        ...course,
        exams: [],
        expanded: false,
        examsLoading: false,
      }));
      setCourses(coursesData);
    } catch (err) {
      console.error(err);
      setError("Failed to fetch courses");
    } finally {
      setLoading(false);
    }
  };
  fetchCourses();
}, [token]);

// Auto expand selected
useEffect(() => {
  if (selectedCourseId && courses.length > 0) {
    setCourses((prev) =>
      prev.map((course) =>
        course._id === selectedCourseId
          ? { ...course, expanded: true }
          : course
      )
    );
  }
})

```

```

}, [selectedCourseId, courses]);

// Toggle expand/collapse and lazy load exams
const toggleCourse = async (courseId) => {
  setCourses((prev) =>
    prev.map((course) =>
      course._id === courseId
        ? { ...course, expanded: !course.expanded }
        : course
    )
  );
}

const course = courses.find((c) => c._id === courseId);
if (course && course.exams.length === 0) {
  try {
    setCourses((prev) =>
      prev.map((c) =>
        c._id === courseId ? { ...c, examsLoading: true } : c
      )
    );
  }

  const res = await api.get(`/instructor/course/${courseId}/exams`, {
    headers: { Authorization: `Bearer ${token}` },
  });

  setCourses((prev) =>
    prev.map((c) =>
      c._id === courseId
        ? { ...c, exams: res.data.exams || [], examsLoading: false }
        : c
    )
  );
} catch (err) {
  console.error(err);
  setError("Failed to fetch exams");
  setCourses((prev) =>
    prev.map((c) =>
      c._id === courseId ? { ...c, examsLoading: false } : c
    )
  );
}
};


```

```

// Handle form field change
const handleChange = (e) => {
  const { name, value } = e.target;
  setForm((prev) => ({ ...prev, [name]: value }));
};

// Question functions
const addQuestion = () => {
  setForm((prev) => ({
    ...prev,
    questions: [
      ...prev.questions,
      {
        questionText: '',
        options: ['', '', '', ''],
        correctAnswer: '',
        marks: 1,
      },
    ],
  }));
};

const removeQuestion = (index) => {
  setForm((prev) => ({
    ...prev,
    questions: prev.questions.filter((_, i) => i !== index),
  }));
};

const handleQuestionChange = (index, field, value) => {
  setForm((prev) => {
    const updated = [...prev.questions];
    updated[index][field] = value;
    return { ...prev, questions: updated };
  });
};

const handleOptionChange = (qIndex, optIndex, value) => {
  setForm((prev) => {
    const updated = [...prev.questions];
    updated[qIndex].options[optIndex] = value;
    return { ...prev, questions: updated };
  });
};

```

```

// Open modal
const openModal = (courseId, exam = null) => {
  const course = courses.find((c) => c._id === courseId);
  if (!isCourseEditable(course.status)) {
    setWarning("⚠ You cannot add/edit exams for an approved course!");
    return; // Prevent opening modal
  }

  setCurrentCourseId(courseId);
  if (exam) {
    setEditingExam(exam);
    setForm({
      title: exam.title,
      duration: exam.duration,
      questions: exam.questions || [],
    });
  } else {
    setEditingExam(null);
    setForm({ title: "", duration: "", questions: [] });
  }

  setModalOpen(true);
  setError("");
  setWarning("");
};

// Submit add/edit exam
const handleSubmit = async (e) => {
  e.preventDefault();

  const course = courses.find((c) => c._id === currentCourseId);
  if (!isCourseEditable(course.status)) {
    setWarning("⚠ You cannot modify exams in an approved course!");
    return;
  }

  setSubmitting(true);
  try {
    const payload = {
      title: form.title,
      duration: form.duration,
      questions: form.questions,
    };

```

```

let res;
if (editingExam) {
  res = await api.put(`/instructor/exam/${editingExam._id}`, payload, {
    headers: { Authorization: `Bearer ${token}` },
  });
}

setCourses((prev) =>
  prev.map((c) =>
    c._id !== currentCourseId
      ? c
      : {
        ...c,
        exams: c.exams.map((ex) =>
          ex._id === editingExam._id ? res.data.exam : ex
        ),
      }
  )
);
} else {
  res = await api.post(`/instructor/course/${currentCourseId}/add-exam`, payload, {
    headers: { Authorization: `Bearer ${token}` },
  });
}

setCourses((prev) =>
  prev.map((c) =>
    c._id !== currentCourseId
      ? c
      : { ...c, exams: [...c.exams, res.data.exam] }
  )
);
}

setModalOpen(false);
setWarning("");
} catch (err) {
  console.error(err);
  setError(err.response?.data?.message || "Failed to save exam");
} finally {
  setSubmitting(false);
}
};

// Delete exam

```

```

const handleDelete = async (courseld, examId) => {
  const course = courses.find((c) => c._id === courseld);
  if (!isCourseEditable(course.status)) {
    setWarning("⚠ You cannot delete exams from an approved course!");
    return;
  }

  if (!window.confirm("Are you sure you want to delete this exam?")) return;

  try {
    await api.delete(`/instructor/exam/${examId}`, {
      headers: { Authorization: `Bearer ${token}` },
    });
    setCourses((prev) =>
      prev.map((c) =>
        c._id !== courseld
          ? c
          : { ...c, exams: c.exams.filter((ex) => ex._id !== examId) }
      )
    );
  } catch (err) {
    console.error(err);
    setError("Failed to delete exam");
  }
};

if (loading) {
  return (
    <div style={{ display: "flex", flexDirection: "column", justifyContent: "center", alignItems: "center", height: "70vh", color: "#6f42c1" }}>
      <div style={{ border: "4px solid #f3f3f3", borderTop: "4px solid #6f42c1", borderRadius: "50%", width: "50px", height: "50px", animation: "spin 1s linear infinite", marginBottom: "20px" }} />
      <p>Loading exams...</p>
      <style>{@keyframes spin { 0% { transform: rotate(0deg); } 100% { transform: rotate(360deg); } }}</style>
    </div>
  );
}

return (
  <div style={{ padding: "20px" }}>
    <h2 style={{ marginBottom: "30px" }}>Manage Exams</h2>
    {error && <p style={{ color: "red" }}>{error}</p>}

```

```

{warning && <p style={{ color: "#d97706", fontWeight: "500" }}>{warning}</p>

{courses.map((course) =>
  <div key={course._id} style={{ marginBottom: "20px" }}>
    <h6
      onClick={() => toggleCourse(course._id)}
      style={{
        cursor: "pointer",
        userSelect: "none",
        display: "flex",
        justifyContent: "space-between",
        alignItems: "center",
        background: course._id === selectedCourseId ? "#e0d4f7" : "#f0f0f0",
        padding: "10px",
        borderRadius: "5px",
        fontWeight: "600",
      }}
    >
      {course.title} {course.expanded ? "▲" : "▼"}
      <span style={{ fontSize: "12px", opacity: 0.7 }}>
        Status: {course.status}
      </span>
    </h6>

    <div style={{ maxHeight: course.expanded ? "1000px" : "0px", overflow: "hidden", transition: "max-height 0.3s ease" }}>
      <div style={{ padding: course.expanded ? "10px 0" : "0 0" }}>
        <button
          onClick={() => openModal(course._id)}
          disabled={submitting || !isCourseEditable(course.status)}
          style={{
            marginBottom: "10px",
            padding: "6px 12px",
            background: isCourseEditable(course.status) ? "#784bd9ff" : "#ccc",
            color: "#fff",
            border: "none",
            borderRadius: "5px",
            cursor: isCourseEditable(course.status) ? "pointer" : "not-allowed",
          }}
        >
          Add Exam
        </button>

        {course.examsLoading ? (

```

```

<p>Loading exams...</p>
) : course.exams.length === 0 ? (
  <p>No exams for this course.</p>
) : (
  <table style={{ width: "100%", borderCollapse: "collapse" }}>
    <thead>
      <tr>
        <th style={thStyle}>Title</th>
        <th style={thStyle}>Duration</th>
        <th style={thStyle}>Questions</th>
        <th style={thStyle}>Actions</th>
      </tr>
    </thead>
    <tbody>
      {course.exams.map((exam) => (
        <tr key={exam._id}>
          <td style={tdStyle}>{exam.title}</td>
          <td style={tdStyle}>{exam.duration} mins</td>
          <td style={tdStyle}>{exam.questions?.length || 0}</td>
          <td style={tdStyle}>
            <button
              onClick={() => openModal(course._id, exam)}
              disabled={!isCourseEditable(course.status)}
              style={{ ...actionButtonStyle("#0fb425ff"), marginRight: "8px", cursor: isCourseEditable(course.status) ? "pointer" : "not-allowed" }}
            >
              Edit
            </button>
            <button
              onClick={() => handleDelete(course._id, exam._id)}
              disabled={!isCourseEditable(course.status)}
              style={{ ...actionButtonStyle("#dc3545"), cursor: isCourseEditable(course.status) ? "pointer" : "not-allowed" }}
            >
              Delete
            </button>
          </td>
        </tr>
      ))}
    </tbody>
  </table>
)
</div>
</div>

```

```

        </div>
    )})

/* Modal */
{modalOpen && (
<div style={modalOverlayStyle}>
<form onSubmit={handleSubmit} style={modalContentStyle}>
<h3>{editingExam ? "Edit Exam" : "Add Exam"}</h3>

{warning && <p style={{ color: "#d97706", marginBottom: "10px", fontWeight: "500" }}>{warning}</p>}

<label>Title:</label>
<input
  name="title"
  value={form.title}
  onChange={handleChange}
  placeholder="Enter exam title"
  required
  style={inputStyle}
/>

<label>Duration (minutes):</label>
<input
  name="duration"
  type="number"
  value={form.duration}
  onChange={handleChange}
  placeholder="Enter duration in minutes"
  required
  style={inputStyle}
/>

<h4>Questions</h4>
{form.questions.map((q, qIndex) => (
<div key={qIndex} style={questionContainerStyle}>
<label>Question Text:</label>
<input
  value={q.questionText}
  onChange={(e) => handleQuestionChange(qIndex, "questionText", e.target.value)}
  placeholder="Enter question text"
  required
  style={inputStyle}
/>

```

```

<label>Options:</label>
{q.options.map((opt, i) => (
  <input
    key={i}
    placeholder={`Option ${i + 1}`}
    value={opt}
    onChange={(e) => handleOptionChange(qIndex, i, e.target.value)}
    required
    style={inputStyle}
  />
))}

<label>Correct Answer:</label>
<select
  value={q.correctAnswer}
  onChange={(e) => handleQuestionChange(qIndex, "correctAnswer", e.target.value)}
  required
  style={inputStyle}>
  <option value="">-- Select Correct Answer --</option>
  {q.options.map(
    (opt, idx) =>
      opt.trim() && (
        <option key={idx} value={opt}>
          {opt}
        </option>
      )
  )}
</select>

<label>Marks:</label>
<input
  type="number"
  value={q.marks}
  onChange={(e) => handleQuestionChange(qIndex, "marks", e.target.value)}
  style={inputStyle}
/>

<button
  type="button"
  onClick={() => removeQuestion(qIndex)}
  style={actionButtonStyle("#dc3545")}>

```

```

        Remove
      </button>
    </div>
  )}

<button
  type="button"
  onClick={addQuestion}
  style={{ ...actionButtonStyle("#9a46e9ff"), marginBottom: "15px" }}
>
  + Add Question
</button>

<div style={{ display: "flex", justifyContent: "flex-end", gap: "10px" }}>
  <button
    type="button"
    onClick={() => { setModalOpen(false); setWarning(""); }}
    disabled={submitting}
    style={actionButtonStyle("#6c757d")}
  >
    Cancel
  </button>
  <button
    type="submit"
    disabled={submitting}
    style={actionButtonStyle("#198754", true)}
  >
    {submitting ? "Saving..." : editingExam ? "Update" : "Add"}
  </button>
</div>
</form>
</div>
)
);
};

// Styles
const thStyle = { border: "1px solid #ccc", padding: "8px", textAlign: "left" };
const tdStyle = { border: "1px solid #ccc", padding: "8px" };
const inputStyle = {
  width: "100%",
  marginBottom: "8px",
  padding: "6px",

```

```
borderRadius: "5px",
border: "1px solid #ccc",
boxSizing: "border-box",
};

const questionContainerStyle = {
  border: "1px solid #ccc",
  padding: "10px",
  marginBottom: "10px",
  borderRadius: "6px",
};

const modalOverlayStyle = {
  position: "fixed",
  top: 0,
  left: 0,
  width: "100%",
  height: "100%",
  background: "rgba(0,0,0,0.5)",
  display: "flex",
  justifyContent: "center",
  alignItems: "center",
  zIndex: 999,
  overflowY: "auto",
  padding: "20px",
};

const modalContentStyle = {
  background: "#fff",
  padding: "20px",
  borderRadius: "10px",
  width: "100%",
  maxWidth: "600px",
  maxHeight: "90vh",
  overflowY: "auto",
  boxSizing: "border-box",
};

const actionButtonStyle = (bgColor, isPrimary = false) => ({
  padding: "6px 12px",
  background: bgColor,
  color: "#fff",
  border: "none",
  borderRadius: "5px",
  cursor: "pointer",
  fontWeight: isPrimary ? "600" : "500",
});
```

Enrolled Student Lists

The screenshot shows a dashboard interface with a sidebar on the left containing navigation links for Courses, Categories, Lessons, Exams, Students, Earnings, and Analytics / Reports. The Students link is currently selected, indicated by a blue background. The main content area is titled "Enrolled Students" and features a search bar and three dropdown filters: "All Courses", "All Statuses", and "All Certificates". Below these is a table with columns: Student Name, Email, Course, Payment, Course Status, Progress, Expiry, Certificate, and Enrolled On. Three rows of data are displayed:

Student Name	Email	Course	Payment	Course Status	Progress	Expiry	Certificate	Enrolled On
student1	hmsonline3111@gmail.com	Next JS Fundamentals	₹199 complete	Active	0%	18/5/2026	Not Issued	19/11/2025
student1	hmsonline3111@gmail.com	Advanced MERN Project	₹299 complete	Active	80%	18/5/2026	Not Issued	19/11/2025
student1	hmsonline3111@gmail.com	Mastering Next.js	Free complete	Completed	100%	1/5/2026	View	2/11/2025

enrolledStudents.jsx

```
import { useEffect, useState } from "react";
import api from "../../api/api";

function EnrolledStudents() {
  const [students, setStudents] = useState([]);
  const [filtered, setFiltered] = useState([]);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState("");
  const [filters, setFilters] = useState({
    search: "",
    course: "",
    status: "",
    certificate: ""
  });

  const token = localStorage.getItem("token");

  useEffect(() => {
    const fetchStudents = async () => {
      try {
        setLoading(true);
        const res = await api.get("/instructor/enrolled-students", {
          headers: { Authorization: `Bearer ${token}` },
        });
      
```

```

const data = res.data.students || [];
setStudents(data);
setFiltered(data);
} catch (err) {
  console.error(err);
  setError("Failed to fetch enrolled students");
} finally {
  setLoading(false);
}
};

fetchStudents();
}, [token]);

// Apply filters
useEffect(() => {
  let data = [...students];

  if (filters.search) {
    const searchLower = filters.search.toLowerCase();
    data = data.filter(
      (s) =>
        s.student?.name?.toLowerCase().includes(searchLower) ||
        s.student?.email?.toLowerCase().includes(searchLower)
    );
  }

  if (filters.course)
    data = data.filter((s) => s.course?.title === filters.course);

  if (filters.status)
    data = data.filter((s) => s.status === filters.status);

  if (filters.certificate === "issued")
    data = data.filter((s) => s.certificate);
  else if (filters.certificate === "not-issued")
    data = data.filter((s) => !s.certificate);

  setFiltered(data);
}, [filters, students]);

const uniqueCourses = [
  ...new Set(students.map((s) => s.course?.title).filter(Boolean)),
];
const uniqueStatuses = [

```

```

...new Set(students.map((s) => s.status).filter(Boolean)),
];

// 🎨 Helper: color for course status
const getStatusColor = (status) => {
  switch (status?.toLowerCase()) {
    case "active":
    case "in-progress":
      return "orange";
    case "completed":
      return "green";
    case "expired":
    case "inactive":
      return "red";
    case "not-started":
      return "#6c757d";
    default:
      return "black";
  }
};

return (
  <div style={{ padding: "20px", background: "#f9f9ff", minHeight: "100vh" }}>
    <h2 style={{ marginBottom: "20px" }}>
      Enrolled Students
    </h2>

    {error && <p style={{ color: "red" }}>{error}</p>}

    {/* 🔎 Filter Section */}
    <div
      style={{
        display: "flex",
        gap: "10px",
        marginBottom: "20px",
        flexWrap: "wrap",
      }}
    >
      <input
        type="text"
        placeholder="Search by student name or email"
        value={filters.search}
        onChange={(e) => setFilters({ ...filters, search: e.target.value })}
        style={{

```

```

padding: "8px",
borderRadius: "5px",
border: "1px solid #ccc",
flex: "1",
minWidth: "200px",
}}
/>
<select
value={filters.course}
onChange={(e) => setFilters({ ...filters, course: e.target.value })}
style={{
padding: "8px",
borderRadius: "5px",
border: "1px solid #ccc",
}}
>
<option value="">All Courses</option>
{uniqueCourses.map((course) => (
<option key={course} value={course}>
{course}
</option>
))}
</select>

/*  Course Status Filter */
<select
value={filters.status}
onChange={(e) => setFilters({ ...filters, status: e.target.value })}
style={{
padding: "8px",
borderRadius: "5px",
border: "1px solid #ccc",
}}
>
<option value="">All Statuses</option>
{uniqueStatuses.map((status) => (
<option key={status} value={status}>
{status.charAt(0).toUpperCase() + status.slice(1)}
</option>
))}
</select>

<select

```

```

        value={filters.certificate}
        onChange={(e) =>
          setFilters({ ...filters, certificate: e.target.value })
        }
        style={{
          padding: "8px",
          borderRadius: "5px",
          border: "1px solid #ccc",
        }}
      >
      <option value="">All Certificates</option>
      <option value="issued">Issued</option>
      <option value="not-issued">Not Issued</option>
    </select>
  </div>

  {loading ? (
    <div style={{ display: "flex", flexDirection: "column", justifyContent: "center", alignItems: "center", height: "70vh", color: "#6f42c1", }}>
      <div style={{ border: "4px solid #f3f3f3", borderTop: "4px solid #6f42c1", borderRadius: "50%", width: "50px", height: "50px", animation: "spin 1s linear infinite", marginBottom: "20px", }} />
      <p>Loading enrolled students...</p>
      <style>{@keyframes spin { 0% { transform: rotate(0deg); } 100% { transform: rotate(360deg); } }}</style>
    </div>
  ) : filtered.length === 0 ? (
    <p>No enrolled students found.</p>
  ) : (
    <table
      style={{
        width: "100%",
        borderCollapse: "collapse",
        background: "#fff",
        borderRadius: "10px",
        overflow: "hidden",
        boxShadow: "0 2px 8px rgba(0,0,0,0.1)",
      }}
    >
      <thead style={{ background: "#6f42c1", color: "#fff" }}>
        <tr>
          <th style={{ padding: "10px" }}>Student Name</th>

```

```

<th style={{ padding: "10px" }}>Email</th>
<th style={{ padding: "10px" }}>Course</th>
<th style={{ padding: "10px" }}>Payment</th>
<th style={{ padding: "10px" }}>Course Status</th>
<th style={{ padding: "10px" }}>Progress</th>
<th style={{ padding: "10px" }}>Expiry</th>
<th style={{ padding: "10px" }}>Certificate</th>
<th style={{ padding: "10px" }}>Enrolled On</th>
</tr>
</thead>
<tbody>
  {filtered.map((s) => (
    <tr
      key={s._id}
      style={{
        borderBottom: "1px solid #eee",
        textAlign: "center",
      }}
    >
      <td style={{ padding: "8px" }}>{s.student?.name}</td>
      <td style={{ padding: "8px" }}>{s.student?.email}</td>
      <td style={{ padding: "8px" }}>{s.course?.title}</td>

      <td style={{ padding: "8px" }}>
        {s.amount > 0 ? `₹${s.amount}` : "Free"} <br />
        <span
          style={{
            color:
              s.paymentStatus === "complete"
                ? "green"
                : s.paymentStatus === "failed"
                ? "red"
                : "orange",
            fontWeight: "bold",
          }}
        >
          {s.paymentStatus}
        </span>
      </td>

      /* 🎨 Colored Course Status */
      <td
        style={{
          padding: "8px",
        }}
      >
        {courseStatusColor(s.courseStatus)}
      </td>
    
```

```

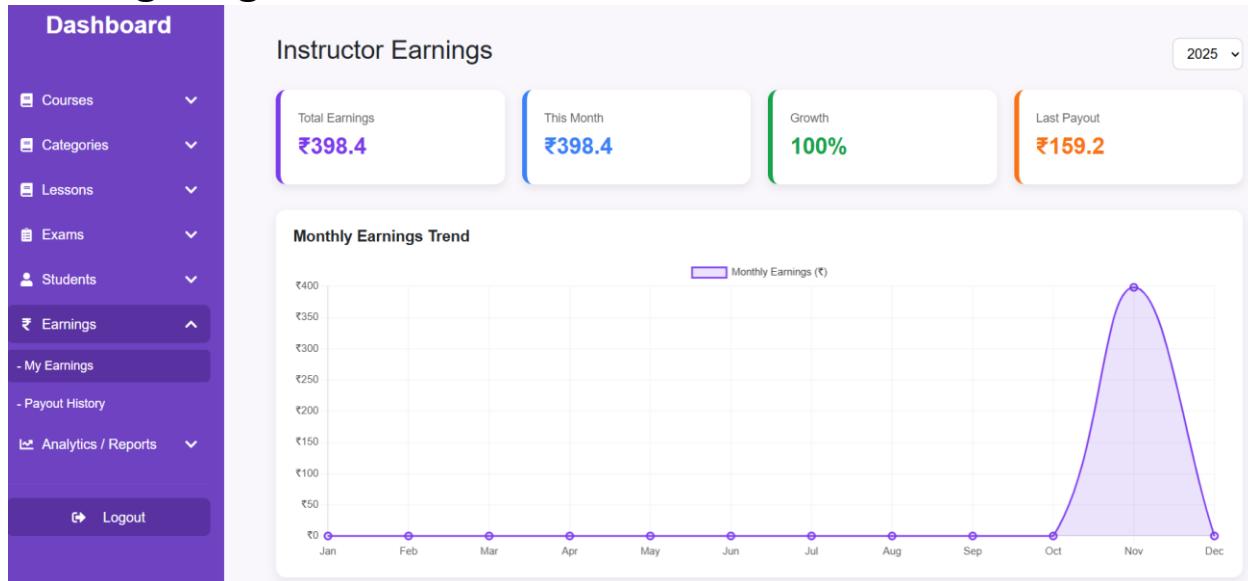
        textTransform: "capitalize",
        color: getStatusColor(s.status),
        fontWeight: "bold",
    }}
>
{s.status || "N/A"
</td>

<td style={{ padding: "8px" }}>{s.progress}%</td>
<td style={{ padding: "8px" }}>
    {new Date(s.expiryDate).toLocaleDateString()}
</td>
<td style={{ padding: "8px" }}>
    {s.certificate ? (
        <a
            href={`${import.meta.env.VITE_BASE_URL}${s.certificate.startsWith("/") ?
                s.certificate :
                "/" + s.certificate
            }`}
            target="_blank"
            rel="noopener noreferrer"
            style={{
                color: "#6f42c1",
                textDecoration: "underline",
            }}
        >
            View
        </a>
    ) : (
        "Not Issued"
    )}
</td>
<td style={{ padding: "8px" }}>
    {new Date(s.createdAt).toLocaleDateString()}
</td>
</tr>
))}
</tbody>
</table>
)}
</div>
);
}
}

```

```
export default EnrolledStudents;
```

Earnings Page



Earnings.jsx

```
import { useEffect, useState } from "react";
import { Line } from "react-chartjs-2";
import {
  Chart as ChartJS,
  LineElement,
  CategoryScale,
  LinearScale,
  PointElement,
  Filler,
  Tooltip,
  Legend,
} from "chart.js";
import api from "../../api/api";

// ✅ Register Chart.js components
ChartJS.register(
  LineElement,
  CategoryScale,
  LinearScale,
```

```

PointElement,
Filler,
Tooltip,
Legend
);

export default function InstructorEarnings() {
  const [data, setData] = useState({
    totalEarning: 0,
    monthly: {},
    lastPayout: 0,
  });
  const [loading, setLoading] = useState(true);
  const [selectedYear, setSelectedYear] = useState("2025");

  const months = [
    "Jan", "Feb", "Mar", "Apr", "May", "Jun",
    "Jul", "Aug", "Sep", "Oct", "Nov", "Dec",
  ];

  // ✨ Fetch earnings data
  useEffect(() => {
    const fetchEarnings = async () => {
      try {
        setLoading(true);
        const res = await api.get(`/instructor/earnings?year=${selectedYear}`);
        console.log("Fetched Earnings:", res.data);
        setData(res.data || {});
      } catch (err) {
        console.error("Error fetching earnings:", err);
        setData({ totalEarning: 0, monthly: {}, lastPayout: 0 });
      } finally {
        setLoading(false);
      }
    };
    fetchEarnings();
  }, [selectedYear]);

  // ✅ Ensure full 12 months data (fill missing with 0)
  const monthlyData = Array.from({ length: 12 }, (_, i) => {
    return Number(data?.monthly?.[i + 1] || 0);
  });
}

```

```

// ✅ Get current and previous month for "This Month" and Growth
const currentMonthIndex = new Date().getMonth(); // 0 = Jan, 11 = Dec
const thisMonth = monthlyData[currentMonthIndex] || 0;
const prevMonth =
  currentMonthIndex > 0 ? monthlyData[currentMonthIndex - 1] || 0 : 0;

const growth = prevMonth === 0 ? "100" : (((thisMonth - prevMonth) / prevMonth) *
100).toFixed(2) + "%";

// ✅ Chart Data
const chartData = {
  labels: months,
  datasets: [
    {
      label: "Monthly Earnings (₹)",
      data: monthlyData,
      borderColor: "#7c3aed",
      backgroundColor: "rgba(124,58,237,0.15)",
      tension: 0.4,
      fill: true,
      pointRadius: 4,
      borderWidth: 2,
    },
  ],
};

// ✅ Chart Options
const chartOptions = {
  responsive: true,
  maintainAspectRatio: false,
  plugins: {
    legend: { position: "top", labels: { color: "#333" } },
    tooltip: {
      callbacks: {
        label: (ctx) => `₹${ctx.parsed.y.toLocaleString()}`,
      },
    },
    scales: {
      x: {
        ticks: { color: "#555" },
        grid: { color: "#f3f4f6" },
      },
    },
  },
};

```

```

y: {
  beginAtZero: true,
  ticks: {
    callback: (val) => `₹${val}`,
    color: "#555",
  },
  grid: { color: "#f3f4f6" },
},
},
};

if (loading) {
  return (
    <div style={{ display: "flex", flexDirection: "column", justifyContent: "center", alignItems: "center", height: "70vh", color: "#6f42c1", }}>
      <div style={{ border: "4px solid #f3f3f3", borderTop: "4px solid #6f42c1", borderRadius: "50%", width: "50px", height: "50px", animation: "spin 1s linear infinite", marginBottom: "20px", }} />
      <p>Loading earnings...</p>
      <style>{@keyframes spin { 0% { transform: rotate(0deg); } 100% { transform: rotate(360deg); } }}</style>
    </div>
  );
}

return (
  <div style={styles.wrapper}>
    {/* Header */}
    <div style={styles.header}>
      <h1 style={styles.heading}>Instructor Earnings</h1>
      <select
        style={styles.select}
        value={selectedYear}
        onChange={(e) => setSelectedYear(e.target.value)}
      >
        {[2025, 2024, 2023].map((yr) => (
          <option key={yr} value={yr}>{yr}</option>
        )))
      </select>
    </div>

    {/* Summary Cards */}
    <div style={styles.cardRow}>
      <Card

```

```

        title="Total Earnings"
        value={`₹${Number(data.totalEarning || 0).toLocaleString()}`}
        color="#7c3aed"
    />
    <Card
        title="This Month"
        value={`₹${thisMonth.toLocaleString()}`}
        color="#3b82f6"
    />
    <Card
        title="Growth"
        value={`${growth}%`}
        color={growth >= 0 ? "#16a34a" : "#dc2626"}
    />
    <Card
        title="Last Payout"
        value={`₹${Number(data.lastPayout || 0).toLocaleString()}`}
        color="#f97316"
    />
</div>

/* Chart */
<div style={styles.chartBox}>
    <h3 style={styles.chartTitle}>Monthly Earnings Trend</h3>
    <div style={{ height: 350 }}>
        <Line data={chartData} options={chartOptions} />
    </div>
</div>
</div>
);

/*
 * 📈 Card Component *
 */
function Card({ title, value, color }) {
    return (
        <div style={{ ...styles.card, borderLeft: `6px solid ${color}` }}>
            <span style={styles.cardTitle}>{title}</span>
            <h2 style={{ ...styles.cardValue, color }}>{value}</h2>
        </div>
    );
}

/*
 * 💃 Styles *
 */
const styles = {

```

```
wrapper: {
  padding: 30,
  maxWidth: 1200,
  margin: "0 auto",
},
header: {
  display: "flex",
  justifyContent: "space-between",
  alignItems: "center",
  flexWrap: "wrap",
  marginBottom: 20,
  gap: 10,
},
heading: {
  fontSize: 30,
  fontWeight: 500,
  WebkitBackgroundClip: "text"
},
select: {
  padding: "8px 12px",
  borderRadius: 8,
  border: "1px solid #ccc",
  cursor: "pointer",
},
cardRow: {
  display: "grid",
  gridTemplateColumns: "repeat(auto-fill, minmax(220px, 1fr))",
  gap: 20,
  marginTop: 20,
},
card: {
  background: "#fff",
  padding: 20,
  borderRadius: 12,
  boxShadow: "0 4px 10px rgba(0,0,0,0.08)",
  transition: "transform 0.2s ease",
},
cardTitle: {
  fontSize: 14,
  opacity: 0.7,
},
cardValue: {
  fontSize: 26,
  fontWeight: "bold",
}
```

```

        marginTop: 6,
    },
    chartBox: {
        background: "#fff",
        padding: 20,
        borderRadius: 12,
        marginTop: 30,
        boxShadow: "0 4px 12px rgba(0,0,0,0.08)",
    },
    chartTitle: {
        fontSize: 18,
        marginBottom: 15,
        fontWeight: 600,
    },
},
);

```

Payout History Page

The screenshot shows a user interface for a payout history page. On the left, there is a dark purple sidebar titled 'Dashboard' with a navigation menu. The menu items include 'Courses', 'Categories', 'Lessons', 'Exams', 'Students', 'Earnings' (which is currently selected), and 'Analytics / Reports'. Below the menu is a 'Logout' button. The main content area has a light gray background. At the top, it says 'Payout History' and includes a date selector set to '2025'. The main content displays two rows of payout information in a table format:

#	Date	Amount	Payment Method	Status
1	19 Nov 2025	₹159.2	Razorpay	completed
2	19 Nov 2025	₹239.2	Razorpay	completed

At the bottom of the main content area, there are navigation buttons for 'First', 'Prev', 'Page 1 of 1', 'Next', and 'Last >>'.

PayoutHistory.jsx

```

import { useEffect, useState } from "react";
import api from "../../api/api";

export default function PayoutHistory() {
    const [payouts, setPayouts] = useState([]);
    const [loading, setLoading] = useState(true);
    const [page, setPage] = useState(1);
    const [totalPages, setTotalPages] = useState(1);
}

```

```

const [filterYear, setFilterYear] = useState(new Date().getFullYear());
const limit = 10;

// Generate current year and past 5 years
const generateYears = () => {
  const currentYear = new Date().getFullYear();
  return Array.from({ length: 6 }, (_, i) => currentYear - i);
};

useEffect(() => {
  const fetchPayouts = async () => {
    try {
      setLoading(true);
      const res = await api.get(
        `/instructor/payouts?page=${page}&limit=${limit}&year=${filterYear}`
      );
      setPayouts(res.data.payouts);
      setTotalPages(res.data.totalPages);
    } catch (err) {
      console.error("Error fetching payout history:", err);
      setPayouts([]);
    } finally {
      setLoading(false);
    }
  };
  fetchPayouts();
}, [page, filterYear]);

const formatDate = (dateStr) => {
  const date = new Date(dateStr);
  return date.toLocaleDateString("en-GB", {
    day: "2-digit",
    month: "short",
    year: "numeric",
  });
};

const statusColor = (status) => {
  switch (status.toLowerCase()) {
    case "completed":
      return "#16a34a";
    case "pending":
      return "#f59e0b";
    case "failed":

```

```

        return "#dc2626";
    default:
        return "#6b7280";
    }
};

if (loading) {
    return (
        <div style={{ display: "flex", flexDirection: "column", justifyContent: "center", alignItems: "center", height: "70vh", color: "#6f42c1", }}>
            <div style={{ border: "4px solid #f3f3f3", borderTop: "4px solid #6f42c1", borderRadius: "50%", width: "50px", height: "50px", animation: "spin 1s linear infinite", marginBottom: "20px", }} />
                <p>Loading payout history...</p>
                <style>{@keyframes spin { 0% { transform: rotate(0deg); } 100% { transform: rotate(360deg); } }}</style>
        </div>
    );
}

return (
    <div style={styles.wrapper}>
        <div style={styles.header}>
            <h1 style={styles.heading}>Payout History</h1>
            <select
                style={styles.select}
                value={filterYear}
                onChange={(e) => {
                    setPage(1);
                    setFilterYear(Number(e.target.value));
                }}
            >
                {generateYears().map((yr) => (
                    <option key={yr} value={yr}>{yr}</option>
                )))
            </select>
        </div>

        {payouts.length === 0 ? (
            <p style={{ textAlign: "center", marginTop: 40 }}>No payouts found for {filterYear}.</p>
        ) : (
            <div style={styles.cardsContainer}>
                {payouts.map((p, idx) => (
                    <div key={p._id} style={styles.card}>

```

```

        <CardRow label="#" value={(page - 1) * limit + idx + 1} />
        <CardRow label="Date" value={formatDate(p.paymentDate)} />
        <CardRow label="Amount" value={`₹${p.instructorEarning.toLocaleString()}`} />
        <CardRow label="Payment Method" value={p.paymentMethod || "N/A"} />
        <CardRow
            label="Status"
            value={p.status}
            valueColor={statusColor(p.status)}
        />
    </div>
)}
```

```

</div>
})
}

/* Pagination */
<div style={styles.pagination}>
    <button onClick={() => setPage(1)} disabled={page === 1}>"<<" First</button>
    <button onClick={() => setPage(p => Math.max(p - 1, 1))} disabled={page ===
1}>Prev</button>
    <span>Page {page} of {totalPages}</span>
    <button onClick={() => setPage(p => Math.min(p + 1, totalPages))} disabled={page ===
totalPages}>Next</button>
    <button onClick={() => setPage(totalPages)} disabled={page === totalPages}>Last
    {">>"></button>
</div>
</div>
);
}

/* Card row */
function CardRow({ label, value, valueColor }) {
    return (
        <div style={styles.cardRow}>
            <span style={styles.cardLabel}>{label}</span>
            <span style={{ fontWeight: valueColor ? "bold" : "500", color: valueColor || "#111" }}>
                {value}
            </span>
        </div>
    );
}

/* Styles */
const styles = {
    wrapper: {

```

```
padding: 30,  
maxWidth: 1200,  
margin: "0 auto",  
},  
header: {  
  display: "flex",  
  justifyContent: "space-between",  
  alignItems: "center",  
  flexWrap: "wrap",  
  marginBottom: 20,  
  gap: 10,  
},  
heading: {  
  fontSize: 28,  
  fontWeight: 500,  
  WebkitBackgroundClip: "text",  
},  
select: {  
  padding: "8px 12px",  
  borderRadius: 8,  
  border: "1px solid #ccc",  
  cursor: "pointer",  
},  
cardsContainer: {  
  display: "grid",  
  gridTemplateColumns: "repeat(auto-fill, minmax(250px, 1fr))",  
  gap: 20,  
},  
card: {  
  background: "#fff",  
  padding: 20,  
  borderRadius: 12,  
  boxShadow: "0 4px 12px rgba(0,0,0,0.08)",  
  transition: "transform 0.2s ease",  
},  
cardRow: {  
  display: "flex",  
  justifyContent: "space-between",  
  marginBottom: 8,  
},  
cardLabel: {  
  fontWeight: 600,  
  color: "#6b7280",  
},
```

```

pagination: {
  marginTop: 30,
  display: "flex",
  justifyContent: "center",
  alignItems: "center",
  gap: 10,
  flexWrap: "wrap",
},
};

```

Course Analytics Page

The screenshot displays the Course Analytics Page. On the left, a purple sidebar titled 'Dashboard' contains links for Courses, Categories, Lessons, Exams, Students, Earnings, and Analytics / Reports (which is currently selected). Below this is a 'Logout' button. The main content area is titled 'Course Analytics'.

Enrollments per Course: A bar chart titled 'Enrollments per Course' shows the number of enrollments for six courses. The y-axis ranges from 0 to 1. The x-axis lists the courses: Mastering Next.js, Advanced MERN Project, Next JS Fundamentals, Backend Development with Node.js, Frontend Frameworks with React, abc, and xyz. All courses except 'Backend Development with Node.js' have 1 enrollment.

Course	Enrollments	Completed	Completion Rate (%)	Revenue (₹)
Mastering Next.js	1	1	100.0%	₹0
Advanced MERN Project	1	0	0.0%	₹299
Next JS Fundamentals	1	0	0.0%	₹199
Backend Development with Node.js	0	0	0%	₹0
Frontend Frameworks with React	0	0	0%	₹0
abc	0	0	0%	₹0
xyz	0	0	0%	₹0

Revenue Distribution: A pie chart titled 'Revenue Distribution' shows the revenue contribution of each course. The chart is divided into two main segments: a green segment for 'xyz' (₹199) and a yellow segment for 'Advanced MERN Project' (₹299). Smaller segments for other courses are visible but unlabeled.

courseAnalytics.jsx

```

import { useState, useEffect } from "react";
import api from "../../api/api";
import {
  BarChart,
  Bar,
  XAxis,
  YAxis,
  Tooltip,
}

```

```

Legend,
PieChart,
Pie,
Cell,
ResponsiveContainer,
} from "recharts";

function CourseAnalytics() {
  const [analytics, setAnalytics] = useState([]);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState("");
  const token = localStorage.getItem("token");

  useEffect(() => {
    const fetchAnalytics = async () => {
      try {
        setLoading(true);
        const res = await api.get("/instructor/course-analytics", {
          headers: { Authorization: `Bearer ${token}` },
        });
        setAnalytics(res.data.analytics || []);
      } catch (err) {
        console.error(err);
        setError("Failed to fetch course analytics");
      } finally {
        setLoading(false);
      }
    };
    fetchAnalytics();
  }, [token]);
}

const COLORS = ["#6f42c1", "#28a745", "#ffc107", "#dc3545", "#17a2b8"];

// ✅ Format chart data
const revenueData = analytics.map((c) => ({
  name: c.courseTitle.length > 15 ? c.courseTitle.slice(0, 15) + "..." : c.courseTitle,
  value: c.revenue || 0,
}));

const enrollmentData = analytics.map((c) => ({
  course: c.courseTitle.length > 15 ? c.courseTitle.slice(0, 15) + "..." : c.courseTitle,
  enrollments: c.totalStudents || c.totalEnrollments || 0,
}));

```

```

return (
  <div
    style={{
      padding: "20px",
      background: "#f9f9ff",
      minHeight: "100vh",
      fontFamily: "Poppins, sans-serif",
    }}
  >
  <h2
    style={{
      marginBottom: "25px",
      fontWeight: "500",
    }}
  >
  Course Analytics
  </h2>

{error && <p style={{ color: "red" }}>{error}</p>}
{loading ? (
  <div style={{ display: "flex", flexDirection: "column", justifyContent: "center", alignItems: "center", height: "70vh", color: "#6f42c1", }}>
    <div style={{ border: "4px solid #f3f3f3", borderTop: "4px solid #6f42c1", borderRadius: "50%", width: "50px", height: "50px", animation: "spin 1s linear infinite", marginBottom: "20px", }} />
    <p>Loading...</p>
    <style>{@keyframes spin { 0% { transform: rotate(0deg); } 100% { transform: rotate(360deg); } }}</style>
  </div>
) : analytics.length === 0 ? (
  <p style={{ textAlign: "center" }}>No course analytics available.</p>
) : (
  <>
    /* ===== Charts Section ===== */
    <div
      style={{
        display: "flex",
        flexWrap: "wrap",
        gap: "30px",
        justifyContent: "center",
        marginBottom: "40px",
      }}
    >
      /* Enrollments Bar Chart */

```

```

<div
  style={{
    background: "#fff",
    padding: "20px",
    borderRadius: "12px",
    boxShadow: "0 4px 12px rgba(0,0,0,0.1)",
    flex: "1 1 500px",
    minWidth: "350px",
  }}
>
  <h4
    style={{
      color: "#6f42c1",
      marginBottom: "15px",
      textAlign: "center",
    }}
  >
    Enrollments per Course
  </h4>
  <div style={{ width: "100%", height: 300, overflowX: "auto" }}>
    <ResponsiveContainer width="100%" height="100%">
      <BarChart
        data={enrollmentData}
        margin={{ top: 20, right: 30, left: 20, bottom: 70 }}
      >
        <XAxis
          dataKey="course"
          interval={0}
          angle={-40}
          textAnchor="end"
          height={80}
          tick={{ fontSize: 12 }}
        />
        <YAxis />
        <Tooltip />
        <Legend />
        <Bar dataKey="enrollments" fill="#6f42c1" barSize={40} />
      </BarChart>
    </ResponsiveContainer>
  </div>
</div>

/* Revenue Pie Chart */
<div

```

```

        style={{
          background: "#fff",
          padding: "20px",
          borderRadius: "12px",
          boxShadow: "0 4px 12px rgba(0,0,0,0.1)",
          flex: "1 1 350px",
          textAlign: "center",
        }}
      >
      <h4
        style={{
          color: "#6f42c1",
          marginBottom: "15px",
          textAlign: "center",
        }}
      >
        Revenue Distribution
      </h4>
      <div style={{ width: "100%", height: 300 }}>
        <ResponsiveContainer>
          <PieChart>
            <Pie
              data={revenueData}
              dataKey="value"
              nameKey="name"
              outerRadius={100}
              label
            >
              {revenueData.map((entry, index) => (
                <Cell
                  key={`cell-${index}`}
                  fill={COLORS[index % COLORS.length]}
                />
              )))
            </Pie>
            <Tooltip />
            <Legend />
          </PieChart>
        </ResponsiveContainer>
      </div>
    </div>
  </div>

  /* ===== Table Section ===== */

```

```

<div
  style={{
    overflowX: "auto",
    borderRadius: "12px",
    boxShadow: "0 4px 12px rgba(0,0,0,0.1)",
    background: "#fff",
  }}
>
  <table
    style={{
      width: "100%",
      borderCollapse: "collapse",
      minWidth: "600px",
    }}
  >
    <thead style={{ background: "#6f42c1", color: "fff" }}>
      <tr>
        <th style={{ padding: "12px" }}>Course</th>
        <th style={{ padding: "12px" }}>Enrollments</th>
        <th style={{ padding: "12px" }}>Completed</th>
        <th style={{ padding: "12px" }}>Completion Rate (%)</th>
        <th style={{ padding: "12px" }}>Revenue (₹)</th>
      </tr>
    </thead>
    <tbody>
      {analytics.map((a, index) => (
        <tr
          key={index}
          style={{
            textAlign: "center",
            borderBottom: "1px solid #eee",
          }}
        >
          <td style={{ padding: "10px" }}>{a.courseTitle}</td>
          <td style={{ padding: "10px" }}>{a.totalStudents}</td>
          <td style={{ padding: "10px" }}>{a.completedStudents}</td>
          <td style={{ padding: "10px" }}>{a.completionRate}%</td>
          <td style={{ padding: "10px" }}>₹{a.revenue}</td>
        </tr>
      ))}
    </tbody>
  </table>
</div>
</>

```

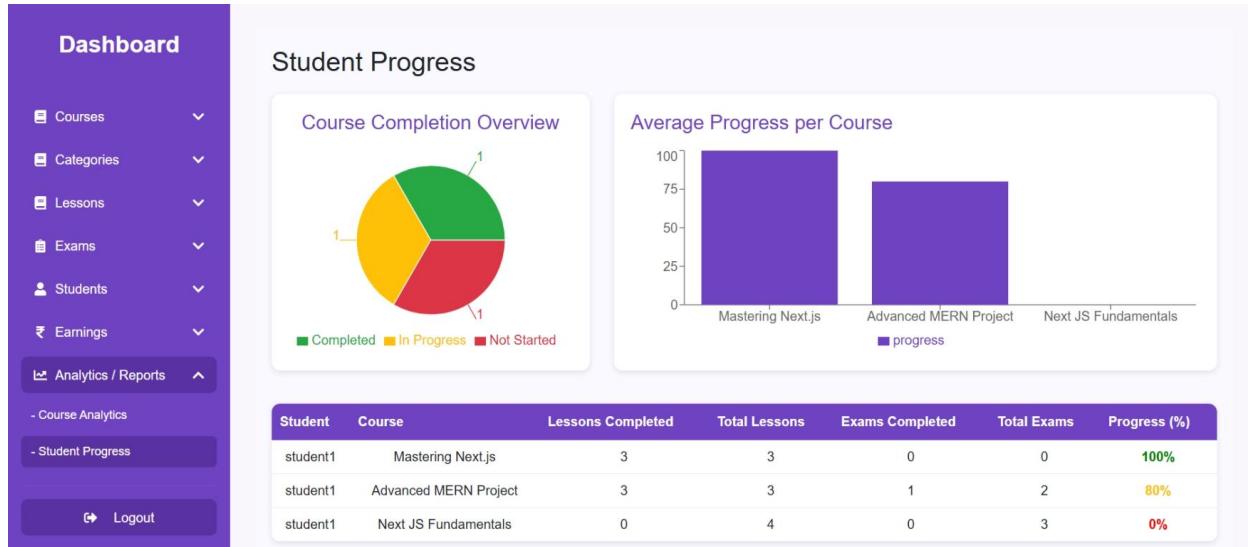
```

        )}
      </div>
    );
}

export default CourseAnalytics;

```

Student Progress Page



studentProgress.jsx

```

import { useState, useEffect } from "react";
import api from "../../api/api";
import {
  PieChart,
  Pie,
  Cell,
  BarChart,
  Bar,
  XAxis,
  YAxis,
  Tooltip,
  Legend,
  ResponsiveContainer,
} from "recharts";

```

```

function StudentProgress() {

```

```

const [progressData, setProgressData] = useState([]);
const [loading, setLoading] = useState(false);
const [error, setError] = useState("");
const token = localStorage.getItem("token");

useEffect(() => {
  const fetchProgress = async () => {
    try {
      setLoading(true);
      const res = await api.get("/instructor/students-progress", {
        headers: { Authorization: `Bearer ${token}` },
      });
      // ✅ Filter only active or completed enrollments
      const filteredData = (res.data.progress || []).filter(
        (p) => p.status === "active" || p.status === "completed"
      );
      setProgressData(filteredData);
    } catch (err) {
      console.error(err);
      setError("Failed to fetch student progress");
    } finally {
      setLoading(false);
    }
  };
  fetchProgress();
}, [token]);

// ----- Chart Data -----
const totalStudents = progressData.length;
const completed = progressData.filter((p) => p.progress >= 100).length;
const inProgress = progressData.filter(
  (p) => p.progress > 0 && p.progress < 100
).length;
const notStarted = totalStudents - completed - inProgress;

const statusData = [
  { name: "Completed", value: completed },
  { name: "In Progress", value: inProgress },
  { name: "Not Started", value: notStarted },
];

const COLORS = ["#28a745", "#ffc107", "#dc3545"];

```

```

// ----- Average progress per course (for Bar Chart) -----
const courseProgress = Object.values(
  progressData.reduce((acc, p) => {
    if (!acc[p.course?.title])
      acc[p.course?.title] = { course: p.course?.title, total: 0, count: 0 };
    acc[p.course?.title].total += p.progress;
    acc[p.course?.title].count += 1;
    return acc;
  }, {})
).map((c) => ({
  course: c.course,
  progress: c.count ? Math.round(c.total / c.count) : 0,
}));


return (
  <div style={{ padding: "20px", background: "#f9f9ff", minHeight: "100vh" }}>
    <h2 style={{ marginBottom: "20px" }}>
      Student Progress
    </h2>

    {error && <p style={{ color: "red" }}>{error}</p>

    {loading ? (
      <div style={{ display: "flex", flexDirection: "column", justifyContent: "center", alignItems: "center", height: "70vh", color: "#6f42c1", }}>
        <div style={{ border: "4px solid #f3f3f3", borderTop: "4px solid #6f42c1", borderRadius: "50%", width: "50px", height: "50px", animation: "spin 1s linear infinite", marginBottom: "20px", }} />
        <p>Loading...</p>
        <style>`@keyframes spin { 0% { transform: rotate(0deg); } 100% { transform: rotate(360deg); } }`</style>
      </div>
    ) : progressData.length === 0 ? (
      <p>No progress data available.</p>
    ) : (
      <>
        {/* ===== Charts Section ===== */}
        <div
          style={{
            display: "flex",
            flexWrap: "wrap",
            gap: "30px",
            justifyContent: "space-around",
          }
        >

```

```

        marginBottom: "40px",
    )}
>
 {/* Pie Chart */}
<div
    style={{
        background: "#fff",
        padding: "20px",
        borderRadius: "10px",
        boxShadow: "0 2px 8px rgba(0,0,0,0.1)",
        flex: "1",
        minWidth: "300px",
        textAlign: "center",
    }}
>
<h4 style={{ color: "#6f42c1", marginBottom: "15px" }}>
    Course Completion Overview
</h4>
<ResponsiveContainer width="100%" height={250}>
    <PieChart>
        <Pie
            data={statusData}
            dataKey="value"
            nameKey="name"
            outerRadius={90}
            label
        >
            {statusData.map((entry, index) => (
                <Cell
                    key={`cell-${index}`}
                    fill={COLORS[index % COLORS.length]}
                />
            ))}
        </Pie>
        <Legend />
        <Tooltip />
    </PieChart>
</ResponsiveContainer>
</div>

 {/* Bar Chart */}
<div
    style={{
        background: "#fff",

```

```

padding: "20px",
borderRadius: "10px",
boxShadow: "0 2px 8px rgba(0,0,0,0.1)",
flex: "2",
minWidth: "400px",
}}
>
<h4 style={{ color: "#6f42c1", marginBottom: "15px" }}>
  Average Progress per Course
</h4>
<ResponsiveContainer width="100%" height={250}>
  <BarChart data={courseProgress}>
    <XAxis dataKey="course" />
    <YAxis />
    <Tooltip />
    <Legend />
    <Bar dataKey="progress" fill="#6f42c1" />
  </BarChart>
</ResponsiveContainer>
</div>
</div>

/* ===== Table Section ===== */
<table
  style={{
    width: "100%",
    borderCollapse: "collapse",
    background: "#fff",
    borderRadius: "10px",
    overflow: "hidden",
    boxShadow: "0 2px 8px rgba(0,0,0,0.1)",
  }}
>
<thead style={{ background: "#6f42c1", color: "#fff" }}>
  <tr>
    <th style={{ padding: "10px" }}>Student</th>
    <th style={{ padding: "10px" }}>Course</th>
    <th style={{ padding: "10px" }}>Lessons Completed</th>
    <th style={{ padding: "10px" }}>Total Lessons</th>
    <th style={{ padding: "10px" }}>Exams Completed</th>
    <th style={{ padding: "10px" }}>Total Exams</th>
    <th style={{ padding: "10px" }}>Progress (%)</th>
  </tr>
</thead>

```

```

<tbody>
  {progressData.map((p) => (
    <tr key={p._id} style={{ borderBottom: "1px solid #eee", textAlign: "center" }}>
      <td style={{ padding: "8px" }}>p.student?.name</td>
      <td style={{ padding: "8px" }}>p.course?.title</td>
      <td style={{ padding: "8px" }}>p.completedLessons</td>
      <td style={{ padding: "8px" }}>p.totalLessons</td>
      <td style={{ padding: "8px" }}>p.completedExams</td>
      <td style={{ padding: "8px" }}>p.totalExams</td>
      <td
        style={{
          padding: "8px",
          fontWeight: "bold",
          color:
            p.progress >= 100
              ? "green"
              : p.progress >= 50
                ? "#ffc107"
                : "red",
        }}
      >
        {Math.round(p.progress)}%
      </td>
    </tr>
  )))
</tbody>

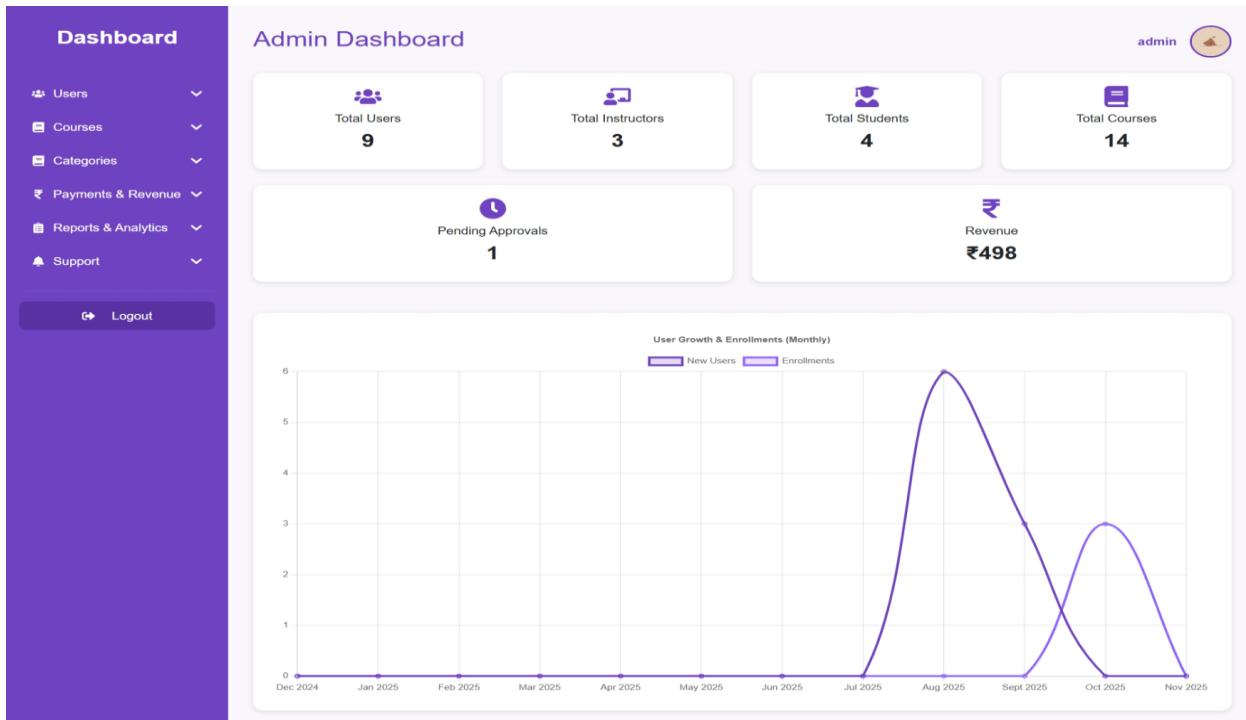
</table>
</>
)}
</div>
);
}
}

export default StudentProgress;

```

ADMIN SIDE

Admin dashboard



adminDashboard.jsx

```
import React, { useState, useEffect } from "react";
import {
  FaUsers,
  FaBook,
  FaChalkboardTeacher,
  FaUserGraduate,
  FaClock,
  FaRupeeSign,
} from "react-icons/fa";
import { Line } from "react-chartjs-2";
import { useLocation, Outlet, useNavigate } from "react-router-dom";
import api from "../..../api/api";
import DashboardLayout, { adminSidebarLinks } from "../dashboardLayout";

import {
  Chart as ChartJS,
```

```

CategoryScale,
LinearScale,
PointElement,
LineElement,
Title,
Tooltip,
Legend,
} from "chart.js";

ChartJS.register(
CategoryScale,
LinearScale,
PointElement,
LineElement,
Title,
Tooltip,
Legend
);

export default function AdminDashboard() {
  const location = useLocation();
  const navigate = useNavigate();

  const [user, setUser] = useState(null);
  const [stats, setStats] = useState([]);
  const [chartData, setChartData] = useState({});
  const [loading, setLoading] = useState(true);
  const [profileImage, setProfileImage] = useState(null);

  const BASE_URL = import.meta.env.VITE_BASE_URL || "";

  useEffect(() => {
    async function fetchDashboardData() {
      try {
        const res = await api.get("/admin/dashboard");
        const { stats: statsData, chartData: chartDataFromApi, user } = res.data;

        setUser(user);
        // Profile image
        const fullUrl = user.profilePic?.startsWith("http")
          ? user.profilePic
          : `${BASE_URL.replace(/\V$/, "")}${user.profilePic}`;
        setProfileImage(fullUrl || `${BASE_URL}/uploads/default.png`);
      }
    }
  }, []);
}

```

```

// Stats cards
setStats([
{
  title: "Total Users",
  value: statsData?.totalUsers ?? 0,
  icon: <FaUsers size={33} color="#6f42c1" />,
},
{
  title: "Total Instructors",
  value: statsData?.totalInstructors ?? 0,
  icon: <FaChalkboardTeacher size={33} color="#6f42c1" />,
},
{
  title: "Total Students",
  value: statsData?.totalStudents ?? 0,
  icon: <FaUserGraduate size={33} color="#6f42c1" />,
},
{
  title: "Total Courses",
  value: statsData?.totalCourses ?? 0,
  icon: <FaBook size={33} color="#6f42c1" />,
},
{
  title: "Pending Approvals",
  value: statsData?.pendingApprovals ?? 0,
  icon: <FaClock size={33} color="#6f42c1" />,
},
{
  title: "Revenue",
  value: `₹${Number(statsData?.revenue ?? 0).toLocaleString("en-IN")}` // formatted
properly
  icon: <FaRupeeSign size={33} color="#6f42c1" />,
},
]);

// Prepare chart data with 12 months
const monthsLabels = [];
const now = new Date();
for (let i = 11; i >= 0; i--) {
  const d = new Date(now.getFullYear(), now.getMonth() - i, 1);
  monthsLabels.push(d.toLocaleString("default", { month: "short", year: "numeric" }));
}

```

```

const usersMap = chartDataFromApi?.monthlyUsers?.reduce((acc, curr) => {
  acc[curr._id] = curr.count;
  return acc;
}, {}) || {};

const enrollMap = chartDataFromApi?.monthlyEnrollments?.reduce((acc, curr) => {
  acc[curr._id] = curr.count;
  return acc;
}, {}) || {};

const userData = monthsLabels.map(_ , idx) => usersMap[idx + 1] || 0;
const enrollmentData = monthsLabels.map(_ , idx) => enrollMap[idx + 1] || 0;

setChartData({
  labels: monthsLabels,
  datasets: [
    {
      label: "New Users",
      data: userData,
      borderColor: "#6f42c1",
      backgroundColor: "rgba(111,66,193,0.2)",
      tension: 0.4,
    },
    {
      label: "Enrollments",
      data: enrollmentData,
      borderColor: "#8f63ff",
      backgroundColor: "rgba(143,99,255,0.2)",
      tension: 0.4,
    },
  ],
});

 setLoading(false);
} catch (err) {
  console.error("Error fetching dashboard data:", err);
  setProfileImage(` ${BASE_URL}/uploads/default.png`);
  setLoading(false);
}
}

fetchDashboardData();
}, [BASE_URL]);

```

```

const chartOptions = {
  responsive: true,
  plugins: {
    legend: { position: "top" },
    title: { display: true, text: "User Growth & Enrollments (Monthly)" },
  },
  scales: { y: { beginAtZero: true } },
};

const styles = {
  profileBorder: {
    width: "50px",
    height: "50px",
    borderRadius: "50%",
    objectFit: "cover",
    border: "3px solid #6f42c1",
    cursor: "pointer",
  },
};

if (loading) {
  return (
    <DashboardLayout sidebarLinks={adminSidebarLinks}>
      <div
        style={{
          display: "flex",
          flexDirection: "column",
          justifyContent: "center",
          alignItems: "center",
          height: "70vh",
          color: "#6f42c1",
        }}
      >
        <div
          style={{
            border: "4px solid #f3f3f3",
            borderTop: "4px solid #6f42c1",
            borderRadius: "50%",
            width: "50px",
            height: "50px",
            animation: "spin 1s linear infinite",
            marginBottom: "20px",
          }}
        >

```

```

        />
      <p>Loading your data...</p>
      <style>{@keyframes spin { 0% { transform: rotate(0deg); } 100% { transform: rotate(360deg); } }}</style>
    </div>
  </DashboardLayout>
);
}

return (
<DashboardLayout sidebarLinks={adminSidebarLinks}>
{location.pathname === "/admin-dashboard" ? (
<>
  {/* Header */}
  <div className="d-flex justify-content-between align-items-center mb-4">
    <h2 style={{ color: "#6f42c1" }}>Admin Dashboard</h2>
    <div className="d-flex align-items-center gap-3">
      <span style={{ fontWeight: "bold", color: "#6f42c1" }}>
        {user?.name || "Admin"}
      </span>
      <img
        src={profileImage || `${BASE_URL}/uploads/default.png`}
        alt="Admin Profile"
        style={styles.profileBorder}
        onClick={() => navigate("/admin-dashboard/profile")}
        title="View Profile"
      />
    </div>
  </div>
</div>
)
  {/* Stats Cards */}
  <div className="d-flex flex-wrap gap-4 mb-5">
    {stats.map((stat, i) => (
      <div
        key={i}
        style={{
          backgroundColor: "#fff",
          padding: "20px",
          borderRadius: "12px",
          textAlign: "center",
          flex: "1 1 220px",
          boxShadow: "0 2px 8px rgba(0,0,0,0.1)",
        }}
      >

```

```
{stat.icon}
<h6 className="mt-2">{stat.title}</h6>
<h3 className="fw-bold">{stat.value}</h3>
</div>
))}
</div>

/* Chart */
{chartData?.datasets && (
<div
style={{
  backgroundColor: "#fff",
  padding: "25px",
  borderRadius: "12px",
  boxShadow: "0 2px 8px rgba(0,0,0,0.1)",
}}
>
  <Line data={chartData} options={chartOptions} />
</div>
)}
</>
):(
<Outlet />
)}
</DashboardLayout>
);
}
```

All Users Lists

The screenshot shows a dashboard titled "All Users". On the left is a sidebar with a purple header "Dashboard" containing links for "Users", "Courses", "Categories", "Payments & Revenue", "Reports & Analytics", and "Support", along with a "Logout" button. The main area is titled "All Users" and features a search bar with dropdowns for "All Roles" and "Search by name or email". Below the search bar is a grid of user profiles. Each profile card contains a user icon, the user's name, their email, role, and joining date, followed by a "View Details" button.

User	Email	Role	Joined
admin1	admin1@gmail.com	admin	23/10/2025
student4	student4@gmail.com	student	15/10/2025
instructor3	instructor3@gmail.com	instructor	9/10/2025
instructor2	instructor2@gmail.com	instructor	19/9/2025
student3	student3@gmail.com	student	19/9/2025
student2	student2@gmail.com	student	19/9/2025
abc	instructor1@gmail.com	instructor	17/9/2025
student1	hmsonline3111@gmail.com	student	17/9/2025
admin	admin@gmail.com	admin	16/9/2025

allUsers.jsx

```
import React, { useEffect, useState } from "react";
import api from "../../api/api";
import { Button, Form, Row, Col, Modal, Spinner } from "react-bootstrap";

export default function AllUsers() {
  const [users, setUsers] = useState([]);
  const [filteredUsers, setFilteredUsers] = useState([]);
  const [roleFilter, setRoleFilter] = useState("all");
  const [searchTerm, setSearchTerm] = useState("");

  const [showDetailsModal, setShowDetailsModal] = useState(false);
  const [selectedUser, setSelectedUser] = useState(null);
  const [userDetails, setUserDetails] = useState(null);
  const [loadingDetails, setLoadingDetails] = useState(false);

  const [showAddAdminModal, setShowAddAdminModal] = useState(false);
  const [newAdmin, setNewAdmin] = useState({ name: "", email: "", password: "", profilePic: null });

  useEffect(() => {
    const fetchUsers = async () => {
      try {
        const response = await api.get("/users");
        setUsers(response.data);
      } catch (error) {
        console.error(error);
      }
    };
    fetchUsers();
  }, [roleFilter, searchTerm]);

  const handleRoleFilterChange = (e) => {
    setRoleFilter(e.target.value);
  };

  const handleSearchTermChange = (e) => {
    setSearchTerm(e.target.value);
  };

  const handleShowDetailsModal = (user) => {
    setSelectedUser(user);
    setShowDetailsModal(true);
  };

  const handleShowAddAdminModal = (user) => {
    setNewAdmin({ ...user });
    setShowAddAdminModal(true);
  };

  const handleUserDetails = (details) => {
    setUserDetails(details);
    setLoadingDetails(true);
  };

  const handleUserDetailsSuccess = (details) => {
    setLoadingDetails(false);
    setShowDetailsModal(false);
  };

  const handleUserDetailsError = (error) => {
    console.error(error);
    setShowDetailsModal(false);
  };

  const handleAddAdminSuccess = (user) => {
    setUsers([user, ...users]);
    setShowAddAdminModal(false);
  };

  const handleAddAdminError = (error) => {
    console.error(error);
    setShowAddAdminModal(false);
  };

  return (
    <div>
      <Form>
        <Row>
          <Col>
            <Form.Select value={roleFilter}>
              <option value="all">All Roles</option>
              <option value="admin">Admin</option>
              <option value="student">Student</option>
              <option value="instructor">Instructor</option>
            </Form.Select>
          </Col>
          <Col>
            <Form.Control type="text" value={searchTerm} onChange={handleSearchTermChange}/>
          </Col>
        </Row>
      </Form>
      <table border="1">
        <thead>
          <tr>
            <th>User</th>
            <th>Email</th>
            <th>Role</th>
            <th>Joined</th>
          </tr>
        </thead>
        <tbody>
          {users.map((user) => (
            <tr>
              <td>{user.name}</td>
              <td>{user.email}</td>
              <td>{user.role}</td>
              <td>{user.joined}</td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
}
```

```

const [addingAdmin, setAddingAdmin] = useState(false);

useEffect(() => {
  fetchUsers();
}, []);

const fetchUsers = async () => {
  try {
    const res = await api.get("/admin/users");
    setUsers(res.data);
    setFilteredUsers(res.data);
  } catch (err) {
    console.error(err);
  }
};

useEffect(() => {
  let tempUsers = [...users];
  if (roleFilter !== "all") tempUsers = tempUsers.filter(u => u.role === roleFilter);
  if (searchTerm) {
    const term = searchTerm.toLowerCase();
    tempUsers = tempUsers.filter(u => u.name.toLowerCase().includes(term) || u.email.toLowerCase().includes(term));
  }
  setFilteredUsers(tempUsers);
}, [roleFilter, searchTerm, users]);

const handleViewDetails = async (user) => {
  setSelectedUser(user);
  setLoadingDetails(true);
  setShowDetailsModal(true);

  try {
    let res;
    if (user.role === "student") res = await api.get(`/admin/students/${user._id}`);
    else if (user.role === "instructor") res = await api.get(`/admin/instructors/${user._id}`);
    else res = { data: {} };
    setUserDetails(res.data);
  } catch (err) {
    console.error(err);
    setUserDetails(null);
  } finally {
    setLoadingDetails(false);
  }
}

```

```

};

const handleAddAdmin = async (e) => {
  e.preventDefault();
  setAddingAdmin(true);
  try {
    await api.post("/addAdmin", {
      name: newAdmin.name,
      email: newAdmin.email,
      password: newAdmin.password
    });
  }

  setShowAddAdminModal(false);
  setNewAdmin({ name: "", email: "", password: "" });
  fetchUsers();
} catch (err) {
  console.error("Error adding admin:", err);
  alert(err.response?.data?.warning || "Failed to add admin");
} finally {
  setAddingAdmin(false);
}
};

const styles = {
  container: { padding: "2rem", fontFamily: "'Segoe UI', sans-serif" },
  heading: { color: "#6f42c1", fontWeight: 700, marginBottom: "1.5rem", fontSize: "1.8rem" },
  filters: { marginBottom: "1.5rem" },
  filterInput: { border: "2px solid #6f42c1", borderRadius: "8px", padding: "0.5rem", fontWeight: 500, color: "#6f42c1" },
  cardsContainer: { display: "grid", gridTemplateColumns: "repeat(auto-fill, minmax(250px, 1fr))", gap: "1rem" },
  card: { background: "#fff", padding: "1rem", borderRadius: "12px", boxShadow: "0 4px 15px rgba(0,0,0,0.08)", transition: "0.3s", cursor: "pointer" },
  cardHover: { transform: "translateY(-5px)", boxShadow: "0 6px 25px rgba(0,0,0,0.15)" },
  cardTitle: { fontWeight: 600, fontSize: "1.1rem", color: "#6f42c1", marginBottom: "0.3rem" },
  cardText: { fontSize: "0.9rem", color: "#333", marginBottom: "0.2rem" },
  viewBtn: { marginTop: "0.5rem", backgroundColor: "#6f42c1", border: "none", fontWeight: 500 },
  modalHeader: { backgroundColor: "#6f42c1", color: "#fff", fontWeight: 600, fontSize: "1.2rem" },
  modalBody: { fontSize: "0.95rem", lineHeight: "1.6", color: "#333" },
  spinner: { display: "flex", justifyContent: "center", padding: "2rem" },
}

```

```

label: { fontWeight: 600, color: "#6f42c1", marginRight: "5px" },
addAdminCard: {
  display: "flex", justifyContent: "center", alignItems: "center",
  fontSize: "2rem", color: "#6f42c1", fontWeight: "700",
  cursor: "pointer", border: "2px dashed #6f42c1", background: "#faf5ff", borderRadius:
"12px"
}
};

return (
  <div style={styles.container}>
    <h2 style={styles.heading}>All Users</h2>

    {/* Filters */}
    <Row style={styles.filters}>
      <Col md={4}>
        <Form.Select value={roleFilter} onChange={(e) => setRoleFilter(e.target.value)} style={styles.filterInput}>
          <option value="all">All Roles</option>
          <option value="admin">Admin</option>
          <option value="instructor">Instructor</option>
          <option value="student">Student</option>
        </Form.Select>
      </Col>
      <Col md={4}>
        <Form.Control type="text" placeholder="Search by name or email" value={searchTerm} onChange={(e) => setSearchTerm(e.target.value)} style={styles.filterInput}>
      </Col>
    </Row>

    {/* User Cards */}
    <div style={styles.cardsContainer}>
      {filteredUsers.length ? filteredUsers.map((u) => (
        <div
          key={u._id}
          style={styles.card}
          onMouseEnter={(e) => Object.assign(e.currentTarget.style, styles.cardHover)}
          onMouseLeave={(e) => Object.assign(e.currentTarget.style, styles.card)}
        >
          <img
            src={`${import.meta.env.VITE_BASE_URL || ""}${u.profilePic}`}
            alt={u.name}

```

```

        style={{ width: "60px", height: "60px", borderRadius: "50%", marginBottom: "0.5rem" }}
      />
      <div style={styles.cardTitle}>{u.name}</div>
      <div style={styles.cardText}><span style={styles.label}>Email:</span>
{u.email}</div>
      <div style={styles.cardText}><span style={styles.label}>Role:</span> {u.role}</div>
      <div style={styles.cardText}><span style={styles.label}>Joined:</span> {new Date(u.createdAt).toLocaleDateString()}</div>
      <Button style={styles.viewBtn} onClick={() => handleViewDetails(u)}>View Details</Button>
    </div>
  )) : <p style={{ textAlign: "center", color: "#6f42c1" }}>No users found.</p>

  {/* Add New Admin Card - only if role filter is "admin" */}
  {roleFilter === "admin" && (
    <div style={styles.addAdminCard} onClick={() => setShowAddAdminModal(true)}>
      + Add New Admin
    </div>
  )}
</div>

 {/* User Details Modal */}
<Modal show={showDetailsModal} onHide={() => setShowDetailsModal(false)} centered size="lg">
  <Modal.Header closeButton style={styles.modalHeader}>
    {selectedUser?.name || "User Details"} - {selectedUser?.role}
  </Modal.Header>
  <Modal.Body style={styles.modalBody}>
    {loadingDetails ? (
      <div style={styles.spinner}><Spinner animation="border" style={{ color: "#6f42c1" }} />
    ) : userDetails ? (
      <div>
        {selectedUser.profilePic && (
          <img
            src={`${import.meta.env.VITE_BASE_URL || ""}${selectedUser.profilePic}`}
            alt={selectedUser.name}
            style={{ width: "80px", height: "80px", borderRadius: "50%", marginBottom: "1rem" }}
          />
        )}
      </div>
    )} : <p><span style={styles.label}>Name:</span> {selectedUser.name}</p>

```

```

<p><span style={styles.label}>Email:</span> {selectedUser.email}</p>

{selectedUser.role === "student" && (
  <>
    <p><span style={styles.label}>Education:</span> {userDetails.education || "—"}</p>
    <p><span style={styles.label}>Enrolled Courses:</span> {(userDetails.enrolledCourses || []).length}</p>
    <p><span style={styles.label}>Interests:</span> {(userDetails.interests || []).join(", ")} || "—"</p>
  </>
)
}

{selectedUser.role === "instructor" && (
  <>
    <p><span style={styles.label}>Bio:</span> {userDetails.bio || "—"}</p>
    <p><span style={styles.label}>Expertise:</span> {(userDetails.expertise || []).join(", ")} || "—"</p>
    <p><span style={styles.label}>Qualifications:</span> {(userDetails.qualifications || []).join(", ")} || "—"</p>
    <p><span style={styles.label}>Experience:</span> {userDetails.experience ? `${userDetails.experience} yrs` : "—"}</p>
  </>
)
}
</div>
) : <p style={{ color: "red", textAlign: "center" }}>No details found.</p>
</Modal.Body>
</Modal>

/* Add Admin Modal */
<Modal show={showAddAdminModal} onHide={() => setShowAddAdminModal(false)} centered>
  <Modal.Header closeButton style={styles.modalHeader}>Add New Admin</Modal.Header>
  <Modal.Body>
    <Form onSubmit={handleAddAdmin}>
      <Form.Group className="mb-3">
        <Form.Label>Name</Form.Label>
        <Form.Control
          type="text"
          value={newAdmin.name}
          onChange={(e) => setNewAdmin({ ...newAdmin, name: e.target.value })}
          required
        />
    
```

```

        </Form.Group>

        <Form.Group className="mb-3">
            <Form.Label>Email</Form.Label>
            <Form.Control
                type="email"
                value={newAdmin.email}
                onChange={(e) => setNewAdmin({ ...newAdmin, email: e.target.value })}
                required
            />
        </Form.Group>

        <Form.Group className="mb-3">
            <Form.Label>Password</Form.Label>
            <Form.Control
                type="password"
                value={newAdmin.password}
                onChange={(e) => setNewAdmin({ ...newAdmin, password: e.target.value })}
                required
            />
        </Form.Group>

        <Button
            type="submit"
            style={{ backgroundColor: "#6f42c1", border: "none" }}
            disabled={addingAdmin}
        >
            {addingAdmin ? "Adding..." : "Add Admin"}
        </Button>
    </Form>

    </Modal.Body>
</Modal>
</div>
);
}

```

All Course Lists

Dashboard

- [Users](#)
- [Courses](#)
- [Categories](#)
- [Payments & Revenue](#)
- [Reports & Analytics](#)
- [Support](#)

[Logout](#)

All Courses

All Status
All Categories

XYZ

Instructor: abc

Category: Python

Level: Intermediate

Status: Rejected

Price: ₹99

► Lessons (0)

► Exams (0)

16/10/2025

abc

Instructor: abc

Category: Next JS

Level: Intermediate

Status: Pending Approval

Price: ₹99

► Lessons (0)

► Exams (0)

14/10/2025

FRONT END DEVELOPMENT WITH REACT

Instructor: abc

Category: Frontend Development

Level: Intermediate

Status: Approved

Price: ₹149

► Lessons (0)

► Exams (0)

09/10/2025

DATA SCIENCE USING PYTHON

Instructor: instructor2

Category: Python

Level: Intermediate

Status: Approved

Price: ₹249

► Lessons (0)

► Exams (0)

09/10/2025

MERN STACK DEVELOPMENT BOOTCAMP

Instructor: instructor3

Category: MERN Stack Development

Level: Advanced

Status: Approved

Price: ₹499

► Lessons (0)

► Exams (0)

09/10/2025

Backend Development with Node.js

Instructor: abc

Category: Backend Development

Level: Intermediate

Status: Draft

Price: ₹290

► Lessons (1)

► Exams (1)

09/10/2025

Advanced Python for Data Science

Instructor: instructor3

Category: Python

Level: Beginner

Status: Approved

Price: ₹399

► Lessons (0)

► Exams (0)

09/10/2025

JavaScript Basics

Instructor: instructor3

Category: Javascript

Level: Beginner

Status: Approved

Price: ₹99

► Lessons (0)

► Exams (0)

09/10/2025

Fullstack Development Project

Instructor: instructor2

Category: Fullstack Development

Level: Advanced

Status: Approved

Price: ₹399

► Lessons (0)

► Exams (0)

30/09/2025

Frontend Development Bootcamp

Instructor: instructor2

Category: Frontend Development

Level: Beginner

Status: Approved

Price: ₹149

► Lessons (0)

► Exams (0)

30/09/2025

Data Science with Python

Instructor: instructor2

Category: Data Science

Level: Intermediate

Status: Approved

Price: ₹299

► Lessons (0)

► Exams (0)

30/09/2025

Next JS Fundamentals

Instructor: abc

Category: Next JS

Level: Beginner

Status: Approved

Price: ₹199

► Lessons (4)

► Exams (3)

30/09/2025

5 Best MERN Projects

To Add in Recipe

Advanced MERN Project

Instructor: abc

Category: MERN Stack Development

Level: Intermediate

Status: Approved

Price: ₹299

► Lessons (3)

► Exams (2)

30/09/2025

Mastering Next JS: Part 1

Instructor: abc

Category: Next JS

Level: Intermediate

Status: Approved

Price: ₹0

► Lessons (3)

► Exams (0)

25/09/2025

LMS Platform

A complete online learning system for students and instructors. Learn, teach, and grow your skills with ease.

Quick Links

Courses
About Us
Contact Us

Resources

Privacy Policy
Terms & Conditions
Support

Follow Us

© 2025 LMS Platform. All Rights Reserved.

allCourses.jsx

Page | 240

```

import React, { useEffect, useState } from "react";
import api from "../../api/api";

export default function AllCourses() {
  const [courses, setCourses] = useState([]);
  const [filtered, setFiltered] = useState([]);
  const [search, setSearch] = useState("");
  const [statusFilter, setStatusFilter] = useState("all");
  const [categoryFilter, setCategoryFilter] = useState("all");

  useEffect(() => {
    api
      .get("/admin/courses")
      .then((res) => {
        setCourses(res.data);
        setFiltered(res.data);
      })
      .catch(console.error);
  }, []);

  // Handle filtering logic
  useEffect(() => {
    let filteredList = courses.filter((c) => {
      const matchesSearch =
        c.title.toLowerCase().includes(search.toLowerCase()) ||
        c.instructor?.name?.toLowerCase().includes(search.toLowerCase());

      const matchesStatus =
        statusFilter === "all" || c.status === statusFilter;

      const matchesCategory =
        categoryFilter === "all" ||
        c.category?.name === categoryFilter ||
        c.category === categoryFilter;

      return matchesSearch && matchesStatus && matchesCategory;
    });

    setFiltered(filteredList);
  }, [search, statusFilter, categoryFilter, courses]);

  // Unique categories
  const categories = [
    ...new Set(courses.map((c) => c.category?.name).filter(Boolean)),
  ];
}

```

```
];

return (
  <div
    style={{
      padding: "20px",
      backgroundColor: "#f8f9fc",
      minHeight: "100vh",
      fontFamily: "'Segoe UI', sans-serif",
    }}
  >
  <h2
    style={{
      color: "#6f42c1",
      textAlign: "center",
      marginBottom: "20px",
      fontWeight: "700",
    }}
  >
    All Courses
  </h2>

/* 🔎 Filters Row */
<div
  style={{
    display: "flex",
    justifyContent: "center",
    gap: "10px",
    flexWrap: "wrap",
    marginBottom: "20px",
  }}
>
/* Search Bar */
<input
  type="text"
  placeholder="Search by title or instructor..."
  value={search}
  onChange={(e) => setSearch(e.target.value)}
  style={{
    padding: "8px 12px",
    borderRadius: "6px",
    border: "1px solid #ccc",
    width: "240px",
    outline: "none",
  }}
/>
```

```

        }
    />

/* Status Filter */
<select
    value={statusFilter}
    onChange={(e) => setStatusFilter(e.target.value)}
    style={{
        padding: "8px 12px",
        borderRadius: "6px",
        border: "1px solid #ccc",
        outline: "none",
    }}
>
    <option value="all">All Status</option>
    <option value="approved">Approved</option>
    <option value="pendingApproval">Pending</option>
    <option value="rejected">Rejected</option>
    <option value="draft">Draft</option>
</select>

/* Category Filter */
<select
    value={categoryFilter}
    onChange={(e) => setCategoryFilter(e.target.value)}
    style={{
        padding: "8px 12px",
        borderRadius: "6px",
        border: "1px solid #ccc",
        outline: "none",
    }}
>
    <option value="all">All Categories</option>
    {categories.map((cat) => (
        <option key={cat} value={cat}>
            {cat}
        </option>
    )))
</select>
</div>

/* Courses Grid */
<div
    style={{

```

```

        display: "grid",
        gridTemplateColumns: "repeat(auto-fill, minmax(260px, 1fr))",
        gap: "16px",
        justifyItems: "center",
    )}
>
{filtered.map((course) => (
    <div
        key={course._id}
        style={{
            width: "100%",
            maxWidth: "280px",
            backgroundColor: "#fff",
            borderRadius: "12px",
            boxShadow: "0 3px 8px rgba(0,0,0,0.08)",
            overflow: "hidden",
            transition: "all 0.2s ease-in-out",
        }}
        onMouseEnter={(e) => {
            e.currentTarget.style.transform = "translateY(-4px)";
            e.currentTarget.style.boxShadow =
                "0 6px 14px rgba(0,0,0,0.15)";
        }}
        onMouseLeave={(e) => {
            e.currentTarget.style.transform = "translateY(0)";
            e.currentTarget.style.boxShadow =
                "0 3px 8px rgba(0,0,0,0.08)";
        }}
    >
        {/* Thumbnail */}
        {course.thumbnail ? (
            <img
                src={
                    course.thumbnail.startsWith("http")
                    ? course.thumbnail
                    : `${import.meta.env.VITE_BASE_URL}${course.thumbnail}`
                }
                alt={course.title}
                style={{
                    width: "100%",
                    height: "140px",
                    objectFit: "cover",
                    borderBottom: "1px solid #eee",
                }}
        )
    
```

```

        />
    ) : (
        <div
            style={{
                height: "140px",
                backgroundColor: "#f2f2f2",
                display: "flex",
                alignItems: "center",
                justifyContent: "center",
                color: "#999",
                fontSize: "14px",
                borderBottom: "1px solid #eee",
            }}
        >
            No Thumbnail
        </div>
    )}
}

/* Body */
<div style={{ padding: "12px 14px" }}>
    <h5
        style={{
            color: "#6f42c1",
            fontWeight: "600",
            fontSize: "16px",
            marginBottom: "6px",
        }}
    >
        {course.title}
    </h5>
    <p style={{ margin: "2px 0", fontSize: "13px", color: "#555" }}>
        <strong>Instructor:</strong> {course.instructor?.name || "N/A"}
    </p>
    <p style={{ margin: "2px 0", fontSize: "13px", color: "#555" }}>
        <strong>Category:</strong> {course.category?.name || "N/A"}
    </p>
    <p style={{ margin: "2px 0", fontSize: "13px", color: "#555" }}>
        <strong>Level:</strong> {course.level}
    </p>
    <p style={{ margin: "2px 0", fontSize: "13px", color: "#555" }}>
        <strong>Status:</strong>{" "}
    <span
        style={{
            backgroundColor:

```

```

course.status === "approved"
? "#28a745"
: course.status === "pendingApproval"
? "#ffc107"
: course.status === "rejected"
? "#dc3545"
: "#6c757d",
color: "#fff",
padding: "1px 6px",
borderRadius: "6px",
fontSize: "11px",
textTransform: "capitalize",
}}
>
{course.status}
</span>
</p>
<p
style={{ margin: "2px 0 8px", fontSize: "13px", color: "#555" }}>
>
<strong>Price:</strong> ₹{course.price}
</p>

/* Lessons */
<details style={{ marginBottom: "4px", fontSize: "13px" }}>
<summary
style={{{
cursor: "pointer",
color: "#6f42c1",
fontWeight: "600",
fontSize: "13px",
}}}
>
Lessons ({course.lessons?.length || 0})
</summary>
<ul
style={{{
marginTop: "6px",
paddingLeft: "16px",
maxHeight: "70px",
overflowY: "auto",
}}}
>
{course.lessons?.map((lesson) => (

```

```

<li key={lesson._id} style={{ padding: "2px 0" }}>
  {lesson.title} ({lesson.contentType})
</li>
)}
</ul>
</details>

/* Exams */
<details style={{ fontSize: "13px" }}>
  <summary
    style={{{
      cursor: "pointer",
      color: "#6f42c1",
      fontWeight: "600",
      fontSize: "13px",
    }}}
  >
    Exams ({course.exams?.length || 0})
  </summary>
  <ul
    style={{{
      marginTop: "6px",
      paddingLeft: "16px",
      maxHeight: "70px",
      overflowY: "auto",
    }}}
  >
    {course.exams?.map((exam) => (
      <li key={exam._id} style={{ padding: "2px 0" }}>
        {exam.title} ({exam.duration} min)
      </li>
    ))}
  </ul>
</details>
</div>

/* Footer */
<div
  style={{{
    backgroundColor: "#fafafa",
    borderTop: "1px solid #eee",
    padding: "8px 14px",
    fontSize: "12px",
    color: "#777",
  }}>
```

```

        textAlign: "right",
    )}
>
    {new Date(course.createdAt).toLocaleDateString("en-GB")}
    </div>
    </div>
)}
</div>
</div>
);
}

```

Pending Courses

The screenshot shows a dashboard interface with a sidebar on the left and a main content area on the right.

Dashboard Sidebar:

- Users
- Courses (selected)
- All Courses
- Pending Approval (highlighted)
- Rejected Courses
- Categories
- Payments & Revenue
- Reports & Analytics
- Support

Main Content Area:

Pending Courses

A course card is displayed:

- Title:** abc
- Instructor:** abc
- Category:** Next JS
- Level:** Intermediate
- Price:** 999
- Lessons:** (0)
- Exams:** (0)

Buttons at the bottom of the card: **Approve** (green) and **Reject** (red).

At the bottom of the main content area: 14/10/2025

pendingCourses.jsx

```

import React, { useEffect, useState } from "react";
import api from "../../api/api";
import { Button, Spinner } from "react-bootstrap";

export default function PendingCourses() {
    const [courses, setCourses] = useState([]);
    const [loading, setLoading] = useState(true);

    useEffect(() => {
        api.get("/admin/courses/pending")
            .then((res) => setCourses(res.data))
    })
}

```

```

    .catch(console.error)
    .finally(() => setLoading(false));
}, []);
}

const approve = async (id) => {
  try { await api.post(`admin/courses/${id}/approve`); setCourses(prev => prev.filter(c => c._id !== id)); }
  catch (err) { console.error(err); }
};

const reject = async (id) => {
  try { await api.post(`admin/courses/${id}/reject`); setCourses(prev => prev.filter(c => c._id !== id)); }
  catch (err) { console.error(err); }
};

const styles = {
  container: { padding: "1.5rem", backgroundColor: "#f8f9fc", minHeight: "100vh", fontFamily: "'Segoe UI', sans-serif" },
  heading: { color: "#6f42c1", fontWeight: 700, textAlign: "center", marginBottom: "1.5rem" },
  cardsContainer: { display: "grid", gridTemplateColumns: "repeat(auto-fill, minmax(260px, 1fr))", gap: "1rem" },
  card: { background: "#fff", borderRadius: "12px", boxShadow: "0 3px 10px rgba(0,0,0,0.08)", overflow: "hidden", cursor: "default", transition: "0.25s" },
  cardHover: { transform: "translateY(-3px)", boxShadow: "0 5px 15px rgba(0,0,0,0.12)" },
  thumb: { width: "100%", height: "120px", objectFit: "cover" },
  body: { padding: "10px" },
  title: { color: "#6f42c1", fontWeight: 600, fontSize: "1rem", marginBottom: "4px" },
  text: { fontSize: "0.85rem", color: "#333", marginBottom: "2px" },
  label: { fontWeight: 600, color: "#6f42c1", marginRight: "4px" },
  btn: { border: "none", fontWeight: 500, color: "#fff", width: "48%", padding: "5px 0", borderRadius: "6px", fontSize: "0.8rem" },
  footer: { backgroundColor: "#fafafa", borderTop: "1px solid #eee", padding: "6px 10px", fontSize: "11px", color: "#777", textAlign: "right" },
  details: { marginBottom: "6px", fontSize: "0.85rem" },
  summary: { cursor: "pointer", color: "#6f42c1", fontWeight: 600, fontSize: "0.85rem" },
  ul: { marginTop: "4px", paddingLeft: "14px", maxHeight: "50px", overflowY: "auto" },
};

return (
  <div style={styles.container}>
    <h2 style={styles.heading}>Pending Courses</h2>
    {loading ? (

```

```

    <div style={{ display: "flex", flexDirection: "column", justifyContent: "center", alignItems: "center", height: "70vh", color: "#6f42c1", }}>
      <div style={{ border: "4px solid #f3f3f3", borderTop: "4px solid #6f42c1", borderRadius: "50%", width: "50px", height: "50px", animation: "spin 1s linear infinite", marginBottom: "20px", }} />
      <p>Loading...</p>
      <style>{@keyframes spin { 0% { transform: rotate(0deg); } 100% { transform: rotate(360deg); } }}</style>
    </div>
  ) : courses.length === 0 ? (
    <p style={{ textAlign: "center", color: "#6f42c1" }}>🎉 No pending courses!</p>
  ) : (
    <div style={styles.cardsContainer}>
      {courses.map(c => (
        <div key={c._id} style={styles.card} onMouseEnter={e =>
          Object.assign(e.currentTarget.style, styles.cardHover) } onMouseLeave={e =>
          Object.assign(e.currentTarget.style, styles.card)}>
          {c.thumbnail ? (
            <img src={c.thumbnail.startsWith("http") ? c.thumbnail :
              `${import.meta.env.VITE_BASE_URL?.replace(/\$/,"")}/${c.thumbnail?.replace(/^\//,"")}`}
            alt={c.title} style={styles.thumb} />
          ) : (
            <div style={{ ...styles.thumb, backgroundColor: "#f2f2f2", display: "flex", alignItems: "center", justifyContent: "center", color: "#999", fontSize: "12px" }}>No Thumbnail</div>
          )}
        <div style={styles.body}>
          <div style={styles.title}>{c.title}</div>
          <div style={styles.text}><span
style={styles.label}>Instructor:</span>{c.instructor?.name || "N/A"}</div>
          <div style={styles.text}><span style={styles.label}>Category:</span>{c.category?.name
          || "N/A"}</div>
          <div style={styles.text}><span style={styles.label}>Level:</span>{c.level || "-"}</div>
          <div style={styles.text}><span style={styles.label}>Price:</span> ₹{c.price ?? 0}</div>
        <div style={styles.details}><summary style={styles.summary}>Lessons
          ({c.lessons?.length || 0})</summary>
          <ul style={styles.ul}>{c.lessons?.slice(0, 5).map(l => <li key={l._id}>{l.title}
          ({l.contentType})</li>)}</ul>
        </details>
        <div style={styles.details}><summary style={styles.summary}>Exams
          ({c.exams?.length || 0})</summary>
        </div>
      )}</div>
    )
  )

```

```
<ul style={styles.ul}>{c.exams?.slice(0, 5).map(e => <li key={e._id}>{e.title}<br/>{e.duration} min</li>)}/>
```

```
</ul>
```

```
</details>
```

```
<div style={{ display: "flex", justifyContent: "space-between", marginTop: "8px" }}>
```

```
    <Button style={{ ...styles.btn, backgroundColor: "#28a745" }} onClick={() => approve(c._id)}>Approve</Button>
```

```
    <Button style={{ ...styles.btn, backgroundColor: "#dc3545" }} onClick={() => reject(c._id)}>Reject</Button>
```

```
    </div>
```

```
</div>
```

```
<div style={styles.footer}>{new Date(c.createdAt).toLocaleDateString("en-GB")}</div>
```

```
</div>
```

```
)>
```

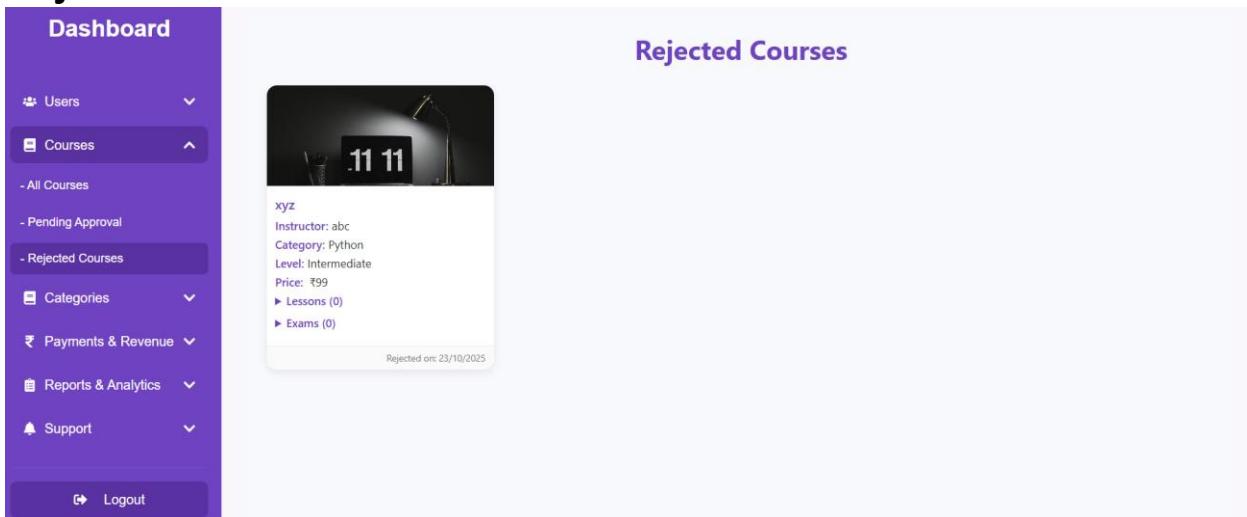
```
</div>
```

```
)>
```

```
</div>
```

```
);
```

Rejected Courses



rejectedCourses.jsx

```
import React, { useEffect, useState } from "react";
import api from "../../api/api";
```

```

export default function RejectedCourses() {
  const [courses, setCourses] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    api.get("/admin/courses/rejected")
      .then(res => setCourses(res.data))
      .catch(console.error)
      .finally(() => setLoading(false));
  }, []);

  const styles = {
    container: { padding: "1.5rem", backgroundColor: "#f8f9fc", minHeight: "100vh", fontFamily: "'Segoe UI', sans-serif" },
    heading: { color: "#6f42c1", fontWeight: 700, textAlign: "center", marginBottom: "1.5rem" },
    cardsContainer: { display: "grid", gridTemplateColumns: "repeat(auto-fill, minmax(260px, 1fr))", gap: "1rem" },
    card: { background: "#fff", borderRadius: "12px", boxShadow: "0 3px 10px rgba(0,0,0,0.08)", overflow: "hidden", cursor: "default", transition: "0.25s" },
    cardHover: { transform: "translateY(-3px)", boxShadow: "0 5px 15px rgba(0,0,0,0.12)" },
    thumb: { width: "100%", height: "120px", objectFit: "cover" },
    body: { padding: "10px" },
    title: { color: "#6f42c1", fontWeight: 600, fontSize: "1rem", marginBottom: "4px" },
    text: { fontSize: "0.85rem", color: "#333", marginBottom: "2px" },
    label: { fontWeight: 600, color: "#6f42c1", marginRight: "4px" },
    details: { marginBottom: "6px", fontSize: "0.85rem" },
    summary: { cursor: "pointer", color: "#6f42c1", fontWeight: 600, fontSize: "0.85rem" },
    ul: { marginTop: "4px", paddingLeft: "14px", maxHeight: "50px", overflowY: "auto" },
    footer: { backgroundColor: "#fafafa", borderTop: "1px solid #eee", padding: "6px 10px", fontSize: "11px", color: "#777", textAlign: "right" },
  };

  return (
    <div style={styles.container}>
      <h2 style={styles.heading}>Rejected Courses</h2>

      {loading ? (
        <div style={{ display: "flex", flexDirection: "column", justifyContent: "center", alignItems: "center", height: "70vh", color: "#6f42c1", }}>
          <div style={{ border: "4px solid #f3f3f3", borderTop: "4px solid #6f42c1", borderRadius: "50%", width: "50px", height: "50px", animation: "spin 1s linear infinite", marginBottom: "20px", }} />
          <p>Loading...</p>
        </div>
      ) : (
        <div style={{ display: "grid", gridTemplateColumns: "repeat(auto-fill, minmax(260px, 1fr))", gap: "1rem" }}>
          {courses.map(course => (
            <div style={{ position: "relative", height: "100%" }}>
              <img alt={course.name} style={{ width: "100%", height: "100%", objectFit: "cover" }} />
              <div style={{ position: "absolute", bottom: "0", left: "0", width: "100%", height: "100%", display: "flex", flexDirection: "column", justifyContent: "space-between", padding: "10px" }}>
                <div style={{ display: "flex", justify-content: "space-between" }}>
                  <div>
                    <img alt="User icon" style={{ width: "24px", height: "24px" }} />
                    {course.name}
                  </div>
                  <div>
                    <img alt="Star icon" style={{ width: "16px", height: "16px" }} />
                    {course.rating}
                  </div>
                </div>
                <div>
                  {course.description}
                </div>
                <div>
                  {course.date}
                </div>
              </div>
            </div>
          )));
        </div>
      )}
    </div>
  );
}

```

```

<style>`@keyframes spin { 0% { transform: rotate(0deg); } 100% { transform: rotate(360deg); } }</style>
</div>
) : courses.length === 0 ? (
  <p style={{ textAlign: "center", color: "#6f42c1" }}>🎉 No rejected courses!</p>
) : (
  <div style={styles.cardsContainer}>
    {courses.map(c => (
      <div key={c._id} style={styles.card} onMouseEnter={e =>
Object.assign(e.currentTarget.style, styles.cardHover)} onMouseLeave={e =>
Object.assign(e.currentTarget.style, styles.card)}>
        {c.thumbnail ? (
          <img src={c.thumbnail.startsWith("http") ? c.thumbnail :
`$import.meta.env.VITE_BASE_URL?.replace(/\/$/, "")}/${c.thumbnail?.replace(/^\//, "")}`}
          alt={c.title} style={styles.thumb} />
        ) : (
          <div style={{ ...styles.thumb, backgroundColor: "#f2f2f2", display: "flex", alignItems: "center", justifyContent: "center", color: "#999", fontSize: "12px" }}>No Thumbnail</div>
        )
      )
      <div style={styles.body}>
        <div style={styles.title}>{c.title}</div>
        <div style={styles.text}><span
style={styles.label}>Instructor:</span>{c.instructor?.name || "N/A"}</div>
        <div style={styles.text}><span style={styles.label}>Category:</span>{c.category?.name
|| "N/A"}</div>
        <div style={styles.text}><span style={styles.label}>Level:</span>{c.level || "-"}</div>
        <div style={styles.text}><span style={styles.label}>Price:</span>₹{c.price ?? 0}</div>

        <details style={styles.details}><summary style={styles.summary}>Lessons
({c.lessons?.length || 0})</summary>
          <ul style={styles.ul}>{c.lessons?.slice(0, 5).map(l => <li key={l._id}>{l.title}
          ({l.contentType})</li>)}</ul>
        </details>
      </div>
    )
  )
<div style={styles.footer}>Rejected on: {new Date(c.updatedAt ||
c.createdAt.toLocaleDateString("en-GB"))}</div>

```

```

        </div>
    )}
</div>
)}
</div>
);
}

```

Manage Category Page

The screenshot shows a dashboard interface with a purple sidebar on the left and a white main content area on the right.

Sidebar (Dashboard):

- Users
- Courses
- Categories** (selected)
 - Manage Categories
 - Suggestions
- Payments & Revenue
- Reports & Analytics
- Support
- Logout

Main Content Area (Manage Categories):

Manage Categories

Category	Status	Requested By	Actions
Next JS	approved	Admin	<button>Edit</button> <button>Delete</button>
Data Science	approved	Admin	<button>Edit</button> <button>Delete</button>
Frontend Development	approved	Admin	<button>Edit</button> <button>Delete</button>
Fullstack Development	approved	Admin	<button>Edit</button> <button>Delete</button>
MERN Stack Development	approved	Admin	<button>Edit</button> <button>Delete</button>
Backend Development	approved	Admin	<button>Edit</button> <button>Delete</button>
Javascript	approved	Admin	<button>Edit</button> <button>Delete</button>
Python	approved	Admin	<button>Edit</button> <button>Delete</button>
inst123	rejected	abc (instructor1@gmail.com)	<button>Edit</button> <button>Delete</button>
inst11	pending	abc (instructor1@gmail.com)	<button>Approve</button> <button>Reject</button> <button>Edit</button> <button>Delete</button>

manageCategories.jsx

```

import React, { useEffect, useState } from "react";
import api from "../../api/api";

export default function ManageCategories() {
    const [categories, setCategories] = useState([]);
    const [newCategory, setNewCategory] = useState("");
    const [search, setSearch] = useState("");
    const [editingId, setEditingId] = useState(null);

```

```

const [editName, setEditName] = useState("");

const fetchCats = async () => {
  const res = await api.get("/categories");
  setCategories(res.data);
};

const addCategory = async () => {
  if (!newCategory.trim()) return;
  await api.post("/categories/create", { name: newCategory });
  setNewCategory("");
  fetchCats();
};

const approveCategory = async (id) => {
  await api.put('/categories/approve/${id}`);
  fetchCats();
};

const rejectCategory = async (id) => {
  await api.put('/categories/reject/${id}');
  fetchCats();
};

const deleteCategory = async (id) => {
  await api.delete('/categories/${id}');
  fetchCats();
};

const saveEdit = async (id) => {
  await api.put('/categories/update/${id}', { name: editName });
  setEditingId(null);
  setEditName("");
  fetchCats();
};

useEffect(() => {
  fetchCats();
}, []);

const highlightText = (text) => {
  if (!search) return text;
  const regex = new RegExp(`(${search})`, "gi");
  return text.replace(regex, "<mark style='background:#e9d5ff'>$1</mark>");
}

```

```
};

return (
  <div
    style={{
      background: "#faf7ff",
      padding: "25px",
      borderRadius: "12px",
      boxShadow: "0 0 18px rgba(0,0,0,0.08)",
    }}
  >
  <h2 style={{ color: "#6a0dad", fontWeight: "700", marginBottom: "25px" }}>
    Manage Categories
  </h2>

  /* ADD CATEGORY */
  <div style={{ display: "flex", gap: "10px", marginBottom: "20px" }}>
    <input
      type="text"
      style={{
        flex: 1,
        padding: "8px",
        borderRadius: "8px",
        border: "2px solid #ddd",
        outline: "none",
        boxShadow: newCategory
          ? "0 0 10px rgba(138, 43, 226, 0.3)"
          : "none",
        transition: "0.3s",
      }}
      placeholder="Enter new category"
      value={newCategory}
      onChange={(e) => setNewCategory(e.target.value)}
    />
    <button
      onClick={addCategory}
      style={{
        padding: "10px 20px",
        background: "linear-gradient(90deg, #6a0dad, #a855f7)",
        border: "none",
        borderRadius: "8px",
        color: "white",
        fontWeight: "600",
        cursor: "pointer",
      }}
    >Add Category</button>
  </div>

```

```

        transition: "0.3s",
    )}
>
    Add
</button>
</div>

/* SEARCH */
<input
    type="text"
    placeholder="Search categories..."
    style={{
        width: "100%",
        padding: "7px",
        borderRadius: "8px",
        border: "1px solid #ccc",
        marginBottom: "20px",
    }}
    value={search}
    onChange={(e) => setSearch(e.target.value)}
/>

/* TABLE */
<table
    style={{
        width: "100%",
        borderCollapse: "separate",
        borderSpacing: "0 10px",
    }}
>
<thead>
<tr style={{ background: "#eee" }}>
    <th style={{ padding: "10px" }}>Category</th>
    <th>Status</th>
    <th>Requested By</th>
    <th>Actions</th>
</tr>
</thead>

<tbody>
{categories
    .filter((cat) =>
        cat.name.toLowerCase().includes(search.toLowerCase())
    )
}

```

```

.map((cat) => (
  <tr
    key={cat._id}
    style={{
      background: "white",
      borderRadius: "10px",
      boxShadow: "0 3px 12px rgba(0,0,0,0.05)",
      transition: "0.2s",
    }}
    onMouseEnter={(e) =>
      (e.currentTarget.style.transform = "scale(1.01)")
    }
    onMouseLeave={(e) =>
      (e.currentTarget.style.transform = "scale(1)")
    }
  >
  {/* NAME WITH EDIT MODE */}
  <td style={{ padding: "10px", fontWeight: "600" }}>
    {editingId === cat._id ? (
      <input
        type="text"
        value={editName}
        onChange={(e) => setEditName(e.target.value)}
        style={{
          padding: "5px 8px",
          borderRadius: "6px",
          border: "1px solid #aaa",
        }}
      />
    ) : (
      <span
        dangerouslySetInnerHTML={{
          __html: highlightText(cat.name),
        }}
      />
    )}
  </td>

  {/* STATUS */}
  <td style={{ padding: "10px" }}>
    <span
      style={{
        padding: "5px 12px",
        borderRadius: "6px",
      }}
    >

```

```

        color: "white",
        background:
          cat.status === "approved"
            ? "green"
            : cat.status === "pending"
            ? "orange"
            : "red",
      )}
    >
  {cat.status}
</span>
</td>

/* REQUESTED BY */
<td style={{ padding: "10px", color: "#555" }}>
  {cat.suggestedBy
    ? `${cat.suggestedBy.name} (${cat.suggestedBy.email})`
    : "Admin"}
</td>

/* ACTIONS */
<td style={{ padding: "10px" }}>
  {editingId === cat._id ? (
    <>
    <button
      onClick={() => saveEdit(cat._id)}
      style={{
        padding: "6px 12px",
        background: "#6a0dad",
        color: "white",
        borderRadius: "6px",
        border: "none",
        marginRight: "6px",
      }}
    >
      Save
    </button>
    <button
      onClick={() => setEditingId(null)}
      style={{
        padding: "6px 12px",
        background: "grey",
        color: "white",
        borderRadius: "6px",
      }}
    >
  
```

```
        border: "none",
    )}
>
    Cancel
</button>
</>
):(
<>
    {cat.status === "pending" && (
        <>
            <button
                onClick={() => approveCategory(cat._id)}
                style={{
                    padding: "6px 12px",
                    marginRight: "8px",
                    background: "#6a0dad",
                    color: "white",
                    borderRadius: "6px",
                    border: "none",
                    cursor: "pointer",
                }}
            >
                Approve
            </button>

            <button
                onClick={() => rejectCategory(cat._id)}
                style={{
                    padding: "6px 12px",
                    background: "red",
                    color: "white",
                    borderRadius: "6px",
                    border: "none",
                    cursor: "pointer",
                    marginRight: "8px",
                }}
            >
                Reject
            </button>
        </>
    )}

    <button
        onClick={() => {
```

```

        setEditingId(cat._id);
        setEditName(cat.name);
    )}
    style={{
        padding: "6px 12px",
        background: "#a855f7",
        color: "white",
        borderRadius: "6px",
        border: "none",
        cursor: "pointer",
        marginRight: "8px",
    }}
>
    Edit
</button>

<button
    onClick={() => deleteCategory(cat._id)}
    style={{
        padding: "6px 12px",
        background: "darkred",
        color: "white",
        borderRadius: "6px",
        border: "none",
        cursor: "pointer",
    }}
>
    Delete
</button>
</>
    )}
</td>
</tr>
))}

/* NO RESULTS FOUND */
{categories.filter((cat) =>
    cat.name.toLowerCase().includes(search.toLowerCase())
).length === 0 && (
<tr>
    <td colSpan="4" style={{ textAlign: "center", padding: "20px" }}>
        No matching categories found.
    </td>
</tr>
)
}

```

```

        )}
      </tbody>
    </table>
  </div>
);
}

```

Category Suggestion Page

The screenshot shows a user interface for managing category suggestions. On the left, there is a purple sidebar menu titled "Dashboard" with the following items:

- Users
- Courses
- Categories** (selected)
- Manage Categories
- Suggestions
- Payments & Revenue
- Reports & Analytics
- Support

At the bottom of the sidebar is a "Logout" button.

The main content area is titled "Category Suggestions" and contains a search bar labeled "Search suggestions...". To the right of the search bar is a "Refresh" button. Below the search bar is a table with the following columns:

Suggested Name	Suggested By	Date	Actions
inst11	abc instructor1@gmail.com	3 days ago	<button>Approve</button> <button>Reject</button>

categorySuggestions.jsx

```

import React, { useEffect, useState } from "react";
import api from "../../api/api";
import moment from "moment";

export default function CategorySuggestions() {
  const [suggestions, setSuggestions] = useState([]);
  const [filtered, setFiltered] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState("");
  const [search, setSearch] = useState("");

  const fetchSuggestions = async () => {
    setLoading(true);
    setError("");
    try {
      const res = await api.get("/categories/pending");
      setSuggestions(res.data);
    }
  }
}

```

```

    setFiltered(res.data);
  } catch (err) {
    setError("Failed to load suggestions. Please try again.");
  }

  setLoading(false);
};

const approve = async (id) => {
  try {
    await api.put(`/categories/approve/${id}`);
    fetchSuggestions();
  } catch {
    alert("Something went wrong");
  }
};

const reject = async (id) => {
  try {
    await api.put(`/categories/reject/${id}`);
    fetchSuggestions();
  } catch {
    alert("Something went wrong");
  }
};

const onSearch = (text) => {
  setSearch(text);
  const filteredList = suggestions.filter((s) =>
    s.name.toLowerCase().includes(text.toLowerCase())
  );
  setFiltered(filteredList);
};

useEffect(() => {
  fetchSuggestions();
}, []);

return (
  <div>
    <h2 className="mb-3">Category Suggestions</h2>

    {/* Search Bar */}
    <div className="d-flex justify-content-between align-items-center mb-3">

```

```

<input
  type="text"
  className="form-control w-50"
  placeholder="Search suggestions..."
  value={search}
  onChange={(e) => onSearch(e.target.value)}
/>

<button className="btn btn-secondary" onClick={fetchSuggestions}>
  Refresh
</button>
</div>

{/* Loading */}
{loading && <p>Loading...</p>}

{/* Error */}
{error && (
  <div className="alert alert-danger">
    {error}
    <button className="btn btn-sm btn-light ms-2" onClick={fetchSuggestions}>
      Retry
    </button>
  </div>
)}

{/* No Data */}
{!loading && filtered.length === 0 && (
  <div className="alert alert-info">No pending category suggestions.</div>
)}

{/* Table */}
{!loading && filtered.length > 0 && (
  <table className="table table-bordered table-striped">
    <thead className="table-light">
      <tr>
        <th>Suggested Name</th>
        <th>Suggested By</th>
        <th>Date</th>
        <th style={{ width: "200px" }}>Actions</th>
      </tr>
    </thead>

    <tbody>

```

```

{filtered.map((cat) => (
  <tr key={cat._id}>
    <td>
      <strong>{cat.name}</strong>
    </td>

    <td>
      {cat.suggestedBy?.name || "Unknown"} <br />
      <span className="badge bg-dark mt-1">
        {cat.suggestedBy?.email}
      </span>
    </td>

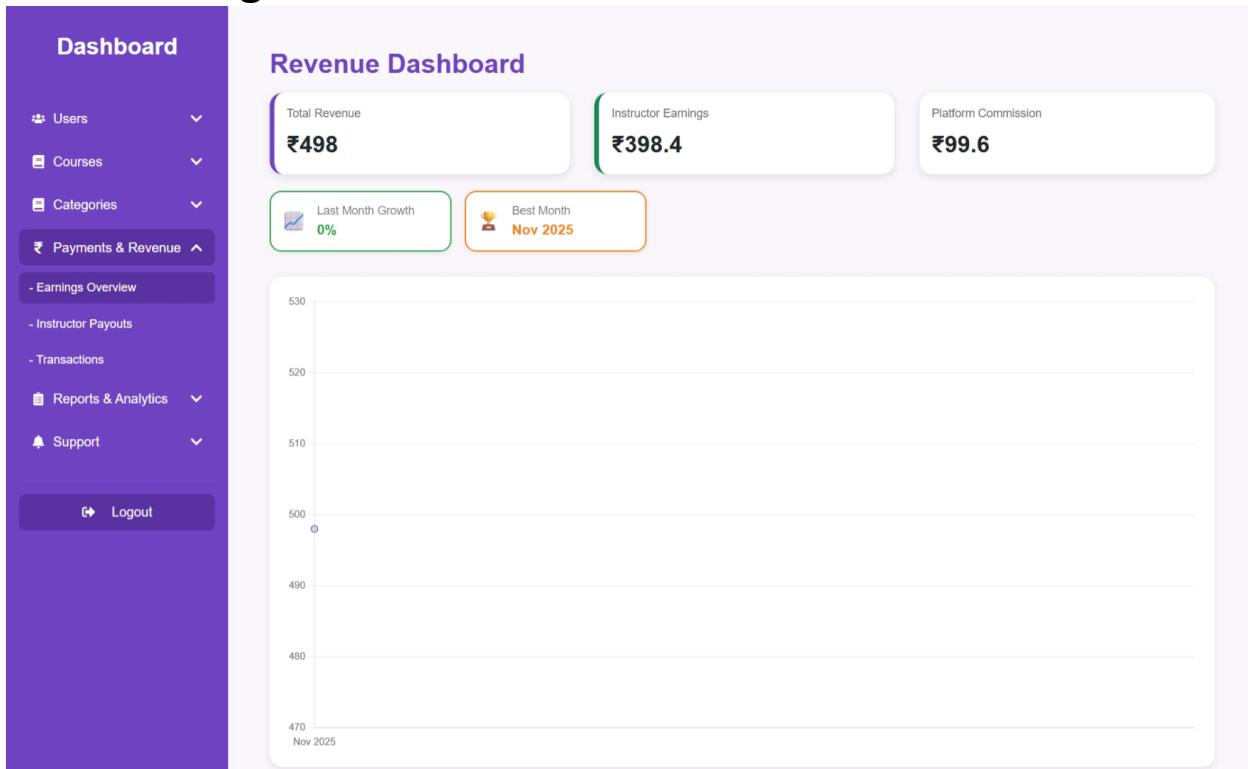
    <td>{moment(cat.createdAt).fromNow()}</td>

    <td>
      <button
        className="btn btn-success btn-sm me-2"
        onClick={() => approve(cat._id)}
      >
        Approve
      </button>

      <button
        className="btn btn-danger btn-sm"
        onClick={() => reject(cat._id)}
      >
        Reject
      </button>
    </td>
  </tr>
))
</tbody>
</table>
)
</div>
);
}

```

Revenue Page



Revenue.jsx

```
import React, { useEffect, useState } from "react";
import { Line } from "react-chartjs-2";
import { Chart, LineElement, CategoryScale, LinearScale, PointElement } from "chart.js";
import api from "../../api/api";

Chart.register(LineElement, CategoryScale, LinearScale, PointElement);

export default function Revenue() {
  const [summary, setSummary] = useState(null);

  useEffect(() => {
    api.get("/admin/revenue")
      .then(res => setSummary(res.data))
      .catch(console.error);
  }, []);

  if (!summary) {
    return (

```

```

        <div style={{ display: "flex", flexDirection: "column", justifyContent: "center", alignItems: "center", height: "70vh", color: "#6f42c1", }}>
            <div style={{ border: "4px solid #f3f3f3", borderTop: "4px solid #6f42c1", borderRadius: "50%", width: "50px", height: "50px", animation: "spin 1s linear infinite", marginBottom: "20px", }} />
            <p>Loading revenue dashboard...</p>
            <style>{@keyframes spin { 0% { transform: rotate(0deg); } 100% { transform: rotate(360deg); } }}</style>
        </div>
    );
}

const months = Object.keys(summary.monthlyData || {});
const amounts = months.map(m => summary.monthlyData[m]);

// ✅ Calculate month-to-month growth
const last = amounts[amounts.length - 1] || 0;
const prev = amounts[amounts.length - 2] || 0;
const growth = prev ? (((last - prev) / prev) * 100).toFixed(1) : 0;

// ✅ Find best revenue month
const maxAmount = Math.max(...amounts);
const bestMonth = months[amounts.indexOf(maxAmount)];

function formatAmount(amount) {
    return Number(amount).toString().includes(".")
        ? Number(amount).toFixed(3).replace(/\.\?0+$/, "")
        : amount;
}

return (
    <div style={styles.wrapper}>
        <h2 style={styles.heading}>Revenue Dashboard</h2>

        <div style={styles.cardRow}>
            <Card title="Total Revenue" value={`₹${summary.totalRevenue}`} color="#6f42c1" />
            <Card title="Instructor Earnings" value={`₹${summary.totalInstructorEarning}`} color="#198754" />
            <Card title="Platform Commission" value={`₹${formatAmount(summary.platformCommission)}`}/>
        </div>

        {/* 💡 Extra Insights */}

```

```

<div style={styles.subCards}>
  <SmallCard
    title="Last Month Growth"
    value={`${growth}%`}
    icon={growth >= 0 ? "📈" : "📉"}
    color={growth >= 0 ? "#28a745" : "#dc3545"}
  />

  <SmallCard
    title="Best Month"
    value={bestMonth}
    icon="🏆"
    color="#fd7e14"
  />
</div>

/* Chart */
<div style={styles.chartBox}>
  <Line
    data={{
      labels: months,
      datasets: [
        {
          label: "Monthly Revenue",
          data: amounts,
          borderColor: "#6f42c1",
          backgroundColor: "rgba(111,66,193,0.2)",
          tension: 0.3,
          pointRadius: 4,
        }
      ]
    }}
    options={{
      responsive: true,
      animation: { duration: 900 },
      plugins: { legend: { display: false } },
      scales: {
        x: { grid: { display: false } },
        y: { grid: { color: "#eee" } }
      }
    }}
  />
</div>
</div>
);

}

```

```

function Card({ title, value, color }) {
  return (
    <div style={{ ...styles.card, borderLeft: `6px solid ${color}` }}>
      <div style={styles.cardTitle}>{title}</div>
      <div style={styles.cardValue}>{value}</div>
    </div>
  );
}

function SmallCard({ title, value, icon, color }) {
  return (
    <div style={{ ...styles.smallCard, borderColor: color }}>
      <div style={{ ...styles.smallIcon, color }}>{icon}</div>
      <div>
        <div style={styles.smallTitle}>{title}</div>
        <div style={{ ...styles.smallValue, color }}>{value}</div>
      </div>
    </div>
  );
}

/* ✅ Internal Inline CSS (clean, modern, responsive) */
const styles = {
  wrapper: {
    padding: 20,
    animation: "fadeln 0.4s ease",
  },
  heading: {
    color: "#6f42c1",
    fontWeight: "600",
    marginBottom: 16,
  },
  loading: {
    fontSize: 22,
    fontWeight: "bold",
    textAlign: "center",
    padding: 40,
  },
  cardRow: {
    display: "flex",
    gap: 30,
    flexWrap: "wrap",
  },
};

```

```
card: {
  flex: 1,
  minWidth: 250,
  background: "white",
  padding: 15,
  borderRadius: 16,
  boxShadow: "0 3px 10px rgba(0,0,0,0.1)",
  transition: "transform 0.3s",
},
cardTitle: {
  fontSize: 14,
  opacity: 0.7,
},
cardValue: {
  fontSize: 28,
  fontWeight: "700",
  marginTop: 6,
},
chartBox: {
  background: "white",
  marginTop: 30,
  padding: 20,
  borderRadius: 16,
  boxShadow: "0 3px 10px rgba(0,0,0,0.08)",
},
subCards: {
  display: "flex",
  gap: 16,
  marginTop: 20,
  flexWrap: "wrap",
},
smallCard: {
  display: "flex",
  alignItems: "center",
  gap: 12,
  border: "2px solid",
  padding: 12,
  borderRadius: 14,
  minWidth: 220,
  background: "white",
  boxShadow: "0 2px 6px rgba(0,0,0,0.06)",
},
smallIcon: {
  fontSize: 23,
```

```

},
smallTitle: {
  fontSize: 14,
  opacity: 0.6,
},
smallValue: {
  fontSize: 17,
  fontWeight: 700,
},
},
);

```

Instructor Payouts History

Instructor	Courses Sold	Total Earnings	Last Payout	Trend
A abc	2	₹398.4	19/11/2025	▼

Payout.jsx

```

import React, { useEffect, useState } from "react";
import api from "../../api/api";

export default function Payouts() {
  const [payouts, setPayouts] = useState([]);
  const [filtered, setFiltered] = useState([]);
  const [monthFilter, setMonthFilter] = useState("all");
  const [search, setSearch] = useState("");

  useEffect(() => {
    api.get("/admin/payouts")

```

```

.then(res => {
  setPayouts(res.data);
  setFiltered(res.data);
})
.catch(console.error);
}, []));

// ✅ Apply filters (search + month)
useEffect(() => {
  let data = [...payouts];

  if (search.trim()) {
    data = data.filter(p =>
      (p.instructor || "").toLowerCase().includes(search.toLowerCase())
    );
  }

  if (monthFilter !== "all") {
    data = data.filter(p => {
      if (!p.lastPayout) return false;
      return new Date(p.lastPayout).getMonth() + 1 === Number(monthFilter);
    });
  }

  setFiltered(data);
}, [search, monthFilter, payouts]);

const monthNames = [
  "January", "February", "March", "April", "May", "June",
  "July", "August", "September", "October", "November", "December"
];

return (
  <>
  <h2 style={{ color: "#6f42c1", marginBottom: 16 }}>Instructor Payouts</h2>

  <div style={styles.cardRow}>

```

```

<SummaryCard title="Total Instructors" value={payouts.length} />
<SummaryCard title="Total Earnings" value={"₹" + total(payouts,
"totalEarning")}>/
<SummaryCard title="Total Courses Sold" value={total(payouts,
"coursesSold")}>/
</div>

<div style={styles.filters}>
<input
  type="text"
  placeholder="Search instructor..."
  value={search}
  onChange={(e) => setSearch(e.target.value)}
  style={styles.search}
/>

<select
  value={monthFilter}
  onChange={(e) => setMonthFilter(e.target.value)}
  style={styles.select}>
  <option value="all">All Months</option>
  {monthNames.map((name, index) => (
    <option key={index} value={index + 1}>
      {name}
    </option>
  )))
</select>

</div>

/* ✅ Table Section */
<div style={styles.card}>
  <table style={styles.table}>
    <thead>
      <tr>

```

```

<th style={styles.th}>Instructor</th>
<th style={styles.th}>Courses Sold</th>
<th style={styles.th}>Total Earnings</th>
<th style={styles.th}>Last Payout</th>
<th style={styles.th}>Trend</th>
</tr>
</thead>

<tbody>
{filtered.map((p, i) => (
  <tr key={i} style={styles.tr}>
    {/* ✓ Avatar + Name */}
    <td style={styles.td}>
      <div style={styles.userBox}>
        <div style={styles.avatar}>
          {(p.instructor || "U").charAt(0).toUpperCase()}
        </div>
        <div>
          <strong>{p.instructor}</strong>
        </div>
      </div>
    </td>
    {/* ✓ Course Badge */}
    <td style={styles.td}>
      <span style={styles.chip}>{p.coursesSold}</span>
    </td>
    {/* ✓ Colored earning */}
    <td style={{ ...styles.td, color: earningColor(p.totalEarning) }}>
      ₹{p.totalEarning}
    </td>
    {/* ✓ Last payout date */}
    <td style={styles.td}>
      {p.lastPayout ? new Date(p.lastPayout).toLocaleDateString() : "—"}
    </td>
  </tr>
))
</tbody>

```

```

/* ✅ Trend */
<td style={styles.td}>
  {p.trend === "up" ? (
    <span style={styles.trendUp}>▲</span>
  ) : (
    <span style={styles.trendDown}>▼</span>
  )}
</td>
</tr>
))}
</tbody>
</table>
</div>
</>
);
}

/*
 * ✅ Summation Helper *
 */
function total(list, field) {
  return list.reduce((a, b) => a + (b[field] || 0), 0);
}

/*
 * ✅ Dynamic earning color *
 */
function earningColor(amount) {
  if (amount > 50000) return "#28a745"; // green
  if (amount > 10000) return "#fd7e14"; // orange
  return "#dc3545"; // red
}

/*
 * ✅ Summary Card Component *
 */
function SummaryCard({ title, value }) {
  return (
    <div style={styles.summaryCard}>
      <h5 style={styles.summaryTitle}>{title}</h5>
      <h2 style={styles.summaryValue}>{value}</h2>
    </div>
  )
}

```

```
};

/* ✅ Internal CSS - modern & classy */
const styles = {
  cardRow: {
    display: "flex",
    gap: 20,
    flexWrap: "wrap",
    marginBottom: 20
  },
  summaryCard: {
    flex: "1 1 200px",
    background: "#fff",
    padding: 20,
    borderRadius: 14,
    boxShadow: "0 3px 8px rgba(0,0,0,0.08)",
  },
  summaryTitle: {
    margin: 0,
    color: "#6f42c1",
    fontWeight: 600,
  },
  summaryValue: {
    margin: "10px 0 0 0",
    fontWeight: 700,
    color: "#2c3e50",
  },
  filters: {
    display: "flex",
    justifyContent: "space-between",
    marginBottom: 15,
  },
  search: {
    padding: "10px 14px",
    width: 250,
```

```
borderRadius: 10,  
border: "1px solid #ccc",  
,  
select: {  
padding: "10px 14px",  
borderRadius: 10,  
border: "1px solid #ccc",  
,  
card: {  
background: "#fff",  
padding: 20,  
borderRadius: 14,  
boxShadow: "0 3px 8px rgba(0,0,0,0.08)",  
overflowX: "auto",  
,  
table: {  
width: "100%",  
borderCollapse: "separate",  
borderSpacing: "0 12px",  
,  
th: {  
textAlign: "left",  
padding: "10px 14px",  
fontWeight: 700,  
color: "#6f42c1",  
whiteSpace: "nowrap",  
,  
tr: {  
background: "#f7f5ff",  
transition: "0.2s",  
,  
td: {  
padding: "14px",  
,
```

```
userBox: {
  display: "flex",
  alignItems: "center",
  gap: 12,
},
avatar: {
  width: 34,
  height: 34,
  borderRadius: "50%",
  background: "#6f42c1",
  color: "#fff",
  display: "flex",
  justifyContent: "center",
  alignItems: "center",
  fontWeight: 700,
},
chip: {
  background: "#eee",
  padding: "6px 12px",
  borderRadius: 20,
  fontWeight: 600,
},
trendUp: {
  color: "#28a745",
  fontSize: 20,
},
trendDown: {
  color: "#dc3545",
  fontSize: 20,
},
};
```

Transactions Page

The screenshot shows a dashboard interface for managing transactions. On the left, a sidebar menu includes sections for Users, Courses, Categories, Payments & Revenue (with Earnings Overview, Instructor Payouts, and Transactions selected), Reports & Analytics, and Support. A Logout button is at the bottom. The main area is titled 'Transactions' and displays summary statistics: Total Transactions (2), Total Revenue (₹498), and Successful Payments (2). Below this is a search bar and a table of transaction details:

Student	Instructor	Course	Amount	Status	Date
student1	abc	Next JS Fundamentals	₹199	Completed	19/11/2025, 1:47:18 am
student1	abc	Advanced MERN Project	₹299	Completed	19/11/2025, 1:42:16 am

Transactions.jsx

```
import React, { useEffect, useState } from "react";
import api from "../../api/api";

export default function Transactions() {
  const [txns, setTxns] = useState([]);
  const [filtered, setFiltered] = useState([]);

  const [search, setSearch] = useState("");
  const [status, setStatus] = useState("all");

  useEffect(() => {
    api
      .get("/admin/transactions")
      .then((res) => {
        setTxns(res.data);
        setFiltered(res.data);
      })
      .catch(console.error);
  }, []);

  // ✅ Filtering logic
  useEffect(() => {
    let data = [...txns];

    if (search.trim()) {
```

```

data = data.filter((t =>
  `${t.student?.name} ${t.instructor?.name} ${t.course?.title}`
    .toLowerCase()
    .includes(search.toLowerCase())
);
}

if (status !== "all") {
  data = data.filter((t => t.status === status);
}

setFiltered(data);
}, [search, status, txns]);

return (
  <>
  <h2 style={{ color: "#6f42c1", marginBottom: 16 }}>Transactions</h2>

  {/* ✅ Summary Cards */}
  <div style={styles.summaryRow}>
    <SummaryCard title="Total Transactions" value={txns.length} />
    <SummaryCard title="Total Revenue" value={"₹" + total(txns, "amount")} />
    <SummaryCard
      title="Successful Payments"
      value={txns.filter((t => t.status === "completed").length}
    />
  </div>

  {/* ✅ Filters */}
  <div style={styles.filters}>
    <input
      type="text"
      placeholder="Search student, instructor, or course..."
      value={search}
      onChange={(e) => setSearch(e.target.value)}
      style={styles.search}
    />

    <select
      value={status}
      onChange={(e) => setStatus(e.target.value)}
      style={styles.select}
    >
      <option value="all">All Status</option>

```

```

<option value="completed">Completed</option>
<option value="failed">Failed</option>
<option value="pending">Pending</option>
</select>
</div>

/* ✅ Table */
<div style={styles.card}>
  <table style={styles.table}>
    <thead>
      <tr>
        <th style={styles.th}>Student</th>
        <th style={styles.th}>Instructor</th>
        <th style={styles.th}>Course</th>
        <th style={styles.th}>Amount</th>
        <th style={styles.th}>Status</th>
        <th style={styles.th}>Date</th>
      </tr>
    </thead>

    <tbody>
      {filtered.map((t) => (
        <tr key={t._id} style={styles.tr}>
          <td style={styles.td}>{t.student?.name}</td>
          <td style={styles.td}>{t.instructor?.name}</td>
          <td style={styles.td}>{t.course?.title}</td>
          <td style={{ ...styles.td, fontWeight: 600 }}>₹{t.amount}</td>

          /* ✅ Status badge */
          <td style={styles.td}>
            <span style={statusBadge(t.status)}>{t.status}</span>
          </td>

          <td style={styles.td}>
            {new Date(t.paymentDate).toLocaleString()}
          </td>
        </tr>
      ))}
    </tbody>
  </table>
</div>
</>
);

}

```

```

/* ✅ Summary helper */
function total(list, field) {
  return list.reduce((a, b) => a + (b[field] || 0), 0);
}

/* ✅ Status color badge */
function statusBadge(status) {
  const base = {
    padding: "5px 12px",
    borderRadius: 20,
    fontSize: 13,
    textTransform: "capitalize",
    fontWeight: 600,
    color: "#fff",
  };

  switch (status) {
    case "completed":
      return { ...base, background: "#28a745" };
    case "failed":
      return { ...base, background: "#dc3545" };
    case "pending":
      return { ...base, background: "#ffc107", color: "#000" };
    default:
      return base;
  }
}

/* ✅ Summary card component */
function SummaryCard({ title, value }) {
  return (
    <div style={styles.summaryCard}>
      <h5 style={styles.summaryTitle}>{title}</h5>
      <h2 style={styles.summaryValue}>{value}</h2>
    </div>
  );
}

/* ✅ Styles */
const styles = {
  summaryRow: {
    display: "flex",

```

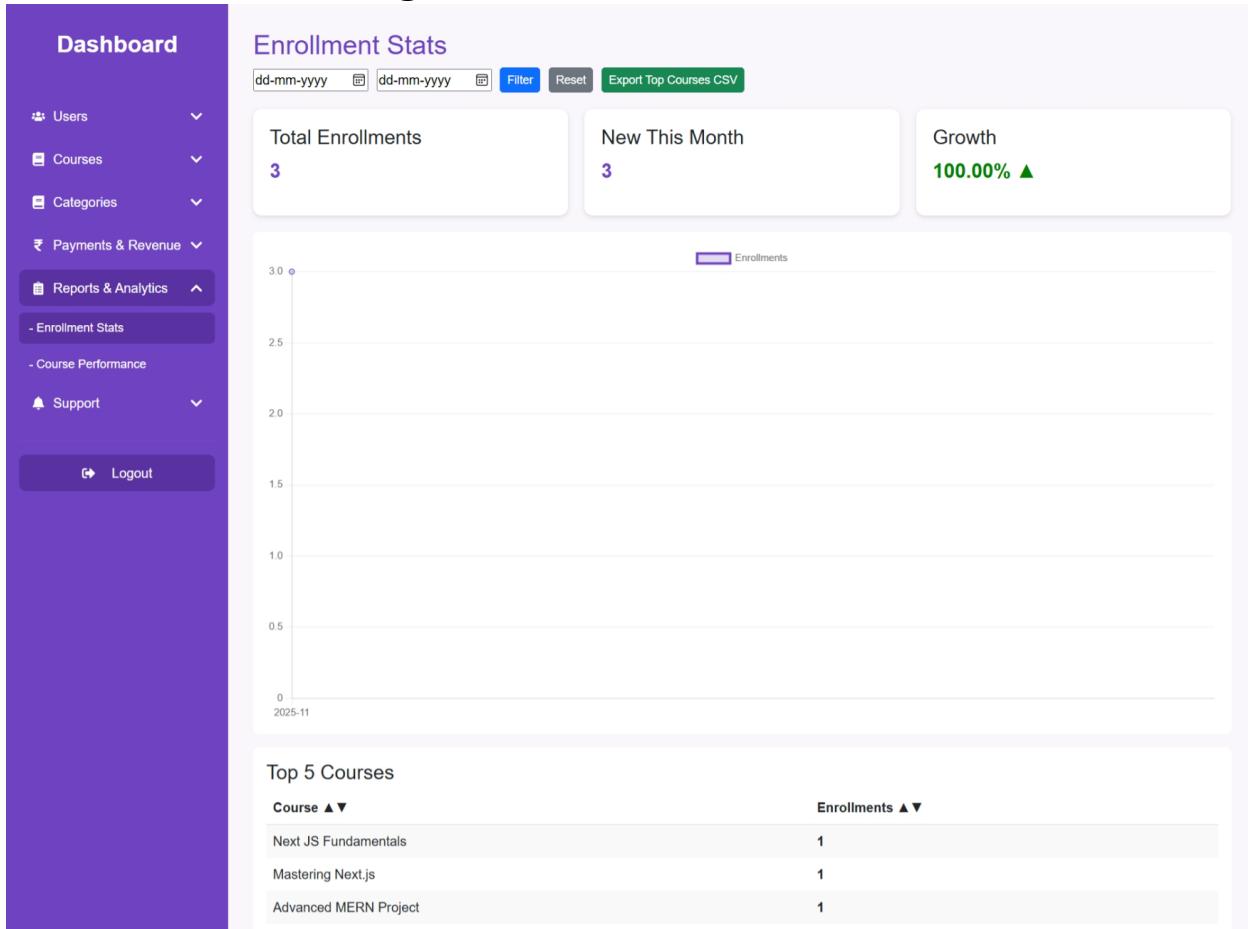
```
gap: 20,
flexWrap: "wrap",
marginBottom: 20,
},

summaryCard: {
  flex: "1 1 200px",
  background: "#fff",
  padding: 20,
  borderRadius: 14,
  boxShadow: "0 3px 8px rgba(0,0,0,0.08)",
},
summaryTitle: { margin: 0, color: "#6f42c1", fontWeight: 600 },
summaryValue: { margin: "10px 0 0 0", fontWeight: 700, color: "#2c3e50" },

filters: {
  display: "flex",
  justifyContent: "space-between",
  marginBottom: 15,
},
search: {
  padding: "10px 14px",
  width: 300,
  borderRadius: 10,
  border: "1px solid #ccc",
},
select: {
  padding: "10px 14px",
  borderRadius: 10,
  border: "1px solid #ccc",
},
card: {
  background: "#fff",
  padding: 20,
  borderRadius: 14,
  boxShadow: "0 3px 8px rgba(0,0,0,0.08)",
  overflowX: "auto",
},
table: { width: "100%", borderCollapse: "separate", borderSpacing: "0 12px" },
th: { textAlign: "left", padding: "10px 14px", fontWeight: 700, color: "#6f42c1" },
```

```
tr: { background: "#f7f5ff" },  
td: { padding: "14px" },  
};
```

Enrollment Stats Page



enrollmentStats.jsx

```
import React, { useEffect, useState } from "react";  
import { Line } from "react-chartjs-2";  
import api from "../../api/api";  
import "chart.js/auto";  
  
export default function EnrollmentStats() {  
  const [data, setData] = useState(null);  
  const [loading, setLoading] = useState(true);  
  const [startDate, setStartDate] = useState("");
```

```

const [endDate, setEndDate] = useState("");
const [sortKey, setSortKey] = useState("count");
const [sortOrder, setSortOrder] = useState("desc");

useEffect(() => {
  async function loadData() {
    await fetchData();
  }
  loadData();
}, []);

const fetchData = async (start, end) => {
  setLoading(true);
  let url = "/admin/enrollment-stats";
  if (start && end) url += `?start=${start}&end=${end}`;
  try {
    const res = await api.get(url);
    setData(res.data);
  } catch (err) {
    console.error(err);
    setData({
      labels: [],
      values: [],
      totalEnrollments: 0,
      newThisMonth: 0,
      growth: 0,
      topCourses: []
    });
  } finally {
    setLoading(false);
  }
};

const downloadCSV = () => {
  if (!data || !data.topCourses.length) return;
  const headers = ["Course", "Enrollments"];
  const rows = data.topCourses.map(c => [c.course, c.count]);
  const csvContent =
    "data:text/csv;charset=utf-8," +
    [headers, ...rows].map(e => e.join(",")).join("\n");
  const encodedUri = encodeURI(csvContent);
  const link = document.createElement("a");
  link.setAttribute("href", encodedUri);
  link.setAttribute("download", "top_courses.csv");
}

```

```

document.body.appendChild(link);
link.click();
document.body.removeChild(link);
};

const sortTopCourses = key => {
  const order = sortKey === key && sortOrder === "desc" ? "asc" : "desc";
  setSortKey(key);
  setSortOrder(order);
  if (!data) return;
  const sorted = [...data.topCourses].sort((a, b) => {
    if (key === "course")
      return order === "asc"
        ? a.course.localeCompare(b.course)
        : b.course.localeCompare(a.course);
    return order === "asc" ? a[key] - b[key] : b[key] - a[key];
  });
  setData({ ...data, topCourses: sorted });
};

if (loading)
  return (
    <p style={{ textAlign: "center", marginTop: "50px" }}>
      Loading enrollment stats...
    </p>
  );
}

if (!data || !data.labels.length)
  return (
    <p style={{ textAlign: "center", marginTop: "50px" }}>
      No enrollment data found.
    </p>
  );
}

return (
  <div>
    <h2 style={{ color: "#6f42c1" }}>Enrollment Stats</h2>

    /* Date Range Filter */
    <div
      style={{
        display: "flex",
        gap: "10px",
        marginBottom: "20px",
      }
    >

```

```

        flexWrap: "wrap",
        alignItems: "center",
    )}
>
<input
    type="date"
    value={startDate}
    onChange={e => setStartDate(e.target.value)}
/>
<input
    type="date"
    value={endDate}
    onChange={e => setEndDate(e.target.value)}
/>
<button
    className="btn btn-sm btn-primary"
    onClick={() => fetchData(startDate, endDate)}
>
    Filter
</button>
<button
    className="btn btn-sm btn-secondary"
    onClick={() => {
        setStartDate("");
        setEndDate("");
        fetchData();
    }}
>
    Reset
</button>
<button className="btn btn-sm btn-success" onClick={downloadCSV}>
    Export Top Courses CSV
</button>
</div>

/* KPI Cards */
<div
    style={{
        display: "flex",
        gap: "20px",
        marginBottom: "20px",
        flexWrap: "wrap",
    }}
>

```

```

<div style={cardStyle}>
  <h4>Total Enrollments</h4>
  <p style={kpiStyle}>{data.totalEnrollments}</p>
</div>
<div style={cardStyle}>
  <h4>New This Month</h4>
  <p style={kpiStyle}>{data.newThisMonth}</p>
</div>
<div style={cardStyle}>
  <h4>Growth</h4>
  <p
    style={{...kpiStyle,
      color: data.growth >= 0 ? "green" : "red",
    }}
  >
    {data.growth.toFixed(2)}% {data.growth >= 0 ? "▲" : "▼"}
  </p>
</div>
</div>

/* Enrollment Line Chart */
<div className="bg-white p-3 rounded mt-3">
  <Line
    data={{
      labels: data.labels,
      datasets: [
        {
          label: "Enrollments",
          data: data.values,
          borderColor: "#6f42c1",
          backgroundColor: "rgba(111,66,193,0.2)",
          tension: 0.3,
        },
      ],
    }}
    options={{
      responsive: true,
      plugins: {
        legend: { display: true, position: "top" },
        tooltip: { mode: "index", intersect: false },
      },
      scales: {
        y: { beginAtZero: true, grid: { color: "#f0f0f0" } },
      }
    }}
  </Line>
</div>

```

```

        x: { grid: { color: "#f0f0f0" } },
      },
    }}
  />
</div>

/* Top Courses */
<div className="bg-white p-3 rounded mt-3">
  <h4>Top 5 Courses</h4>
  <table style={{ width: "100%", borderCollapse: "collapse" }}>
    <thead>
      <tr>
        <th style={thStyle} onClick={() => sortTopCourses("course")}>
          Course ▲▼
        </th>
        <th style={thStyle} onClick={() => sortTopCourses("count")}>
          Enrollments ▲▼
        </th>
      </tr>
    </thead>
    <tbody>
      {data.topCourses.map((c, idx) => (
        <tr
          key={idx}
          style={{ background: idx % 2 === 0 ? "#f9f9f9" : "#fff" }}
        >
          <td style={tdStyle}>{c.course}</td>
          <td style={{ ...tdStyle, fontWeight: "bold" }}>{c.count}</td>
        </tr>
      ))}
    </tbody>
  </table>
</div>
</div>
);

}

// Styles
const cardStyle = {
  flex: 1,
  background: "#fff",
  padding: "20px",
  borderRadius: "10px",
  boxShadow: "0px 2px 5px rgba(0,0,0,0.1)",
}

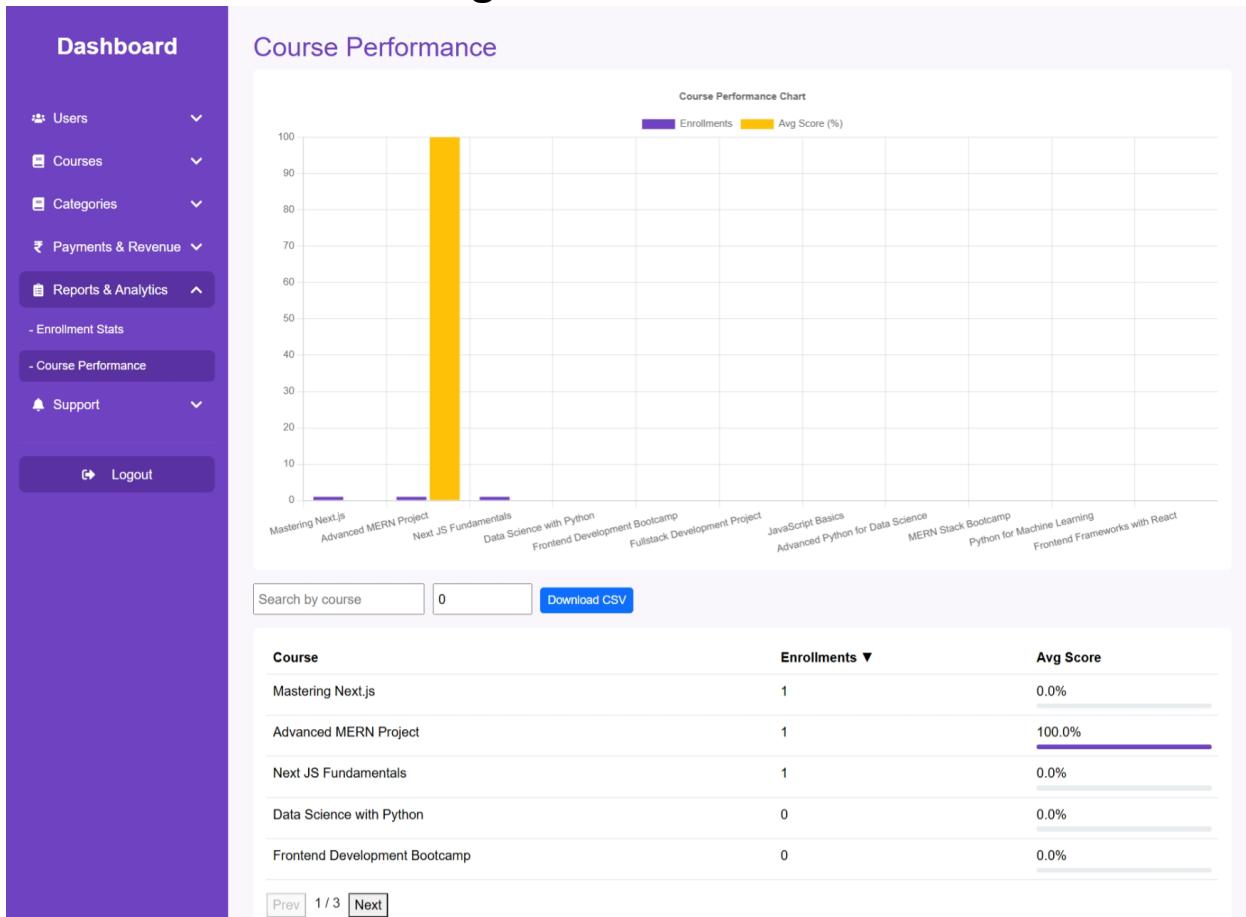
```

```

        minWidth: "150px",
    );
const kpiStyle = { fontSize: "24px", fontWeight: "bold", color: "#6f42c1" };
const thStyle = {
    textAlign: "left",
    padding: "8px",
    borderBottom: "1px solid #ddd",
    cursor: "pointer",
};
const tdStyle = { padding: "8px" };

```

Course Performance Page



coursePerformance.jsx

```

import React, { useEffect, useState } from "react";
import api from "../../api/api";
import { Bar } from "react-chartjs-2";

```

```

import {
  Chart as ChartJS,
  CategoryScale,
  LinearScale,
  BarElement,
  Title,
  Tooltip,
  Legend
} from "chart.js";

ChartJS.register(CategoryScale, LinearScale, BarElement, Title, Tooltip, Legend);

export default function CoursePerformance() {
  const [report, setReport] = useState([]);
  const [loading, setLoading] = useState(true);
  const [sortKey, setSortKey] = useState("enrollments");
  const [sortOrder, setSortOrder] = useState("desc");
  const [search, setSearch] = useState("");
  const [minEnroll, setMinEnroll] = useState(0);
  const [page, setPage] = useState(1);
  const pageSize = 5;

  useEffect(() => {
    api.get("/admin/course-performance")
      .then(res => setReport(res.data))
      .catch(console.error)
      .finally(() => setLoading(false));
  }, []);

  const sortData = (key) => {
    const order = sortKey === key && sortOrder === "desc" ? "asc" : "desc";
    setSortKey(key);
    setSortOrder(order);
    const sorted = [...report].sort((a, b) =>
      key === "title"
        ? (order === "asc" ? a.title.localeCompare(b.title) : b.title.localeCompare(a.title))
        : (order === "asc" ? a[key] - b[key] : b[key] - a[key])
    );
    setReport(sorted);
  };

  const filteredReport = report
    .filter(r => r.enrollments >= minEnroll)
    .filter(r => r.title.toLowerCase().includes(search.toLowerCase())));
}

```

```

const paginatedReport = filteredReport.slice((page - 1) * pageSize, page * pageSize);
const totalPages = Math.ceil(filteredReport.length / pageSize);

const downloadCSV = () => {
  const headers = ["Course", "Enrollments", "Avg Score"];
  const rows = filteredReport.map(r => [r.title, r.enrollments, r.avgScore.toFixed(1)]);
  let csvContent = "data:text/csv;charset=utf-8," + [headers, ...rows].map(e =>
    e.join(",")).join("\n");
  const encodedUri = encodeURI(csvContent);
  const link = document.createElement("a");
  link.setAttribute("href", encodedUri);
  link.setAttribute("download", "course_performance.csv");
  document.body.appendChild(link);
  link.click();
  document.body.removeChild(link);
};

if (loading) {
  return (
    <div style={{ display: "flex", flexDirection: "column", justifyContent: "center", alignItems: "center", height: "70vh", color: "#6f42c1", }}>
      <div style={{ border: "4px solid #f3f3f3", borderTop: "4px solid #6f42c1", borderRadius: "50%", width: "50px", height: "50px", animation: "spin 1s linear infinite", marginBottom: "20px", }} />
        <p>Loading course performance...</p>
        <style>`@keyframes spin { 0% { transform: rotate(0deg); } 100% { transform: rotate(360deg); } }`</style>
      </div>
    );
}
}

// Bar chart data
const chartData = {
  labels: filteredReport.map(r => r.title),
  datasets: [
    {
      label: "Enrollments",
      data: filteredReport.map(r => r.enrollments),
      backgroundColor: "#6f42c1"
    },
    {
      label: "Avg Score (%)",
      data: filteredReport.map(r => r.avgScore),
    }
  ]
};

```

```

        backgroundColor: "#ffc107"
    }
]
};

const chartOptions = {
    responsive: true,
    plugins: {
        legend: { position: "top" },
        title: { display: true, text: "Course Performance Chart" }
    },
    scales: {
        y: { beginAtZero: true }
    }
};

return (
    <>
    <h2 style={{ color: "#6f42c1" }}>Course Performance</h2>

    /* Chart */
    <div className="bg-white p-3 rounded mb-3">
        <Bar data={chartData} options={chartOptions} />
    </div>

    /* Filters */
    <div className="filters mb-3">
        <input
            type="text"
            placeholder="Search by course"
            value={search}
            onChange={e => setSearch(e.target.value)}
            style={{ marginRight: "10px", padding: "5px" }}
        />
        <input
            type="number"
            placeholder="Min enrollments"
            value={minEnroll}
            onChange={e => setMinEnroll(Number(e.target.value))}
            style={{ marginRight: "10px", padding: "5px", width: "120px" }}
        />
        <button onClick={downloadCSV} className="btn btn-sm btn-primary">Download
        CSV</button>
    </div>

```

```

/* Table */
<div className="bg-white p-3 rounded">
  <table className="table table-hover">
    <thead>
      <tr>
        <th onClick={() => sortData("title")}>{`Course ${sortKey === "title" ? (sortOrder === "asc" ? "▲" : "▼") : ""}`}</th>
        <th onClick={() => sortData("enrollments")}>{`Enrollments ${sortKey === "enrollments" ? (sortOrder === "asc" ? "▲" : "▼") : ""}`}</th>
        <th onClick={() => sortData("avgScore")}>{`Avg Score ${sortKey === "avgScore" ? (sortOrder === "asc" ? "▲" : "▼") : ""}`}</th>
      </tr>
    </thead>
    <tbody>
      {paginatedReport.map(r => {
        let bgColor = r.avgScore >= 80 ? "#d4edda" : r.avgScore >= 50 ? "#fff3cd" : "#f8d7da";
        return (
          <tr key={r._id}>
            <td>{r.title}</td>
            <td>{r.enrollments}</td>
            <td>
              {r.avgScore.toFixed(1)}%
              <div style={{ background: "#e9ecef", height: "6px", borderRadius: "3px", marginTop: "3px" }}>
                <div style={{
                  width: `${r.avgScore}%`,
                  background: "#6f42c1",
                  height: "6px",
                  borderRadius: "3px"
                }}/>
              </div>
            </td>
          </tr>
        );
      )})
    </tbody>
  </table>

/* Pagination */
<div className="pagination mt-2">

```

```

        <button disabled={page === 1} onClick={() => setPage(page - 1)}>Prev</button>
        <span style={{ margin: "0 10px" }}>{page} / {totalPages}</span>
        <button disabled={page === totalPages} onClick={() => setPage(page + 1)}>Next</button>
      </div>
    </div>
  </>
);
}

```

Contact Messages Page

The screenshot shows a dashboard interface with a sidebar on the left and a main content area on the right.

Sidebar (Dashboard):

- Users
- Courses
- Categories
- Payments & Revenue
- Reports & Analytics
- Support (selected)
- Contact Messages

Main Content Area (Contact Messages):

Contact Messages

All messages received from users

Name	Email	Subject	Message	Status	Actions
user2	user2@gmail.com	Contact Form Message	thjjmnghnjk	Resolved	<button>View</button> <button>Delete</button>
user1	user@gmail.com	Contact Form Message	for test..	Pending	<button>View</button> <button>Resolve</button> <button>Delete</button>

contactMessages.jsx

```

import { useEffect, useState } from "react";
import api from "../../api/api";

export default function AdminContactMessages() {
  const [messages, setMessages] = useState([]);
  const [loading, setLoading] = useState(true);

  const fetchMessages = async () => {
    try {
      const res = await api.get("/contact");
      setMessages(res.data.data || []);
    } catch (err) {
      console.log("Error:", err);
      setMessages([]);
    }
  }

  useEffect(() => {
    fetchMessages();
  }, []);
}

```

```
        }
        setLoading(false);
    };

useEffect(() => {
    fetchMessages();
}, []);

const resolveMessage = async (id) => {
    await api.put(`/contact/${id}`);
    setMessages((prev) =>
        prev.map((m) => (m._id === id ? { ...m, status: "Resolved" } : m))
    );
};

const deleteMessage = async (id) => {
    if (!window.confirm("Delete this message?")) return;
    await api.delete(`/contact/${id}`);
    setMessages((prev) => prev.filter((m) => m._id !== id));
};

return (
    <div
        style={{
            padding: "30px",
            background: "#faf7ff",
            minHeight: "100vh",
        }}
    >
        /* HEADER */
        <div
            style={{
                background: "#f3e8ff",
                padding: "25px",
                borderRadius: "15px",
                boxShadow: "0 0 18px rgba(0,0,0,0.08)",
                marginBottom: "25px",
            }}
        >
```

```

    textAlign: "center",
  }}
>
<h1 style={{ fontWeight: "700", color: "#6a0dad" }}>
  Contact Messages
</h1>
<p style={{ opacity: 0.8 }}>All messages received from users</p>
</div>

/* TABLE CARD */
<div
  style={{
    background: "white",
    padding: "25px",
    borderRadius: "12px",
    boxShadow: "0 0 18px rgba(0,0,0,0.08)",
  }}
>
{loading ? (
  <p className="text-center fw-bold">Loading messages...</p>
) : messages.length === 0 ? (
  <p className="text-center">No messages found</p>
) : (
  <div className="table-responsive">
    <table
      style={{
        width: "100%",
        borderCollapse: "separate",
        borderSpacing: "0 10px",
      }}
    >
      <thead>
        <tr style={{ background: "#eee" }}>
          <th style={{ padding: "10px" }}>Name</th>
          <th>Email</th>
          <th>Subject</th>
          <th>Message</th>

```

```

<th>Status</th>
<th>Actions</th>
</tr>
</thead>

<tbody>
{messages.map((msg) => (
<tr
  key={msg._id}
  style={{
    background: "white",
    borderRadius: "10px",
    boxShadow: "0 3px 12px rgba(0,0,0,0.05)",
    transition: "0.2s",
  }}
  onMouseEnter={(e) =>
    (e.currentTarget.style.transform = "scale(1.01)")
  }
  onMouseLeave={(e) =>
    (e.currentTarget.style.transform = "scale(1)")
  }
>
<td style={{ padding: "12px", fontWeight: "600" }}>
  {msg.name}
</td>
<td style={{ padding: "12px" }}>{msg.email}</td>
<td style={{ padding: "12px" }}>{msg.subject}</td>

/* MESSAGE SHORTENED */
<td style={{ padding: "12px", maxWidth: "250px" }}>
<span
  title={msg.message}
  style={{
    display: "inline-block",
    whiteSpace: "nowrap",
    overflow: "hidden",
    textOverflow: "ellipsis",
  }}
>

```

```
        maxWidth: "250px",
    }}
>
    {msg.message}
</span>
</td>

/* STATUS */
<td style={{ padding: "12px" }}>
<span
    style={{
        padding: "6px 12px",
        borderRadius: "6px",
        color: "white",
        background:
            msg.status === "Resolved" ? "green" : "orange",
    }}
>
    {msg.status}
</span>
</td>

/* ACTIONS */
<td style={{ padding: "12px" }}>
<button
    onClick={() => alert(msg.message)}
    style={{
        padding: "6px 12px",
        background: "#6a0dad",
        color: "white",
        borderRadius: "6px",
        border: "none",
        marginRight: "6px",
    }}
>
    View
</button>
```

```
{msg.status === "Pending" && (
  <button
    onClick={() => resolveMessage(msg._id)}
    style={{
      padding: "6px 12px",
      background: "#a855f7",
      color: "white",
      borderRadius: "6px",
      border: "none",
      marginRight: "6px",
    }}
  >
  Resolve
  </button>
)})

<button
  onClick={() => deleteMessage(msg._id)}
  style={{
    padding: "6px 12px",
    background: "darkred",
    color: "white",
    borderRadius: "6px",
    border: "none",
  }}
>
Delete
</button>
</td>
</tr>
))}
</tbody>
</table>
</div>
)}
</div>
```

```

    </div>
  );
}

```

About Us Page

The screenshot shows the LearnX platform's About Us page. At the top, there is a purple header bar with the LearnX logo, a search bar, and navigation links for Home, Courses, and a user profile icon. Below the header, the main content area has a light blue background. The title "About Us" is centered at the top of the content area, with the subtitle "Empowering learners & instructors through modern digital education" underneath it. A section titled "Who We Are" follows, containing a brief description of the team's mission to transform online learning. Below this, a section titled "Our Story" includes a paragraph about the company's journey and innovation, followed by a small image of a graduate in a cap and gown. To the right of the story, there is a large image of a graduation ceremony. The next section, "What Our Platform Offers," features four cards: "Smart Learning Tools" (quizzes, assignments, progress tracking, certification & forums), "Instructor Dashboard" (manage courses, track earnings, analyze student performance), "Secure Payments" (fast & safe checkout, invoices, and multiple payment methods), and "HD Video Player" (smooth playback, speed control & bookmarking for easy learning). Finally, the "Why Students & Instructors Love Us" section compares the platform's offerings for both groups.

Aboutus.jsx

```

export default function AboutUs() {
  return (
    <div style={{ minHeight: "100vh", background: "#ffffff", color: "#444" }}>

```

```
/* HEADER */
<div
style={{
  textAlign: "center",
  marginBottom: "40px",
  marginTop: "70px",
  background: "#f3e8ff",
  padding: "35px 15px",
  borderRadius: "18px",
  boxShadow: "0 6px 14px rgba(0,0,0,0.08)",
}}
>
  /* <h1 style={{ fontSize: "42px", fontWeight: "700", margin: 0 }}>About
Us</h1>
  <p style={{ marginTop: "10px", fontSize: "20px", opacity: 0.95 }}>
    Empowering learners & instructors through modern digital education
  </p> */
  <h1 style={{ fontWeight: "700", fontSize: "40px", color: "#7c3aed" }}>
    About Us
  </h1>
  <p style={{ opacity: 0.8, fontSize: "18px" }}>
    Empowering learners & instructors through modern digital education
  </p>
</div>
```

```
/* WHO WE ARE */
<div style={{ padding: "60px 20px", maxWidth: "1200px", margin: "0 auto" }}>
  <h2
    style={{
      textAlign: "center",
      fontWeight: "700",
      marginBottom: "20px",
      color: "#7c3aed",
    }}
  >
```

Who We Are

</h2>

```
<p  
style={{  
  textAlign: "center",  
  fontSize: "19px",  
  maxWidth: "850px",  
  margin: "0 auto",  
  lineHeight: "1.7",  
}}>
```

>

We are a team of passionate educators, developers, and creators on a mission

to transform online learning. Our LMS provides a powerful yet easy-to-use platform that connects learners and instructors worldwide.

</p>

</div>

```
{/* STORY */}
```

```
<div style={{ padding: "60px 20px", maxWidth: "1200px", margin: "0 auto" }}>  
  <div className="row align-items-center">
```

```
    <div className="col-md-6">
```

```
      <h2 style={{ fontWeight: "700", marginBottom: "15px", color: "#7c3aed" }}>
```

>

Our Story

</h2>

```
    <p style={{ fontSize: "18px", lineHeight: "1.7" }}>
```

Our journey began with a belief — education should be accessible for everyone.

Today, we have built a strong learning ecosystem helping thousands gain modern skills.

</p>

```
    <p style={{ fontSize: "18px", lineHeight: "1.7" }}>
```

Fuelled by innovation and powerful technology, we designed an LMS that simplifies

teaching, boosts productivity, and enhances every learner's growth journey.

```
</p>
```

```
</div>
```

```
<div className="col-md-6 text-center mt-4 mt-md-0">
  
</div>
</div>
</div>
```

```
{/* PLATFORM FEATURES */}
```

```
<div style={{ padding: "60px 20px", maxWidth: "1200px", margin: "0 auto" }}>
  <h2
```

```
    style={{
      textAlign: "center",
      fontWeight: "700",
      marginBottom: "30px",
      color: "#7c3aed",
    }}
  >
```

```
    What Our Platform Offers
  </h2>
```

```
<div className="row g-4">
```

```
  {[
```

```

        { title: "Smart Learning Tools", desc: "Quizzes, assignments, progress tracking, certification & forums." },
        { title: "Instructor Dashboard", desc: "Manage courses, track earnings, analyze student performance." },
        { title: "Secure Payments", desc: "Fast & safe checkout, invoices, and multiple payment methods." },
        { title: "HD Video Player", desc: "Smooth playback, speed control & bookmarking for easy learning." },
    ].map((item, idx) => (
        <div key={idx} className="col-12 col-sm-6 col-lg-3">
            <div
                style={{
                    padding: "25px",
                    background: "#ffffff",
                    borderRadius: "15px",
                    border: "2px solid #eee",
                    boxShadow: "0 4px 12px rgba(0,0,0,0.08)",
                    transition: "all 0.3s ease",
                    cursor: "pointer",
                    height: "100%",
                }}
                onMouseEnter={(e) => {
                    e.currentTarget.style.transform = "translateY(-6px)";
                    e.currentTarget.style.borderColor = "#7c3aed";
                }}
                onMouseLeave={(e) => {
                    e.currentTarget.style.transform = "translateY(0)";
                    e.currentTarget.style.borderColor = "#eee";
                }}
            >
                <h5 style={{ fontWeight: "700", color: "#7c3aed", fontSize: "18px" }}>
                    ★ {item.title}
                </h5>
                <p style={{ marginTop: "10px", fontSize: "16px" }}>{item.desc}</p>
            </div>
        </div>
    )))

```

```
</div>
</div>

/* BENEFITS */
<div style={{ padding: "60px 20px", maxWidth: "1200px", margin: "0 auto" }}>
<h2
  style={{
    textAlign: "center",
    fontWeight: "700",
    marginBottom: "30px",
    color: "#7c3aed",
  }}
>
  Why Students & Instructors Love Us
</h2>

<div className="row g-4">

/* Students Box */
<div className="col-md-6 d-flex justify-content-center">
<div
  style={{
    padding: "25px",
    background: "#faf5ff",
    borderRadius: "15px",
    border: "2px solid #e9d5ff",
    width: "97%",
  }}
>
  <h5 style={{ fontWeight: "700", color: "#7c3aed" }}>For Students</h5>
  <ul style={{ marginTop: "15px", fontSize: "18px", lineHeight: "1.8" }}>
    <li>Structured & high-quality courses</li>
    <li>Affordable learning</li>
    <li>Certificate of completion</li>
    <li>Learn at your own pace</li>
  </ul>
</div>
```

```
</div>

/* Instructors Box */
<div className="col-md-6 d-flex justify-content-center">
  <div
    style={{
      padding: "25px",
      background: "#faf5ff",
      borderRadius: "15px",
      border: "2px solid #e9d5ff",
      width: "97%",
    }}
  >
    <h5 style={{ fontWeight: "700", color: "#7c3aed" }}>For Instructors</h5>
    <ul style={{ marginTop: "15px", fontSize: "18px", lineHeight: "1.8" }}>
      <li>High revenue share</li>
      <li>Easy course creation tools</li>
      <li>Real-time analytics</li>
      <li>Community & support</li>
      <li>Zero technical complexity</li>
    </ul>
  </div>
</div>

</div>
</div>

/* VISION */
<div style={{ padding: "60px 20px", textAlign: "center" }}>
  <h2
    style={{
      fontWeight: "700",
      marginBottom: "20px",
      color: "#7c3aed",
    }}
  >
    Our Vision for the Future
  </h2>
</div>
```

```
</h2>
```

```
<p  
style={{  
  fontSize: "19px",  
  maxWidth: "850px",  
  margin: "0 auto",  
  lineHeight: "1.7",  
}}>
```

```
  Our vision is to create the world's most trusted digital learning environment
```

```
  — where anyone can learn new skills and instructors can grow their teaching  
  careers globally.
```

```
  We aim to introduce AI mentorship, advanced analytics, and global  
  instructor collaboration.
```

```
  </p>  
  </div>
```

```
  </div>  
};  
}
```

Contact Us Page

Contact Us

We're here to help! Reach out for support, questions, or feedback.

Send us a Message

Full Name
Enter your name

Email Address
Enter your email

Message
Write your message...

Send Message

Our Contact Details

Email: support@lms.com
Phone: +91 9876543210
Address: Surat, Gujarat, India

Contactus.jsx

```
import { useState } from "react";
import axios from "axios";
import api from "../api/api";
import { toast } from "react-toastify";

export default function ContactUs() {
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    message: ""
  });

  const [loading, setLoading] = useState(false);
  const [msg, setMsg] = useState("");

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
```

```

 setLoading(true);
setMsg("");

try {
  const res = await axios.post("/contact", {
    name: formData.name,
    email: formData.email,
    subject: "Contact Form Message",
    message: formData.message,
  });

  toast.success("Your message has been sent successfully!");
  setFormData({ name: "", email: "", message: "" });
} catch (error) {
  toast.error("Something went wrong. Please try again.");
}

 setLoading(false);
};

return (
  <div style={{ background: "#ffffff", minHeight: "100vh", padding: "60px 20px" }}>

    {/* Header */}
    <div
      style={{
        textAlign: "center",
        marginBottom: "40px",
        background: "#f3e8ff",
        padding: "35px 15px",
        borderRadius: "18px",
        boxShadow: "0 6px 14px rgba(0,0,0,0.08)",
      }}
    >
      <h1 style={{ fontWeight: "700", fontSize: "40px", color: "#7c3aed" }}>
        Contact Us
      </h1>
      <p style={{ opacity: 0.8, fontSize: "18px" }}>
        We're here to help! Reach out for support, questions, or feedback.
      </p>
    </div>

    {/* Main Content */}
    <div className="container">

```

```

<div className="row g-4">

  /* Left Side – Form */
  <div className="col-md-6">
    <div
      style={{
        background: "#faf5ff",
        padding: "25px",
        borderRadius: "15px",
        border: "2px solid #e9d5ff",
        boxShadow: "0 4px 14px rgba(0,0,0,0.08)",
      }}
    >
      <h3 style={{ fontWeight: "700", marginBottom: "20px", color: "#7c3aed" }}>
        Send us a Message
      </h3>

      {msg && (
        <div className="alert alert-info py-2 text-center">
          {msg}
        </div>
      )}

      <form onSubmit={handleSubmit}>
        <div className="mb-3">
          <label className="form-label fw-bold">Full Name</label>
          <input
            type="text"
            name="name"
            className="form-control"
            placeholder="Enter your name"
            value={formData.name}
            onChange={handleChange}
            required
            style={{ padding: "10px" }}
          />
        </div>

        <div className="mb-3">
          <label className="form-label fw-bold">Email Address</label>
          <input
            type="email"
            name="email"
            className="form-control"
          >
        </div>
      </form>
    </div>
  </div>
</div>

```

```

placeholder="Enter your email"
value={formData.email}
onChange={handleChange}
required
style={{ padding: "10px" }}
/>
</div>

<div className="mb-3">
  <label className="form-label fw-bold">Message</label>
  <textarea
    name="message"
    className="form-control"
    rows="4"
    placeholder="Write your message..."
    value={formData.message}
    onChange={handleChange}
    required
    style={{ padding: "10px" }}
  ></textarea>
</div>

<button
  disabled={loading}
  className="btn w-100"
  style={{
    background: "#7c3aed",
    color: "white",
    padding: "12px",
    fontWeight: "600",
    borderRadius: "10px",
    fontSize: "17px",
  }}
>
  {loading ? "Sending..." : "Send Message"}
</button>
</form>
</div>
</div>

/* Right Side – Contact info */
<div className="col-md-6">
  <div
    style={{

```

```

background: "#faf5ff",
padding: "25px",
borderRadius: "15px",
border: "2px solid #e9d5ff",
boxShadow: "0 4px 14px rgba(0,0,0,0.08)",
}}
>
<h3 style={{ fontWeight: "700", marginBottom: "20px", color: "#7c3aed" }}>
  Our Contact Details
</h3>

<ul style={{ listStyle: "none", paddingLeft: 0, fontSize: "17px", lineHeight: "1.9" }}>
  <li><strong>Email:</strong> support@lms.com</li>
  <li><strong>Phone:</strong> +91 9876543210</li>
  <li><strong>Address:</strong> Surat, Gujarat, India</li>
</ul>

/* Map */
<div className="mt-4">
  <iframe
    title="map"
    width="100%"
    height="280"
    style={{
      border: 0,
      borderRadius: "15px",
      boxShadow: "0 4px 14px rgba(0,0,0,0.1)",
    }}
    loading="lazy"
    allowFullScreen
    src="https://maps.google.com/maps?q=Surat&t=&z=13&ie=UTF8&iwloc=&output=embed"
  ></iframe>
</div>
</div>
</div>

</div>
</div>

</div>
);
}

```

Privacy Policy Page

The screenshot shows a purple header bar with the 'LearnX' logo, a search bar, and navigation links for 'Home', 'Courses', and a user profile icon. The main content area has a light blue background with a central white column containing the title 'Privacy Policy' and a sub-section 'Learn how we collect, use, and protect your personal information.' Below this, there are six numbered sections in boxes:

- 1. Data Collection**
We collect essential information such as your name, email, and usage activity to deliver a personalized and effective learning experience.
- 2. How We Use Your Data**
Your information is used to provide services like course personalization, account updates, communication, and platform support.
- 3. Cookies & Tracking**
We use cookies to store preferences, enhance performance, and improve overall navigation across the platform.
- 4. Data Protection & Security**
Your data is stored securely and encrypted where required. We do not share personal information with third parties unless legally required.
- 5. Your Rights**
You may request access, modification, exporting, or deletion of your personal information at any time.
- 6. Policy Updates**
This policy may be updated periodically. Significant changes will be communicated to all users.

privacyPolicy.jsx

```
export default function PrivacyPolicy() {
  return (
    <div className="min-h-screen bg-white py-5">

      {/* HEADER */}
      <div className="container mb-5">
        <div
          className="text-center p-4 p-md-5 rounded shadow-sm"
          style={{
            background: "#f3e8ff",
            borderRadius: "20px",
            padding: "10px 0",
            margin: "0 auto",
            width: "fit-content",
            maxWidth: "300px",
            height: "auto",
            border: "1px solid #d9e1f2",
            boxShadow: "0 4px 8px #d9e1f2"
          }}>
```

```
        boxShadow: "0 6px 20px rgba(0,0,0,0.08)",
    }}
>
<h1 className="fw-bold" style={{ color: "#7c3aed", fontSize: "42px" }}>
    Privacy Policy
</h1>
<p className="mt-2 text-muted" style={{ fontSize: "18px" }}>
    Learn how we collect, use, and protect your personal information.
</p>
</div>
</div>

/* CONTENT SECTIONS */
<div className="container" style={{ maxWidth: "950px" }}>
    <div className="row g-4">

        {[

            {
                title: "1. Data Collection",
                text:
                    "We collect essential information such as your name, email, and usage activity to deliver a personalized and effective learning experience.",
            },
            {
                title: "2. How We Use Your Data",
                text:
                    "Your information is used to provide services like course personalization, account updates, communication, and platform support.",
            },
            {
                title: "3. Cookies & Tracking",
                text:
                    "We use cookies to store preferences, enhance performance, and improve overall navigation across the platform.",
            },
            {
                title: "4. Data Protection & Security",
                text:
                    "Your data is stored securely and encrypted where required. We do not share personal information with third parties unless legally required.",
            },
            {
                title: "5. Your Rights",
                text:

```

"You may request access, modification, exporting, or deletion of your personal information at any time.",

```
  },
  {
    title: "6. Policy Updates",
    text:
      "This policy may be updated periodically. Significant changes will be communicated to all users.",
    },
    ].map((item, index) => (
      <div className="col-12" key={index}>
        <div
          className="p-4 rounded shadow-sm h-100"
          style={{
            background: "#faf5ff",
            borderLeft: "6px solid #7c3aed",
            lineHeight: "1.7",
          }}
        >
          <h4 className="fw-bold mb-2" style={{ color: "#7c3aed" }}>
            {item.title}
          </h4>
          <p className="text-secondary">{item.text}</p>
        </div>
      </div>
    )));
  </div>
</div>
</div>
);
}
```

Terms & Conditions Page

The screenshot shows a purple-themed learning platform interface. At the top, there's a navigation bar with 'LearnX' on the left, a search bar in the center containing 'Search courses...', and 'Home Courses' on the right with a user icon. Below the navigation is a main content area with a light blue header titled 'Terms & Conditions'. A sub-header below it says 'Please review these terms carefully before accessing or using our platform.' The main content is organized into six sections, each with a purple border:

- 1. Acceptance of Terms**

By using our platform, you agree to follow all rules, policies, and guidelines listed here. If you disagree with any part, please stop using the platform immediately.
- 2. Course Access**

Course access is provided only after successful payment. Each course has its own validity period depending on instructor settings.
- 3. Refund Policy**

Refunds are granted only after admin review in genuine cases like duplicate payments or technical issues. Completed courses or downloaded files are not refundable.
- 4. Instructor Guidelines**

Instructors must upload original content and follow platform rules. Any copyright violation may result in content removal or account suspension.
- 5. Account Termination**

We may suspend or terminate accounts involved in misuse, fraud, or violations of platform rules.
- 6. Updates to Terms**

These terms may be updated regularly. Continued use of the platform means you accept the latest version.

Terms.jsx

```
export default function Terms() {
  return (
    <div className="min-h-screen bg-white py-5">

      {/* HEADER */}
      <div className="container mb-5">
        <div
          className="text-center p-4 p-md-5 rounded shadow-sm"
          style={{
            background: "#f3e8ff",

```

```

borderRadius: "20px",
  boxShadow: "0 6px 20px rgba(0,0,0,0.08)",
}
>
<h1 className="fw-bold" style={{ color: "#7c3aed", fontSize: "42px" }}>
  Terms & Conditions
</h1>
<p className="mt-2 text-muted" style={{ fontSize: "18px" }}>
  Please review these terms carefully before accessing or using our platform.
</p>
</div>
</div>

{/* TERMS SECTIONS */}
<div className="container" style={{ maxWidth: "950px" }}>
  <div className="row g-4">

    {/* CARD TEMPLATE */}
    {[{
      {
        title: "1. Acceptance of Terms",
        text:
          "By using our platform, you agree to follow all rules, policies, and guidelines listed here. If you disagree with any part, please stop using the platform immediately.",
      },
      {
        title: "2. Course Access",
        text:
          "Course access is provided only after successful payment. Each course has its own validity period depending on instructor settings.",
      },
      {
        title: "3. Refund Policy",
        text:
          "Refunds are granted only after admin review in genuine cases like duplicate payments or technical issues. Completed courses or downloaded files are not refundable.",
      },
      {
        title: "4. Instructor Guidelines",
        text:
          "Instructors must upload original content and follow platform rules. Any copyright violation may result in content removal or account suspension.",
      },
    ]}
  </div>
</div>

```

```
        title: "5. Account Termination",
        text:
          "We may suspend or terminate accounts involved in misuse, fraud, or violations of
platform rules.",
      },
      {
        title: "6. Updates to Terms",
        text:
          "These terms may be updated regularly. Continued use of the platform means you
accept the latest version.",
      },
    ].map((item, index) => (
      <div className="col-12" key={index}>
        <div
          className="p-4 rounded shadow-sm h-100"
          style={{
            background: "#faf5ff",
            borderLeft: "6px solid #7c3aed",
            lineHeight: "1.7",
          }}
        >
          <h4 className="fw-bold mb-2" style={{ color: "#7c3aed" }}>
            {item.title}
          </h4>
          <p className="text-secondary">{item.text}</p>
        </div>
      </div>
    )));
  </div>
</div>
</div>
);
}
```

Support and Help Center Page

The screenshot shows a purple-themed LMS interface. At the top, there's a navigation bar with 'LearnX' on the left, a search bar in the center containing 'Search courses...', and 'Home Courses' on the right with a user icon. Below the navigation is a main content area titled 'Support & Help Center'. A section titled 'How can we assist you?' contains text about 24/7 support for course access, payments, and account issues. It includes three contact options: 'Email Support' (support@lms.com), 'Phone' (+91 9876543210), and 'Helpdesk' (www.lmissupport.com). A 'Common Issues' section lists several troubleshooting points.

Support.jsx

```
export default function Support() {
  return (
    <div className="min-h-screen bg-white py-5">
      <div className="container">

        {/* TITLE */}
        <h1 className="text-center fw-bold mb-4" style={{ color: "#6a0dad", marginTop:"50px" }}>
          Support & Help Center
        </h1>

        <div
          className="mx-auto p-4 p-md-5 shadow-lg rounded-4"
          style={{ maxWidth: "850px", background: "#f8f3ff" }}
        >
          {/* Intro */}
          <h4 className="fw-bold" style={{ color: "#6a0dad" }}>
            How can we assist you?
          </h4>
          <p className="text-muted">
            We're available 24/7 to support your learning journey.
            Whether it's course access, payments, instructor queries, or account issues —
            we're here to help!
          </p>
        </div>
      </div>
    </div>
  );
}
```

```

</p>

/* Contact Cards */


###### Email Support



support@lms.com



###### Phone



+91 9876543210



###### Helpdesk



www.lmssupport.com


/* FAQ / Common Issues */


#### Common Issues



- Course not loading or video buffering issues

```

```

<li>Payment completed but course not unlocked</li>
<li>Instructor content approval pending</li>
<li>Login problems or password reset help</li>
<li>Profile or enrollment details not updating</li>
</ul>
</div>
</div>
</div>
);
}

```

CSS

Home.css

```

/* ===== GENERAL ===== */
.home-container {
  margin-top: 60px;
  font-family: "Poppins", sans-serif;
  color: #333;
  line-height: 1.6;
}

/* ===== HERO ===== */
.hero-section {
  position: relative;
  background-image: url("https://images.unsplash.com/photo-1504384308090-c894fdcc538d?auto=format&fit=crop&w=1600&q=80");
  background-size: cover;
  background-position: center;
  color: #fff;
  padding: 100px 20px 60px;
  text-align: center;
}

.hero-section::before {
  content: "";
  position: absolute;
  inset: 0;
  background: rgba(0, 0, 0, 0.5);
  z-index: 0;
}

```

```
.hero-section * {  
  position: relative;  
  z-index: 1;  
}  
  
.hero-title {  
  font-size: 2.8rem;  
  font-weight: 700;  
}  
  
.hero-title span {  
  color: #ffcc00;  
}  
  
.hero-subtitle {  
  margin-top: 15px;  
  font-size: 1.3rem;  
  opacity: 0.9;  
}  
  
.explore-btn {  
  background-color: #ffcc00;  
  color: #222;  
  padding: 12px 30px;  
  border-radius: 50px;  
  font-weight: 600;  
  margin-top: 25px;  
  transition: all 0.3s;  
}  
  
.explore-btn:hover {  
  transform: translateY(-3px);  
  box-shadow: 0 8px 20px rgba(0,0,0,0.2);  
}  
  
/* ===== WHY CHOOSE ===== */  
.why-choose-section {  
  padding: 60px 20px;  
  background: #f9f9ff;  
  text-align: center;  
}  
  
.features-grid {
```

```
display: grid;
grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));
gap: 25px;
margin-top: 35px;
}

.feature-card {
  padding: 25px 15px;
  border-radius: 12px;
  box-shadow: 0 6px 20px rgba(0,0,0,0.05);
  transition: all 0.3s;
  text-align: center;
  background-color: #fff;
}

.feature-card:hover {
  transform: translateY(-6px);
  box-shadow: 0 10px 25px rgba(0,0,0,0.1);
}

.feature-card i {
  font-size: 30px;
  margin-bottom: 12px;
  color: #9f64f7;
}

.feature-card h4 {
  font-size: 1.15rem;
  margin-bottom: 10px;
}

.feature-card p {
  font-size: 0.9rem;
  color: #555;
}

/* ===== CATEGORIES ===== */
.categories-section {
  padding: 60px 20px;
  background: #f8f8ff;
  text-align: center;
}

.categories-grid {
```

```
display: grid;
grid-template-columns: repeat(auto-fit, minmax(160px, 1fr));
gap: 20px;
margin-top: 30px;
}

.category-card {
padding: 18px 12px;
border-radius: 12px;
box-shadow: 0 5px 15px rgba(0,0,0,0.08);
cursor: pointer;
background-color: #fff;
transition: all 0.3s;
text-align: center;
}

.category-card:hover {
transform: translateY(-5px);
box-shadow: 0 10px 20px rgba(0,0,0,0.12);
}

/* ===== COURSES ===== */
.courses-section {
padding: 60px 20px;
}

.category-title {
font-size: 1.8rem;
font-weight: 700;
margin-bottom: 20px;
}

.courses-wrapper {
position: relative;
overflow: hidden;
}

.courses-scroll {
display: flex;
gap: 20px;
padding: 10px 0;
overflow-x: auto;
scroll-behavior: smooth;
}
```

```
.scroll-card {  
    min-width: 250px;  
    flex: 0 0 auto;  
}  
  
.scroll-btn {  
    position: absolute;  
    top: 50%;  
    transform: translateY(-50%);  
    background-color: rgba(159,100,247,0.9);  
    border: none;  
    color: #fff;  
    font-size: 1.5rem;  
    width: 40px;  
    height: 40px;  
    border-radius: 50%;  
    cursor: pointer;  
    z-index: 2;  
    display: flex;  
    align-items: center;  
    justify-content: center;  
}  
  
.scroll-btn.left {  
    left: -20px;  
}  
  
.scroll-btn.right {  
    right: -20px;  
}  
  
.scroll-btn:hover {  
    background-color: #7b45c5;  
}  
  
.course-card {  
    border-radius: 14px;  
    overflow: hidden;  
    transition: all 0.3s;  
    cursor: pointer;  
    box-shadow: 0 6px 18px rgba(0,0,0,0.06);  
    background-color: #fff;  
}
```

```
.course-card:hover {  
    transform: translateY(-6px);  
    box-shadow: 0 12px 25px rgba(0,0,0,0.12);  
}  
  
.course-image img {  
    width: 100%;  
    height: 160px;  
    object-fit: cover;  
    transition: transform 0.3s;  
}  
  
.course-card:hover .course-image img {  
    transform: scale(1.05);  
}  
  
.course-info {  
    padding: 15px;  
}  
  
.course-title a {  
    font-size: 1rem;  
    font-weight: 600;  
    color: #333;  
    text-decoration: none;  
}  
  
.course-title a:hover {  
    color: #9f64f7;  
}  
  
.course-description {  
    font-size: 0.85rem;  
    margin: 10px 0;  
}  
  
.course-meta {  
    display: flex;  
    justify-content: space-between;  
    align-items: center;  
}  
  
.level {
```

```
padding: 5px 12px;
font-size: 0.8rem;
border-radius: 15px;
background: #eae2ff;
color: #6b42c8;
}

.price {
font-weight: 600;
font-size: 0.95rem;
color: #9f64f7;
}

/* ===== ABOUT ===== */
.about-section {
padding: 80px 20px;
background-color: #fdfdfd;
}

.about-grid {
display: flex;
align-items: center;
gap: 40px;
max-width: 1200px;
margin: 0 auto;
flex-wrap: wrap;
}

.about-image {
flex: 1 1 400px;
display: flex;
justify-content: center;
}

.about-image img {
width: 100%;
max-width: 450px;
border-radius: 20px;
box-shadow: 0 10px 25px rgba(0,0,0,0.1);
transition: transform 0.3s ease, box-shadow 0.3s ease;
}

.about-image img:hover {
transform: scale(1.03);
```

```
    box-shadow: 0 15px 30px rgba(0,0,0,0.15);  
}
```

```
.about-text {  
  flex: 1 1 400px;  
  max-width: 600px;  
}
```

```
.about-text h2 {  
  font-size: 2.2rem;  
  font-weight: 700;  
  margin-bottom: 20px;  
}
```

```
.about-text h2 span {  
  color: #9f64f7;  
}
```

```
.about-text p {  
  font-size: 1rem;  
  color: #555;  
  margin-bottom: 25px;  
  line-height: 1.7;  
}
```

```
.learn-more-btn {  
  padding: 12px 30px;  
  border-radius: 30px;  
  font-weight: 600;  
  background-color: #9f64f7;  
  color: #fff;  
  border: none;  
  cursor: pointer;  
  transition: all 0.3s ease;  
}
```

```
.learn-more-btn:hover {  
  background-color: #7b45c5;  
  transform: translateY(-2px);  
}
```

```
/* ===== RESPONSIVE ===== */  
@media (max-width: 992px) {  
  .hero-title {
```

```
    font-size: 2.4rem;
}
.hero-subtitle {
    font-size: 1.1rem;
}
.features-grid,
.categories-grid {
    gap: 20px;
}
.about-grid {
    gap: 30px;
}
.about-text h2 {
    font-size: 2rem;
}
}

@media (max-width: 768px) {
.about-grid {
    flex-direction: column;
    text-align: center;
}
.about-text {
    max-width: 100%;
}
.about-text h2 {
    font-size: 1.8rem;
}
.about-text p {
    font-size: 0.95rem;
}
.about-image img {
    max-width: 90%;
}
.courses-scroll {
    padding-bottom: 10px;
}
}

@media (max-width: 576px) {
.hero-title {
    font-size: 1.8rem;
}
.hero-subtitle {
```

```
    font-size: 1rem;
}
.course-description {
    font-size: 0.8rem;
}
.feature-card h4 {
    font-size: 1rem;
}
.feature-card p {
    font-size: 0.85rem;
}
.scroll-card {
    min-width: 200px;
}
}
```

Form.css

```
body {
    margin: 0;
    padding: 0;
    font-family: Arial, sans-serif;
    /*background: url("https://cdn.pixabay.com/photo/2018/08/04/11/30/draw-3583548_1280.png");*/
}

.register-container,
.login-container {
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    width: 100%;
}

.register-card,
.login-card {
    border-radius: 10px;
    padding: 2rem;
    width: 500px;
    max-width: 95%;
    background-color: #fff;
    box-shadow: 0 4px 15px rgba(0, 0, 0, 0.2);
```

```
}

.register-card h2,
.login-card h1 {
  text-align: center;
  margin-bottom: 1.5rem;
  font-weight: 600;
  font-size: 27px;
}

.form-input {
  display: flex;
  align-items: center;
  flex-wrap: wrap;
  margin-bottom: 1rem;
}

.form-input label {
  width: 150px;
  font-weight: 500;
  margin-right: 10px;
}

.form-input input {
  flex: 1;
  padding: 6px 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  outline: none;
  font-size: 14px;
}

.form-input input[type="file"] {
  width: 60%;
}

.form-input input.input-error {
  border-color: red;
}

.form-input .error {
  width: 100%;
  color: red;
  font-size: 0.85rem;
```

```
margin: 4px 0 0 160px;
display: block;
text-align: left;
}

.register-card input::placeholder,
.login-card input::placeholder {
color: #aaa;
}
.d-flex {
display: flex;
justify-content: space-between;
gap: 15px;
}

.register-card button,
.login-card button {
flex: 1;
border-radius: 6px;
padding: 5px;
cursor: pointer;
font-weight: 500;
transition: 0.3s;
}

.login-card button[type="submit"]{
width: 200px;
}

.register-card button[type="submit"],
.login-card button[type="submit"] {
background-color: #953ce7;
color: #fff;
}

.register-card button[type="reset"]{
background-color: #6c757d;
color: #fff;
}

.register-card button:hover,
.login-card button:hover {
opacity: 0.9;
}
```

```
.role-buttons {  
    display: flex;  
    justify-content: center;  
    gap: 16px;  
    margin-bottom: 1.6rem;  
}  
  
.role-buttons button {  
    flex: 1;  
    padding: 7px;  
    border-radius: 6px;  
    border: 1px solid #9a9393;  
    background-color: #fff;  
    cursor: pointer;  
    transition: 0.3s;  
    border-bottom: none;  
    border-bottom-left-radius: 0;  
    border-bottom-right-radius: 0;  
}  
  
.role-buttons button.active {  
    background-color: #5d0dfd;  
    color: #fff;  
    border-color: #5d0dfd;  
    border-bottom: none;  
}  
  
.link {  
    text-align: right;  
    margin-top: 6%;  
    margin-bottom: -3%;  
}
```

Profile.css

```
.profile-simple-container {  
    max-width: 600px;  
    margin: 90px auto 40px auto;  
    padding: 25px;  
    border-radius: 10px;  
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.10);  
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
```

```
}

.profile-title {
    text-align: center;
    margin-bottom: 20px;
    font-size: 24px;
    color: #333;
}

.profile-buttons {
    display: flex;
    justify-content: center;
    gap: 10px;
    margin-bottom: 30px;
}

.profile-buttons button {
    padding: 5px 16px;
    border: 1px solid #aaa;
    background-color: #fff;
    color: #555;
    border-radius: 6px;
    cursor: pointer;
    transition: all 0.2s;
    font-weight: 500;
    padding-left: 40px;
    padding-right: 40px;
}

.profile-buttons button.active {
    border-color: #a82beb;
    background-color: #e6f0ff;
    color: #9a34d4;
}

.profile-buttons button:hover:not(.active) {
    background-color: #f0f0f0;
}

.profile-section h4 {
    margin-bottom: 15px;
    color: #444;
    font-size: 18px;
    border-bottom: 1px solid #ddd;
```

```
padding-bottom: 5px;
}

.profile-section p {
  margin: 8px 0;
  color: #555;
}

.input-group-simple {
  display: flex;
  flex-direction: column;
  margin-bottom: 15px;
}

.input-group-simple label {
  margin-bottom: 5px;
  font-weight: 500;
  color: #333;
}

.input-group-simple input {
  padding: 8px 10px;
  border: 1px solid #ccc;
  border-radius: 6px;
  font-size: 14px;
  outline: none;
  transition: border-color 0.2s;
}

.input-group-simple input:focus {
  border-color: #8c00ff;
}

.save-btn-simple {
  padding: 10px 18px;
  background-color: #8000ff;
  color: #fff;
  border: none;
  border-radius: 6px;
  cursor: pointer;
  font-weight: 500;
  transition: background-color 0.2s;
}
```

```
.save-btn-simple:hover {  
    background-color: #801ad4;  
}  
  
.profile-loading {  
    text-align: center;  
    font-size: 16px;  
    color: #555;  
}  
  
.profile-pic {  
    width: 100px;  
    height: 100px;  
    object-fit: cover;  
    border-radius: 50%;  
    border: 2px solid #6D28D9;  
    margin-bottom: 10px;  
}
```

myLearnings.css

```
/* === Progress Bar === */  
.ml-progress-wrapper {  
    margin-top: 10px;  
    width: 100%;  
}  
  
.ml-progress-bg {  
    width: 100%;  
    height: 10px;  
    background-color: #eee;  
    border-radius: 10px;  
    overflow: hidden;  
}  
  
.ml-progress-fill {  
    height: 100%;  
    border-radius: 10px;  
    transition: width 0.8s ease-in-out, background-color 0.8s ease-in-out;  
}  
  
/* Text beside bar */  
.ml-progress-text {
```

```
display: block;
margin-top: 5px;
font-size: 0.9rem;
color: #555;
font-weight: 500;
}

/* Container */
.ml-container {
padding: 30px;
max-width: 1100px;
margin: auto;
}

.ml-container h2 {
color: #5c2a9d;
margin-bottom: 25px;
font-size: 2rem;
}

/* Courses list */
.ml-courses-list {
display: flex;
flex-direction: column;
gap: 20px;
}

/* Course card */
.ml-course-card {
display: flex;
background-color: #fff;
border-radius: 12px;
overflow: hidden;
box-shadow: 0 4px 12px rgba(0,0,0,0.08);
transition: transform 0.2s ease, box-shadow 0.2s ease;
}

.ml-course-card:hover {
transform: translateY(-3px);
box-shadow: 0 8px 20px rgba(0,0,0,0.12);
}

/* Thumbnail */
.ml-course-thumbnail {
```

```
width: 250px;
height: 150px;
background-size: cover;
background-position: center;
flex-shrink: 0;
}

/* Details */
.ml-course-details {
padding: 15px 20px;
display: flex;
flex-direction: column;
justify-content: space-between;
}

.ml-course-details h3 {
margin: 0 0 10px 0;
color: #5c2a9d;
font-size: 1.3rem;
}

.ml-course-details p {
color: #555;
font-size: 0.95rem;
margin-bottom: 15px;
}

/* Progress bar */
.ml-progress-wrapper {
display: flex;
align-items: center;
gap: 10px;
margin-bottom: 15px;
}

.ml-progress-bg {
flex-grow: 1;
background-color: #e0e0e0;
height: 10px;
border-radius: 10px;
overflow: hidden;
}

.ml-progress-fill {
```

```
height: 10px;
background-color: #5c2a9d;
border-radius: 10px;
transition: width 0.4s ease;
}

.ml-progress-text {
font-size: 0.85rem;
color: #333;
min-width: 50px;
}

/* Actions */

.ml-course-actions {
gap: 10px;
}

.ml-course-actions button {
flex: 1;
background-color: #7b41c7;
color: white;
border: none;
padding: 5px 12px;
border-radius: 6px;
cursor: pointer;
font-weight: 600;
transition: background-color 0.3s ease;
}

.ml-course-actions button:hover {
background-color: #7b3fc0;
}

/* Loading / error / info */

.ml-loading-text,
.ml-error-text,
.ml-info-text {
text-align: center;
margin-top: 50px;
color: #5c2a9d;
font-size: 1.1rem;
}

/* Spinner */
```

```
.ml-course-loading {  
    display: flex;  
    flex-direction: column;  
    align-items: center;  
    justify-content: center;  
    margin-top: 60px;  
}  
  
.ml-course-loading .ml-spinner {  
    width: 40px;  
    height: 40px;  
    border: 5px solid #eee;  
    border-top-color: #5c2a9d;  
    border-radius: 50%;  
    animation: ml-spin 1s linear infinite;  
    margin-bottom: 15px;  
}  
  
@keyframes ml-spin {  
    to { transform: rotate(360deg); }  
}
```

lessons.css

```
.lesson-page {  
    display: flex;  
    flex-direction: row;  
    font-family: Arial, sans-serif;  
    background: #f9f7fc;  
    min-height: 100vh;  
    padding: 20px;  
    gap: 20px;  
    margin-top: 70px;  
}  
  
/* Sidebar with all lessons */  
.lesson-list-sidebar {  
    width: 25%;  
    background: #fff;  
    border-radius: 12px;  
    padding: 15px;  
    box-shadow: 0 6px 12px rgba(128, 0, 128, 0.15);  
    max-height: 80vh;
```

```
    overflow-y: auto;
}

.lesson-list-sidebar h3 {
    margin-bottom: 10px;
    font-size: 1.1rem;
    color: #4b0082; /* dark purple */
    font-weight: bold;
}

.lesson-list-sidebar ul {
    list-style: none;
    padding: 0;
    margin: 0;
}

.lesson-list-sidebar li {
    padding: 10px 12px;
    border-radius: 8px;
    cursor: pointer;
    margin-bottom: 6px;
    font-size: 0.95rem;
    transition: background 0.2s, color 0.2s;
    color: #4b0082;
}

.lesson-list-sidebar li:hover {
    background: #f3e6ff;
}

.lesson-list-sidebar li.active {
    background: #6a0dad; /* purple */
    color: #fff;
    font-weight: bold;
}

.lesson-list-sidebar li.completed {
    font-weight: bold;
    color: #7fcf00; /* green for completed */
}

.lesson-list-sidebar li.locked {
    color: #aaa;
    cursor: not-allowed;
```

```
}

/* Preview badge */
.preview-locked {
    background: #d8b6ff;
    color: #4b0082;
    padding: 2px 6px;
    border-radius: 6px;
    font-size: 0.75rem;
    margin-left: 8px;
    font-weight: bold;
}

/* Main lesson content */
.lesson-content {
    flex: 1;
    background: #fff;
    border-radius: 12px;
    padding: 20px;
    box-shadow: 0 6px 12px rgba(128, 0, 128, 0.1);
    max-height: 80vh;
    overflow-y: auto;
}

/* Video player */
.lesson-video {
    width: 100%;
    max-height: 450px;
    border-radius: 10px;
    margin-bottom: 15px;
    background: #000;
    border: 2px solid #6a0dad;
}

/* PDF/iframe */
.lesson-pdf {
    width: 100%;
    height: 450px;
    border-radius: 10px;
    border: 2px solid #6a0dad;
    margin-bottom: 15px;
}

/* Header with lesson title */
```

```
.lesson-header {  
    display: flex;  
    align-items: center;  
    justify-content: space-between;  
    margin-bottom: 15px;  
}
```

```
.lesson-header h2 {  
    font-size: 1.3rem;  
    margin: 0;  
    color: #4b0082;  
}
```

```
/* Back button */  
.back-button {  
    padding: 6px 12px;  
    margin-left: 10px;  
    border: none;  
    border-radius: 8px;  
    background: #6a0dad;  
    color: #fff;  
    cursor: pointer;  
    font-size: 0.9rem;  
    font-weight: bold;  
}
```

```
.back-button:hover {  
    background: #4b0082;  
}
```

```
/* Navigation buttons */  
.lesson-navigation {  
    display: flex;  
    justify-content: space-between;  
    margin-top: 20px;  
}
```

```
.lesson-navigation button {  
    padding: 8px 16px;  
    border: none;  
    border-radius: 8px;  
    cursor: pointer;  
    font-size: 0.95rem;  
    font-weight: bold;
```

```
background: #6a0dad;
color: #fff;
transition: 0.2s;
}

.lesson-navigation button:hover:enabled {
background: #4b0082;
}

.lesson-navigation button:disabled {
background: #ccc;
cursor: not-allowed;
}

/* Scrollbar styling */
.lesson-list-sidebar::-webkit-scrollbar {
width: 8px;
}

.lesson-list-sidebar::-webkit-scrollbar-track {
background: #f9f7fc;
}

.lesson-list-sidebar::-webkit-scrollbar-thumb {
background: #6a0dad;
border-radius: 4px;
}

/* Responsive */
@media (max-width: 1024px) {
.lesson-page {
flex-direction: column;
}

.lesson-list-sidebar {
width: 100%;
max-height: auto;
margin-bottom: 20px;
}

.lesson-content {
max-height: auto;
}
}
```

```
.exam-page-container {  
    display: flex;  
    gap: 20px;  
    padding: 20px;  
}  
  
.exam-sidebar {  
    width: 250px;  
    background: #f9f9f9;  
    border-radius: 12px;  
    padding: 15px;  
    box-shadow: 0 0 8px rgba(0,0,0,0.1);  
}  
  
.exam-sidebar h3 {  
    margin-bottom: 10px;  
    font-size: 18px;  
    color: #5c2a9d;  
}  
  
.exam-sidebar ul li {  
    padding: 8px 10px;  
    margin-bottom: 5px;  
    cursor: pointer;  
    border-radius: 6px;  
}  
  
.exam-sidebar ul li.active {  
    background-color: #eeeeee;  
    color: black;  
    font-weight: bold;  
}  
  
.exam-content {  
    flex: 1;  
    background: white;  
    padding: 20px;  
    border-radius: 12px;  
    box-shadow: 0 0 10px rgba(0,0,0,0.1);  
}  
  
.exam-header {  
    display: flex;
```

```
justify-content: space-between;  
align-items: center;  
margin-bottom: 20px;  
}
```

```
.exam-question-card {  
background: #f1f0f5;  
padding: 15px;  
border-radius: 10px;  
margin-bottom: 15px;  
}
```

```
.options-list {  
list-style: none;  
padding-left: 0;  
}
```

```
.option-label {  
display: flex;  
align-items: center;  
gap: 10px;  
cursor: pointer;  
}
```

```
.submit-button {  
background: green;  
color: white;  
padding: 7px 20px;  
border-radius: 8px;  
margin-top: 10px;  
border: none;  
cursor: pointer;  
}
```

```
.submit-button:hover {  
background: #1a6803;  
}
```

```
.exam-navigation {  
display: flex;  
justify-content: space-between;  
margin-top: 20px;  
}
```

```
.exam-result {  
    text-align: center;  
    padding: 30px;  
    background: #f0f9ff;  
    border-radius: 12px;  
}  
  
.exam-result .score {  
    font-size: 22px;  
    font-weight: bold;  
    color: #5c2a9d;  
}  
  
.exam-navigation {  
    display: flex;  
    justify-content: space-between;  
    margin-top: 20px;  
    gap: 10px;  
}  
  
.exam-navigation button {  
    background-color: #5c2a9d; /* purple theme */  
    color: white;  
    border: none;  
    border-radius: 8px;  
    padding: 7px 20px;  
    cursor: pointer;  
    font-weight: 600;  
    transition: background-color 0.2s ease;  
}  
  
.exam-navigation button:disabled {  
    background-color: #ccc;  
    cursor: not-allowed;  
}  
  
.exam-navigation button:hover:not(:disabled) {  
    background-color: #771cdf;  
}
```

addCourse.css

```
.add-course-page {  
    display: flex;
```

```
justify-content: center;
align-items: flex-start;
padding: 40px 20px;
background-color: #f4f4f9;
min-height: 100vh;
}

.add-course-card {
background-color: #fff;
padding: 30px 25px;
border-radius: 12px;
box-shadow: 0 8px 20px rgba(0, 0, 0, 0.1);
width: 100%;
max-width: 500px;
}

.add-course-title {
text-align: center;
color: #6f42c1;
margin-bottom: 20px;
font-size: 1.8rem;
}

.add-course-form label {
display: block;
margin-bottom: 6px;
margin-top: 12px;
font-weight: 500;
color: #333;
}

.input-field {
width: 100%;
padding: 10px 12px;
border: 1px solid #ccc;
border-radius: 6px;
margin-bottom: 10px;
font-size: 1rem;
outline: none;
transition: all 0.2s ease;
}

.input-field:focus {
border-color: #6f42c1;
```

```
        box-shadow: 0 0 5px rgba(111, 66, 193, 0.3);
    }

textarea.input-field {
    min-height: 80px;
    resize: vertical;
}

.submit-btn {
    width: 100%;
    padding: 12px 0;
    background-color: #6f42c1;
    color: #fff;
    font-size: 1.1rem;
    border: none;
    border-radius: 8px;
    margin-top: 15px;
    cursor: pointer;
    transition: all 0.2s ease;
}

.submit-btn:hover {
    background-color: #5925a3;
}

.submit-btn:disabled {
    background-color: #a78bfa;
    cursor: not-allowed;
}

.error-message {
    color: #e63946;
    font-weight: 500;
    margin-bottom: 10px;
    text-align: center;
}
```

courseDetail.css

```
/* ===== ALIGN BUTTONS INLINE ===== */
.course-completed-actions,
.enroll-actions {
    display: flex;
```

```
align-items: center;
justify-content: flex-start;
gap: 10px;
margin-top: 15px;
flex-wrap: wrap;
}

/* Certificate icons */
.certificate-buttons {
  display: flex;
  align-items: center;
  gap: 10px;
}

.certificate-icon-btn {
  background-color: #6f42c1;
  color: white;
  border: none;
  border-radius: 6px;
  padding: 8px 10px;
  cursor: pointer;
  transition: all 0.2s ease-in-out;
  display: flex;
  align-items: center;
  justify-content: center;
}

.certificate-icon-btn:hover {
  background-color: #59339d;
  transform: scale(1.05);
}

/* Enroll & Unenroll Buttons */
.enroll-button,
.unenroll-button {
  padding: 7px 16px;
  border-radius: 6px;
  font-weight: 500;
  border: none;
  cursor: pointer;
  transition: all 0.2s ease-in-out;
  display: flex;
  align-items: center;
  justify-content: center;
}
```

```
}

.enroll-button {
  background-color: #6f42c1;
  color: white;
}

.enroll-button:hover {
  background-color: #683abc;
}

.unenroll-button {
  background-color: #ff4d4d;
  color: white;
}

.unenroll-button:hover {
  background-color: #f24444;
}

/* Completed Status */
.course-completed-tick {
  font-size: 1.4rem;
  margin-left: 8px;
  color: #28a745;
}

.course-completed-text {
  color: #28a745;
  font-weight: 600;
  margin-top: 8px;
}

/* Certificate Links */
.certificate-link {
  color: #6f42c1;
  font-weight: 600;
  margin-top: 7px;
  display: inline-block;
  transition: color 0.2s ease-in-out;
  margin-bottom: -3%;
}

.course-progress-container {
```

```
margin-top: 10px;
margin-bottom: 15px;
width: 80%;
}

.course-progress-bar {
width: 100%;
height: 10px;
background-color: #e9ecf;
border-radius: 10px;
overflow: hidden;
}

.course-progress-fill {
height: 100%;
background: linear-gradient(90deg, #6f42c1, #a871e0);
border-radius: 10px;
transition: width 0.5s ease-in-out;
}

/* ===== CONTAINER ===== */
.course-detail-container {
font-family: 'Poppins', sans-serif;
padding: 20px;
margin-top: 80px;
max-width: 1200px;
margin-left: auto;
margin-right: auto;
background: #f9f7fd;
color: #333;
}

/* ===== COURSE HEADER ===== */
.course-header {
display: flex;
flex-wrap: wrap;
gap: 20px;
background: #fff;
border-radius: 12px;
padding: 25px;
box-shadow: 0 4px 10px rgba(0,0,0,0.05);
}

.course-header-left {
```

```
flex: 2;
min-width: 300px;
}

.course-header-left h1 {
font-size: 2rem;
color: #6D28D9;
margin-bottom: 10px;
}

.course-header-left p {
color: #555;
margin-bottom: 10px;
}

.course-header-left p span {
font-weight: 600;
}

.course-header-right {
flex: 1;
min-width: 250px;
}

.course-header-right img {
width: 100%;
border-radius: 12px;
object-fit: cover;
box-shadow: 0 4px 10px rgba(0,0,0,0.1);
}

.course-header-right .no-image {
width: 100%;
height: 180px;
background: #EDE9FE;
border-radius: 10px;
display: flex;
align-items: center;
justify-content: center;
color: #8b51e7;
font-weight: 600;
}

/* ===== COURSE BODY ===== */
```

```
.course-body {  
    display: flex;  
    flex-wrap: wrap;  
    gap: 20px;  
    margin-top: 20px;  
}  
  
.lesson-player {  
    flex: 2;  
    min-width: 300px;  
    background: #fff;  
    padding: 20px;  
    border-radius: 12px;  
    box-shadow: 0 4px 10px rgba(0,0,0,0.05);  
}  
  
.lesson-player h3 {  
    color: #8442ed;  
    margin-bottom: 15px;  
    font-size: 1.25rem;  
}  
  
.lesson-player video,  
.lesson-player iframe {  
    width: 100%;  
    max-height: 400px;  
    border-radius: 10px;  
    margin-top: 10px;  
    box-shadow: 0 4px 12px rgba(0,0,0,0.1);  
}  
  
.lesson-player p {  
    color: #444;  
    background: #F3E8FF;  
    padding: 12px;  
    border-radius: 8px;  
    font-size: 0.95rem;  
}  
  
/* ===== LESSON LIST ===== */  
.lesson-list {  
    flex: 1;  
    min-width: 250px;  
    background: #fff;
```

```
padding: 20px;  
border-radius: 12px;  
box-shadow: 0 4px 10px rgba(0,0,0,0.05);  
}
```

```
.lesson-list h3 {  
color: #5B21B6;  
margin-bottom: 15px;  
font-size: 1.1rem;  
}
```

```
.lesson-list ul {  
list-style: none;  
padding: 0;  
max-height: 600px;  
overflow-y: auto;  
margin: 0;  
position: relative;  
}
```

```
.lesson-list li {  
padding: 12px 16px 12px 40px;  
margin-bottom: 16px;  
border-left: 4px solid #ddd;  
position: relative;  
cursor: pointer;  
font-weight: 500;  
transition: all 0.2s ease-in-out;  
border-radius: 8px;  
background: #FAF5FF;  
}
```

```
.lesson-list li.active {  
background: #f0fff4;  
border-left-color: #4caf50;  
font-weight: 600;  
}
```

```
.lesson-list li.disabled {  
opacity: 0.5;  
cursor: not-allowed;  
}
```

```
.lesson-list li.free-preview {
```

```
color: #10B981;
font-weight: 500;
}

/* Circle indicator */
.lesson-list li::before {
content: "";
position: absolute;
left: -12px;
top: 12px;
width: 16px;
height: 16px;
border-radius: 50%;
background: #ddd;
transition: all 0.3s ease-in-out;
z-index: 2;
}

.lesson-list li.active::before,
.lesson-list li.completed::before {
background: #4caf50;
box-shadow: 0 0 6px rgba(76, 175, 80, 0.6);
}

/* Lesson title inside li */
.lesson-title {
display: flex;
justify-content: space-between;
align-items: center;
font-weight: 500;
}

/* Free preview badge */
.free-preview {
background: #ffeb3b;
color: #000;
font-size: 10px;
padding: 2px 6px;
border-radius: 4px;
margin-left: 6px;
font-weight: 600;
}

/* ===== TAB TOGGLE ===== */


```

```
.tab-toggle {  
    display: flex;  
    margin-bottom: 12px;  
    gap: 8px;  
}  
  
.tab-toggle button {  
    flex: 1;  
    padding: 8px;  
    border: none;  
    border-radius: 6px;  
    cursor: pointer;  
    background: #EDE9FE;  
    color: #333;  
    font-weight: 600;  
    transition: all 0.2s ease;  
}  
  
.tab-toggle button.active-tab {  
    background: #8140ea;  
    color: #fff;  
}  
  
/* ===== LOADING ===== */  
.course-loading {  
    display: flex;  
    flex-direction: column;  
    align-items: center;  
    justify-content: center;  
    height: 80vh;  
}  
  
.spinner {  
    width: 48px;  
    height: 48px;  
    border: 5px solid #E9D5FF;  
    border-top: 5px solid #8842f8;  
    border-radius: 50%;  
    animation: spin 1s linear infinite;  
    margin-bottom: 50px;  
}  
  
@keyframes spin {  
    0% { transform: rotate(0deg); }  
}
```

```
100% { transform: rotate(360deg); }

/*
===== RESPONSIVE =====
*/
@media (max-width: 1024px) {
    .course-header { flex-direction: column; }
    .course-body { flex-direction: column; }
}

@media (max-width: 900px) {
    .lesson-player, .lesson-list { min-width: 100%; }
}

@media (max-width: 768px) {
    .lesson-list ul { max-height: 400px; }
}

@media (max-width: 480px) {
    .course-header-left h1 { font-size: 1.5rem; }
    .lesson-player video,
    .lesson-player iframe { max-height: 250px; }
}

/*
===== EXAM SECTION =====
*/
.exam-list {
    display: flex;
    flex-direction: column;
    gap: 1rem;
    margin-top: 10px;
}

.exam-card {
    background: #ffffff;
    border-radius: 12px;
    padding: 0.5rem 1.5rem;
    transition: all 0.3s ease;
    cursor: pointer;
    color: #333;
    box-shadow: 0 2px 6px rgba(76, 91, 253, 0.08);
}

.exam-card-header {
    display: flex;
    justify-content: space-between;
```

```
    align-items: center;
    margin-bottom: 0.7rem;
}

.exam-card-header h4 {
    margin: 0;
    font-size: 1.0rem;
    font-weight: 600;
    color: #4c1d95;
}

.exam-number {
    background: #8b5cf6;
    color: #fff;
    padding: 0.2rem 0.6rem;
    border-radius: 6px;
    font-size: 0.8rem;
    font-weight: 500;
}

.exam-card-details {
    display: flex;
    justify-content: space-between;
    font-size: 0.9rem;
    color: #555;
}

.no-exams {
    color: #888;
    text-align: center;
    font-style: italic;
    margin-top: 1rem;
}

@media (max-width: 600px) {
    .exam-card {
        padding: 1rem;
    }

    .exam-card-details {
        flex-direction: column;
        gap: 0.4rem;
    }
}
```

```
.completed-exam {  
    border: 2px solid #28a745;  
    background-color: #e8f5e9;  
    transition: all 0.3s ease;  
}  
  
.completed-exam:hover {  
    background-color: #d4edda;  
}  
  
.exam-tick {  
    color: #28a745;  
    margin-left: 6px;  
}  
  
.exam-score {  
    color: #1b5e20;  
    font-size: 0.9rem;  
    margin-top: 4px;  
}  
/* ===== ACTION BUTTONS INLINE ===== */  
.action-buttons-row {  
    display: flex;  
    align-items: center;  
    justify-content: flex-start;  
    gap: 12px;  
    flex-wrap: wrap;  
    margin-top: 15px;  
}  
  
.action-buttons-row button {  
    display: flex;  
    align-items: center;  
    justify-content: center;  
    gap: 6px;  
}
```

aboutus.css

```
.aboutus-container {  
    font-family: 'Arial', sans-serif;  
    background-color: #f4f4f9;
```

```
padding: 80px 20px;
min-height: 80vh;
}
.aboutus-content {
max-width: 900px;
margin: 0 auto;
text-align: center;
}
.aboutus-content h1 {
font-size: 2.5rem;
color: #343a40;
margin-bottom: 20px;
font-weight: 700;
}
.aboutus-content h2 {
font-size: 2rem;
color: #343a40;
margin-top: 40px;
margin-bottom: 20px;
font-weight: 600;
}
.aboutus-content p {
font-size: 1.1rem;
color: #555;
line-height: 1.8;
margin-bottom: 20px;
}
.aboutus-content p span {
color: #ffb400;
font-weight: 600;
}
.aboutus-team {
display: flex;
flex-wrap: wrap;
justify-content: center;
gap: 20px;
margin-top: 40px;
}
.aboutus-team-member {
background: #fff;
border-radius: 12px;
padding: 20px;
width: 250px;
```

```
        box-shadow: 0 4px 12px rgba(0,0,0,0.1);
        transition: transform 0.3s ease, box-shadow 0.3s ease;
    }

.aboutus-team-member:hover {
    transform: translateY(-5px);
    box-shadow: 0 6px 18px rgba(0,0,0,0.15);
}

.aboutus-team-member img {
    width: 100px;
    height: 100px;
    object-fit: cover;
    border-radius: 50%;
    margin-bottom: 15px;
}

.aboutus-team-member h4 {
    font-size: 1.1rem;
    font-weight: 600;
    color: #343a40;
    margin-bottom: 5px;
}

.aboutus-team-member p {
    font-size: 0.9rem;
    color: #666;
    line-height: 1.5;
}

@media (max-width: 768px) {
    .aboutus-content h1 {
        font-size: 2rem;
    }
}

.aboutus-content h2 {
    font-size: 1.5rem;
}

.aboutus-team-member {
    width: 100%;
}
```

8. Reports

The LMS includes a reporting module that provides clear, organized, and actionable insights for Admins and Instructors. These reports help monitor platform performance, user activities, and academic progress.

8.1 Admin Reports

- **User Reports:** List of all Students, Instructors, and Admins with activity status.
- **Course Reports:** Details on active, pending, rejected, and published courses.
- **Enrollment Reports:** Total enrollments, expired enrollments, and revenue generated.
- **Exam Reports:** Exam schedules, results summary, and overall performance trends.
- **Revenue & Commission Reports:** Earnings, instructor payouts, and platform commission details.

8.2 Instructor Reports

- **Course Performance:** Enrollments, completion rates, and student engagement.
- **Student Performance:** Individual student scores, progress, and submission reports.
- **Exam Reports:** Marks distribution, average scores, and question-wise performance.
- **Earnings Reports:** Revenue from each course, total earnings, and withdrawal history.

8.3 Student Reports

- **Progress Report:** Completed lessons, pending tasks, and overall course completion percentage.
- **Result Reports:** Exam scores, attempts history, and performance summary.

9. Testing

Testing ensures that the LMS functions correctly, meets requirements, and provides a smooth user experience. Various testing methods are applied to verify system performance, reliability, and security.

9.1 Unit Testing

- Tests individual components such as login, course creation, API endpoints, and database operations.
- Ensures each module works independently without errors.

9.2 Integration Testing

- Verifies that combined modules (e.g., course + lesson + exam) work together smoothly.
- Ensures proper data flow between frontend, backend, and database.

9.3 Functional Testing

- Checks whether all system features meet the functional requirements.
- Includes testing of user roles: Admin, Instructor, and Student.

9.4 User Interface (UI) Testing

- Ensures the platform is responsive, user-friendly, and displays elements correctly on all devices.

9.5 Performance Testing

- Tests loading speed, system stability under multiple users, and response time.

9.6 Security Testing

- Validates secure login, role-based access, data protection, and safe payment handling.

9.7 User Acceptance Testing (UAT)

- Final testing by end users to confirm that the system works as expected before deployment.

10. Conclusion and Future Enhancement

The Learning Management System (LMS) provides a complete digital platform for delivering, managing, and monitoring educational content. It offers a structured environment where students can learn efficiently, instructors can manage courses with ease, and administrators can oversee the entire system.

With features such as course creation, lesson uploads, online exams, progress tracking, and reporting, the LMS ensures a smooth, interactive, and organized learning experience. The system successfully meets its objectives of enhancing accessibility, simplifying management, and supporting effective digital learning.

10.2 Future Enhancements

To continuously improve the LMS and offer a more advanced learning experience, the following enhancements can be considered in future versions:

- **AI-based Recommendation System**
Suggest courses based on student interests and performance.
- **Dark Mode & Light Mode Support**
Provide theme switching (Dark Mode, Light Mode, and Auto Mode) to enhance usability and reduce eye strain.
- **Multi-language Support**
Provide interface translation and multilingual course capabilities.
- **Offline Download Mode**
Allow students to download certain content and view it offline (in app).
- **Analytics Dashboard Enhancements**
Advanced insights with charts, predictions, and behavior analysis.
- Add **student course ratings & reviews** to improve course quality and help others choose better courses.
- Add **course duration tracking** (total hours, lesson time, estimated completion time) for better transparency.

11. References/ Bibliography

<https://getbootstrap.com/>

<https://unsplash.com/>

www.coursera.org

<https://www.udemy.com/>