```
'use client';

import { useState, useEffect } from 'react';

import { useRouter } from 'next/navigation';

import { User, BookOpen, Users, Settings, LogOut, Bell, BarChart3, FileText, Award, ClipboardList,
Menu, X, Clock, GraduationCap, Plus, Edit, Trash2 } from 'lucide-react';


export default function TrainerDashboard() {

  const router = useRouter();

  const [activeMenu, setActiveMenu] = useState('dashboard');

  const [showInstructorSubmenu, setShowInstructorSubmenu] = useState(false);

  const [showUserSubmenu, setShowUserSubmenu] = useState(false);

  const [sidebarOpen, setSidebarOpen] = useState(true);

  const [trainerData, setTrainerData] = useState(null);

  const [loading, setLoading] = useState(true);

  const [showCreateCourse, setShowCreateCourse] = useState(false);

  const [showCreateQuiz, setShowCreateQuiz] = useState(false);

  const [newCourse, setNewCourse] = useState({

    title: '',

    duration: '',

    startDate: '',

    endDate: '',

    startTime: '',

    endTime: '',

    modules: [''],

    beneficiaries: [''],

    color: 'from-green-500 to-green-600'

  });

  const [newQuiz, setNewQuiz] = useState({

    title: '',

    course: '',

    duration: '',
```

```
  totalMarks: '',
  passingMarks: '',
  questions: [{ question: '', options: ['', '', '', ''], correctAnswer: '' }]
});
const [quizzes, setQuizzes] = useState([
  {
    id: 1,
    title: "Post-Harvest Management Basics",
    course: "Post-harvest management of horticulture produce",
    duration: "30 minutes",
    totalMarks: 100,
    passingMarks: 50,
    totalQuestions: 10,
    attempts: [
      { traineeId: 1, traineeName: "John Doe", attemptCount: 3, lastScore: 85, status: "Passed" },
      { traineeId: 2, traineeName: "Jane Smith", attemptCount: 2, lastScore: 72, status: "Passed" },
      { traineeId: 3, traineeName: "Mike Johnson", attemptCount: 1, lastScore: 45, status: "Failed" }
    ]
  },
  {
    id: 2,
    title: "Refrigeration Systems Quiz",
    course: "Cold chain Operations",
    duration: "45 minutes",
    totalMarks: 100,
    passingMarks: 60,
    totalQuestions: 15,
    attempts: [
      { traineeId: 4, traineeName: "Sarah Williams", attemptCount: 2, lastScore: 88, status: "Passed" },
      { traineeId: 5, traineeName: "David Brown", attemptCount: 4, lastScore: 55, status: "Failed" }
    ]
```

```
  }
]);
const [selectedTemplate, setSelectedTemplate] = useState(null);
const [showIssueCertificate, setShowIssueCertificate] = useState(false);
const [certificateForm, setCertificateForm] = useState({
  traineeName: '',
  course: '',
  completionDate: '',
  grade: '',
  certificateId: ''
});
const [issuedCertificates, setIssuedCertificates] = useState([
  {
    id: 'CERT001',
    traineeName: 'John Doe',
    course: 'Post-harvest management of horticulture produce',
    template: 1,
    issuedDate: '2024-10-15',
    grade: 'A',
    status: 'Issued'
  },
  {
    id: 'CERT002',
    traineeName: 'Jane Smith',
    course: 'Cold chain Operations',
    template: 3,
    issuedDate: '2024-10-14',
    grade: 'B+',
    status: 'Issued'
  }
]);
```

```
const certificateTemplates = [
  {
    id: 1,
    name: 'Classic Elegance',
    description: 'Traditional certificate with ornate borders',
    color: 'from-amber-50 to-amber-100',
    borderColor: 'border-amber-600',
    accentColor: 'text-amber-700'
  },
  {
    id: 2,
    name: 'Modern Professional',
    description: 'Clean and contemporary design',
    color: 'from-blue-50 to-blue-100',
    borderColor: 'border-blue-600',
    accentColor: 'text-blue-700'
  },
  {
    id: 3,
    name: 'Agricultural Green',
    description: 'Nature-inspired green theme',
    color: 'from-green-50 to-green-100',
    borderColor: 'border-green-600',
    accentColor: 'text-green-700'
  },
  {
    id: 4,
    name: 'Premium Gold',
    description: 'Luxurious gold accented design',
    color: 'from-yellow-50 to-yellow-100',
```

```
  borderColor: 'border-yellow-600',
  accentColor: 'text-yellow-700'
},
{
  id: 5,
  name: 'Royal Purple',
  description: 'Distinguished purple theme',
  color: 'from-purple-50 to-purple-100',
  borderColor: 'border-purple-600',
  accentColor: 'text-purple-700'
},
{
  id: 6,
  name: 'Corporate Blue',
  description: 'Professional business style',
  color: 'from-indigo-50 to-indigo-100',
  borderColor: 'border-indigo-600',
  accentColor: 'text-indigo-700'
},
{
  id: 7,
  name: 'Fresh Mint',
  description: 'Light and refreshing design',
  color: 'from-teal-50 to-teal-100',
  borderColor: 'border-teal-600',
  accentColor: 'text-teal-700'
},
{
  id: 8,
  name: 'Sunset Orange',
  description: 'Vibrant and energetic theme',
```

```
    color: 'from-orange-50 to-orange-100',
    borderColor: 'border-orange-600',
    accentColor: 'text-orange-700'
  },
  {
    id: 9,
    name: 'Berry Red',
    description: 'Bold and impactful design',
    color: 'from-red-50 to-red-100',
    borderColor: 'border-red-600',
    accentColor: 'text-red-700'
  },
  {
    id: 10,
    name: 'Rose Pink',
    description: 'Elegant and sophisticated style',
    color: 'from-pink-50 to-pink-100',
    borderColor: 'border-pink-600',
    accentColor: 'text-pink-700'
  }
];

// Course data from the document
const [coursesData, setCoursesData] = useState([
  {
    id: 1,
    title: "Post-harvest management of horticulture produce",
    duration: "5 days (40 hours)",
    modules: [
      "Introduction on Post-Harvest Management",
      "Importance of Post-Harvest Science",
```

```
    "Pack house operations",
    "Post-Harvest Management Protocols",
    "Packing and bar coding standards for traceability"
  ],
  beneficiaries: [
    "Graduates in science and agriculture",
    "Agri entrepreneurs",
    "Exporters and importers",
    "Cold chain services providers",
    "Farmers group"
  ],
  color: "from-green-500 to-green-600"
},
{
  id: 2,
  title: "Cold chain Operations",
  duration: "5 days (40 hours)",
  modules: [
    "Introduction to Refrigeration",
    "Refrigeration System and Components",
    "Role of Refrigeration and Applications",
    "Refrigeration In Agri Logistics",
    "Advancement in Refrigeration"
  ],
  beneficiaries: [
    "Agri entrepreneurs",
    "Diploma/graduates in engineering",
    "Exporters and importers",
    "Cold chain services providers",
    "Farmers groups"
  ],
```

```
  color: "from-blue-500 to-blue-600"
 },
 {
  id: 3,
  title: "Cold chain support schemes and incentives provided to encourage investments",
  duration: "3 days",
  modules: [
   "Basics of cold chain",
   "Refrigeration equipment operations",
   "Maintenance of refrigeration plant and insulated doors",
   "Troubleshooting and servicing of refrigeration equipments"
  ],
  beneficiaries: [
   "Agri entrepreneurs",
   "Exporters and importers",
   "Cold chain services providers",
   "Farmers Producers groups",
   "SHM officers"
  ],
  color: "from-purple-500 to-purple-600"
 },
 {
  id: 4,
  title: "Agri logistics and retail distributions",
  duration: "3 days",
  modules: [
   "Supply chain management in agriculture",
   "Distribution channels and logistics",
   "Retail operations and management",
   "Technology in agri-logistics"
  ],
```

```
    beneficiaries: [
      "Agri entrepreneurs",
      "Exporters and importers",
      "Cold chain services providers",
      "Food processors",
      "Farmers Producers groups",
      "SHM officers"
    ],
    color: "from-orange-500 to-orange-600"
  }
]);

// Fetch trainer data from your database
useEffect(() => {
  const fetchTrainerData = async () => {
    try {
      const userData = localStorage.getItem('user');
      if (!userData) {
        router.push('/login');
        return;
      }
      const user = JSON.parse(userData);
      const res = await fetch('/api/trainer/profile', {
        method: 'GET',
        headers: { 'x-user-email': user.email }
      });
      const data = await res.json();
      if (res.ok) setTrainerData(data);
      else router.push('/login');
    } catch (error) {
      console.error('Error fetching trainer data:', error);
```

```
      router.push('/login');
    } finally {
      setLoading(false);
    }
  };
  fetchTrainerData();
}, [router]);

// Fetch courses from database
useEffect(() => {
  const fetchCourses = async () => {
    try {
      const response = await fetch('/api/courses/list');
      if (response.ok) {
        const courses = await response.json();
        setCoursesData(courses);
      }
    } catch (error) {
      console.error('Error fetching courses:', error);
    }
  };

  if (trainerData) {
    fetchCourses();
  }
}, [trainerData]);

// Fetch quizzes from database
useEffect(() => {
  const fetchQuizzes = async () => {
    try {
```

```
      const response = await fetch(`/api/quiz/list?createdBy=${trainerData.email}&userRole=trainer`);
      if (response.ok) {
        const quizzes = await response.json();
        setQuizzes(quizzes);
      }
    } catch (error) {
      console.error('Error fetching quizzes:', error);
    }
  };

  if (trainerData) {
    fetchQuizzes();
  }
 }, [trainerData]);

 // Fetch certificates from database
 useEffect(() => {
  const fetchCertificates = async () => {
   try {
    const response = await
fetch(`/api/certificates/list?issuedBy=${trainerData.email}&userRole=trainer`);
      if (response.ok) {
        const certificates = await response.json();
        setIssuedCertificates(certificates);
      }
    } catch (error) {
      console.error('Error fetching certificates:', error);
    }
  };

  if (trainerData) {
```

```
    fetchCertificates();
  }
}, [trainerData]);

const handleLogout = () => {
  localStorage.removeItem('user');
  router.push('/login');
};

// Navigate to noticeboard page
const handleNoticeboardClick = () => {
  router.push('/trainer/noticeboard');
};

// Handle course creation
const handleAddModule = () => {
  setNewCourse({ ...newCourse, modules: [...newCourse.modules, ''] });
};

const handleRemoveModule = (index) => {
  const updated = newCourse.modules.filter((_, i) => i !== index);
  setNewCourse({ ...newCourse, modules: updated });
};

const handleModuleChange = (index, value) => {
  const updated = [...newCourse.modules];
  updated[index] = value;
  setNewCourse({ ...newCourse, modules: updated });
};

const handleAddBeneficiary = () => {
```

```
  setNewCourse({ ...newCourse, beneficiaries: [...newCourse.beneficiaries, ''] });
 };

 const handleRemoveBeneficiary = (index) => {
  const updated = newCourse.beneficiaries.filter((_, i) => i !== index);
  setNewCourse({ ...newCourse, beneficiaries: updated });
 };

 const handleBeneficiaryChange = (index, value) => {
  const updated = [...newCourse.beneficiaries];
  updated[index] = value;
  setNewCourse({ ...newCourse, beneficiaries: updated });
 };

 const handleCreateCourse = async (e) => {
  e.preventDefault();
  const filteredModules = newCourse.modules.filter(m => m.trim() !== '');
  const filteredBeneficiaries = newCourse.beneficiaries.filter(b => b.trim() !== '');

  // Calculate duration from dates and times
  let durationText = newCourse.duration;
  if (newCourse.startDate && newCourse.endDate) {
   const start = new Date(newCourse.startDate);
   const end = new Date(newCourse.endDate);
   const diffTime = Math.abs(end - start);
   const diffDays = Math.ceil(diffTime / (1000 * 60 * 60 * 24)) + 1; // +1 to include both start and
end days

   const timeInfo = newCourse.startTime && newCourse.endTime
    ? ` (${newCourse.startTime} - ${newCourse.endTime})`
    : '';
```

```
    durationText = `${diffDays} day${diffDays > 1 ? 's' : ''}${timeInfo}`;
  }

  if (newCourse.title && (newCourse.duration || (newCourse.startDate && newCourse.endDate))
&& filteredModules.length > 0 && filteredBeneficiaries.length > 0) {
    const courseData = {
      title: newCourse.title,
      duration: durationText,
      startDate: newCourse.startDate,
      endDate: newCourse.endDate,
      startTime: newCourse.startTime,
      endTime: newCourse.endTime,
      modules: filteredModules,
      beneficiaries: filteredBeneficiaries,
      color: newCourse.color,
      createdBy: trainerData.email,
      status: 'Active'
    };

    try {
      // Save to database
      const response = await fetch('/api/courses/create', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify(courseData)
      });

      if (response.ok) {
```

```
      const savedCourse = await response.json();
      // Add to local state with database ID
      setCoursesData([...coursesData, savedCourse]);
      setShowCreateCourse(false);
      setNewCourse({
        title: '',
        duration: '',
        startDate: '',
        endDate: '',
        startTime: '',
        endTime: '',
        modules: [''],
        beneficiaries: [''],
        color: 'from-green-500 to-green-600'
      });
      alert('Course created successfully!');
    } else {
      alert('Failed to create course. Please try again.');
    }
  } catch (error) {
    console.error('Error creating course:', error);
    alert('Error creating course. Please try again.');
  }
 }
};

const handleCancelCreate = () => {
 setShowCreateCourse(false);
 setNewCourse({
  title: '',
  duration: '',
```

```
    startDate: '',
    endDate: '',
    startTime: '',
    endTime: '',
    modules: [''],
    beneficiaries: [''],
    color: 'from-green-500 to-green-600'
  });
};

// Quiz handlers
const handleAddQuestion = () => {
  setNewQuiz({
    ...newQuiz,
    questions: [...newQuiz.questions, { question: '', options: ['', '', '', ''], correctAnswer: '' }]
  });
};

const handleRemoveQuestion = (index) => {
  const updated = newQuiz.questions.filter((_, i) => i !== index);
  setNewQuiz({ ...newQuiz, questions: updated });
};

const handleQuestionChange = (index, field, value) => {
  const updated = [...newQuiz.questions];
  updated[index][field] = value;
  setNewQuiz({ ...newQuiz, questions: updated });
};

const handleOptionChange = (qIndex, optIndex, value) => {
  const updated = [...newQuiz.questions];
```

```
    updated[qIndex].options[optIndex] = value;
    setNewQuiz({ ...newQuiz, questions: updated });
  };

  const handleCreateQuiz = async (e) => {
    e.preventDefault();
    const filteredQuestions = newQuiz.questions.filter(q =>
      q.question.trim() !== '' &&
      q.options.every(opt => opt.trim() !== '') &&
      q.correctAnswer.trim() !== ''
    );

    if (newQuiz.title && newQuiz.course && newQuiz.duration && newQuiz.totalMarks &&
newQuiz.passingMarks && filteredQuestions.length > 0) {
      try {
        const response = await fetch('/api/quiz/create', {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json',
          },
          body: JSON.stringify({
            title: newQuiz.title,
            course: newQuiz.course,
            duration: newQuiz.duration,
            totalMarks: newQuiz.totalMarks,
            passingMarks: newQuiz.passingMarks,
            questions: filteredQuestions,
            createdBy: trainerData.email
          })
        });
```

```
      if (response.ok) {
        const savedQuiz = await response.json();
        setQuizzes([...quizzes, { ...savedQuiz, attempts: [] }]);
        setShowCreateQuiz(false);
        setNewQuiz({
          title: '',
          course: '',
          duration: '',
          totalMarks: '',
          passingMarks: '',
          questions: [{ question: '', options: ['', '', '', ''], correctAnswer: '' }]
        });
        alert('Quiz created successfully!');
      } else {
        alert('Failed to create quiz. Please try again.');
      }
    } catch (error) {
      console.error('Error creating quiz:', error);
      alert('Error creating quiz. Please try again.');
    }
  }
};

const handleCancelQuiz = () => {
  setShowCreateQuiz(false);
  setNewQuiz({
    title: '',
    course: '',
    duration: '',
    totalMarks: '',
    passingMarks: '',
```

```
    questions: [{ question: '', options: ['', '', '', ''], correctAnswer: '' }]
  });
};

// Certificate handlers
const handleSelectTemplate = (template) => {
  setSelectedTemplate(template);
  setShowIssueCertificate(true);
};

const handleIssueCertificate = async (e) => {
  e.preventDefault();
  try {
    const response = await fetch('/api/certificates/issue', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        traineeName: certificateForm.traineeName,
        traineeEmail: certificateForm.traineeEmail,
        course: certificateForm.course,
        templateId: selectedTemplate.id,
        grade: certificateForm.grade,
        completionDate: certificateForm.completionDate,
        issuedBy: trainerData.email
      })
    });

    if (response.ok) {
      const savedCertificate = await response.json();
```

```
      setIssuedCertificates([...issuedCertificates, {
        id: savedCertificate.certificateId,
        traineeName: savedCertificate.traineeName,
        course: savedCertificate.course,
        template: savedCertificate.templateId,
        issuedDate: savedCertificate.completionDate.split('T')[0],
        grade: savedCertificate.grade,
        status: savedCertificate.status
      }]);
      setShowIssueCertificate(false);
      setSelectedTemplate(null);
      setCertificateForm({
        traineeName: '',
        course: '',
        completionDate: '',
        grade: '',
        certificateId: ''
      });
      alert('Certificate issued successfully!');
    } else {
      alert('Failed to issue certificate. Please try again.');
    }
  } catch (error) {
    console.error('Error issuing certificate:', error);
    alert('Error issuing certificate. Please try again.');
  }
};

const handleCancelCertificate = () => {
  setShowIssueCertificate(false);
  setSelectedTemplate(null);
```

```
  setCertificateForm({
    traineeName: '',
    course: '',
    completionDate: '',
    grade: '',
    certificateId: ''
  });
 };

 const renderContent = () => {
  switch (activeMenu) {
    case 'dashboard':
      return (
        <div>
          <h2 className="text-2xl font-bold text-gray-800 mb-6">Dashboard Overview</h2>
          <div className="grid grid-cols-1 md:grid-cols-2 gap-4 mb-6">
            <div className="bg-gradient-to-br from-green-500 to-green-600 rounded-lg p-6 text-
white">
              <BookOpen className="w-8 h-8 mb-3 opacity-80" />
              <p className="text-3xl font-bold mb-1">{trainerData.totalCourses || 0}</p>
              <p className="text-green-100">Total Courses</p>
            </div>
            <div className="bg-gradient-to-br from-blue-500 to-blue-600 rounded-lg p-6 text-white">
              <Users className="w-8 h-8 mb-3 opacity-80" />
              <p className="text-3xl font-bold mb-1">{trainerData.totalStudents || 0}</p>
              <p className="text-blue-100">Total Students</p>
            </div>
            <div className="bg-gradient-to-br from-purple-500 to-purple-600 rounded-lg p-6 text-
white">
              <FileText className="w-8 h-8 mb-3 opacity-80" />
              <p className="text-3xl font-bold mb-1">{trainerData.activeBatches || 0}</p>
              <p className="text-purple-100">Active Batches</p>
```

```
          </div>
          <div className="bg-gradient-to-br from-orange-500 to-orange-600 rounded-lg p-6 text-white">
            <Award className="w-8 h-8 mb-3 opacity-80" />
            <p className="text-3xl font-bold mb-1">{trainerData.completionRate || '0%'}</p>
            <p className="text-orange-100">Completion Rate</p>
          </div>
        </div>
      </div>
    );

  case 'training-course':
    return (
      <div>
        <div className="mb-6 flex items-center justify-between">
          <div>
            <h2 className="text-2xl font-bold text-gray-800 mb-2">Training Courses</h2>
            <p className="text-gray-600">Comprehensive courses for agricultural professionals</p>
          </div>
          <button
            onClick={() => setShowCreateCourse(true)}
            className="flex items-center gap-2 px-6 py-3 bg-green-600 text-white rounded-lg hover:bg-green-700 transition-colors font-medium shadow-md"
          >
            <Plus className="w-5 h-5" />
            Create Course
          </button>
        </div>

        {/* Create Course Form */}
        {showCreateCourse && (
          <div className="bg-white rounded-lg shadow-md p-6 mb-6 border-2 border-green-200">
```

```
<h3 className="text-xl font-bold text-gray-800 mb-4">Create New Course</h3>

<form onSubmit={handleCreateCourse} className="space-y-4">

 <div className="grid grid-cols-1 md:grid-cols-2 gap-4">

  <div className="md:col-span-2">

   <label className="block text-gray-700 font-medium mb-2">Course Title *</label>

   <input

    type="text"

    required

    value={newCourse.title}

    onChange={(e) => setNewCourse({ ...newCourse, title: e.target.value })}

    className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none
focus:ring-2 focus:ring-green-500"

    placeholder="Enter course title"

   />

  </div>

 </div>


 {/* Date and Time Section */}

 <div className="border-t pt-4">

  <h4 className="text-lg font-semibold text-gray-800 mb-3">Course Schedule</h4>


  <div className="grid grid-cols-1 md:grid-cols-2 gap-4 mb-4">

   <div>

    <label className="block text-gray-700 font-medium mb-2">Start Date *</label>

    <input

     type="date"

     required

     value={newCourse.startDate}

     onChange={(e) => setNewCourse({ ...newCourse, startDate: e.target.value })}

     className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none
focus:ring-2 focus:ring-green-500"

    />
```

```
                </div>
                <div>
                  <label className="block text-gray-700 font-medium mb-2">End Date *</label>
                  <input
                    type="date"
                    required
                    value={newCourse.endDate}
                    min={newCourse.startDate}
                    onChange={(e) => setNewCourse({ ...newCourse, endDate: e.target.value })}
                    className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none
focus:ring-2 focus:ring-green-500"
                  />
                </div>
              </div>

              <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                <div>
                  <label className="block text-gray-700 font-medium mb-2">Start Time
(Optional)</label>
                  <input
                    type="time"
                    value={newCourse.startTime}
                    onChange={(e) => setNewCourse({ ...newCourse, startTime: e.target.value })}
                    className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none
focus:ring-2 focus:ring-green-500"
                  />
                </div>
                <div>
                  <label className="block text-gray-700 font-medium mb-2">End Time
(Optional)</label>
                  <input
                    type="time"
```

```
                value={newCourse.endTime}
                onChange={(e) => setNewCourse({ ...newCourse, endTime: e.target.value })}
                className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500"
              />
            </div>
          </div>

          <div className="mt-3">
            <label className="block text-gray-700 font-medium mb-2">Or Enter Duration Manually (Optional)</label>
            <input
              type="text"
              value={newCourse.duration}
              onChange={(e) => setNewCourse({ ...newCourse, duration: e.target.value })}
              className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500"
              placeholder="e.g., 5 days (40 hours)"
            />
            <p className="text-xs text-gray-500 mt-1">Duration will be auto-calculated from dates if not provided</p>
          </div>
        </div>

        <div>
          <label className="block text-gray-700 font-medium mb-2">Color Theme</label>
          <select
            value={newCourse.color}
            onChange={(e) => setNewCourse({ ...newCourse, color: e.target.value })}
            className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500"
          >
            <option value="from-green-500 to-green-600">Green</option>
```

```
                <option value="from-blue-500 to-blue-600">Blue</option>
                <option value="from-purple-500 to-purple-600">Purple</option>
                <option value="from-orange-500 to-orange-600">Orange</option>
                <option value="from-pink-500 to-pink-600">Pink</option>
                <option value="from-red-500 to-red-600">Red</option>
              </select>
            </div>

            <div>
              <label className="block text-gray-700 font-medium mb-2">Modules *</label>
              {newCourse.modules.map((module, index) => (
                <div key={index} className="flex gap-2 mb-2">
                  <input
                    type="text"
                    value={module}
                    onChange={(e) => handleModuleChange(index, e.target.value)}
                    className="flex-1 px-4 py-2 border border-gray-300 rounded-lg focus:outline-none
focus:ring-2 focus:ring-green-500"
                    placeholder={`Module ${index + 1}`}
                  />
                  {newCourse.modules.length > 1 && (
                    <button
                      type="button"
                      onClick={() => handleRemoveModule(index)}
                      className="px-3 py-2 bg-red-100 text-red-600 rounded-lg hover:bg-red-200"
                    >
                      <Trash2 className="w-4 h-4" />
                    </button>
                  )}
                </div>
              ))}
```

```
                <button
                  type="button"
                  onClick={handleAddModule}
                  className="mt-2 flex items-center gap-2 px-4 py-2 bg-green-50 text-green-600 rounded-lg hover:bg-green-100"
                >
                  <Plus className="w-4 h-4" />
                  Add Module
                </button>
              </div>

              <div>
                <label className="block text-gray-700 font-medium mb-2">Beneficiaries *</label>
                {newCourse.beneficiaries.map((beneficiary, index) => (
                  <div key={index} className="flex gap-2 mb-2">
                    <input
                      type="text"
                      value={beneficiary}
                      onChange={(e) => handleBeneficiaryChange(index, e.target.value)}
                      className="flex-1 px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500"
                      placeholder={`Beneficiary ${index + 1}`}
                    />
                    {newCourse.beneficiaries.length > 1 && (
                      <button
                        type="button"
                        onClick={() => handleRemoveBeneficiary(index)}
                        className="px-3 py-2 bg-red-100 text-red-600 rounded-lg hover:bg-red-200"
                      >
                        <Trash2 className="w-4 h-4" />
                      </button>
                    )}
```

```
                </div>
              ))}
              <button
                type="button"
                onClick={handleAddBeneficiary}
                className="mt-2 flex items-center gap-2 px-4 py-2 bg-green-50 text-green-600 rounded-lg hover:bg-green-100"
              >
                <Plus className="w-4 h-4" />
                Add Beneficiary
              </button>
            </div>

            <div className="flex justify-end gap-3 pt-4 border-t">
              <button
                type="button"
                onClick={handleCancelCreate}
                className="px-6 py-2 bg-gray-200 text-gray-700 rounded-lg hover:bg-gray-300 transition-colors"
              >
                Cancel
              </button>
              <button
                type="submit"
                className="px-6 py-2 bg-green-600 text-white rounded-lg hover:bg-green-700 transition-colors"
              >
                Create Course
              </button>
            </div>
          </form>
        </div>
```

```
        )}


        {/* Course List */}
        <div className="space-y-6">
          {coursesData.map((course, index) => (
            <div key={course.id} className="bg-white rounded-lg shadow-md overflow-hidden
hover:shadow-lg transition-shadow">
              {/* Course Header */}
              <div className={`bg-gradient-to-r ${course.color} p-6 text-white`}>
                <div className="flex items-start justify-between">
                  <div className="flex-1">
                    <div className="flex items-center gap-2 mb-2">
                      <span className="bg-white/20 px-3 py-1 rounded-full text-sm font-semibold">
                        Course {index + 1}
                      </span>
                    </div>
                    <h3 className="text-xl font-bold mb-2">{course.title}</h3>
                    <div className="flex items-center gap-2 text-white/90">
                      <Clock className="w-4 h-4" />
                      <span className="text-sm">{course.duration}</span>
                    </div>
                  </div>
                  <BookOpen className="w-12 h-12 opacity-20" />
                </div>
              </div>

              {/* Course Content */}
              <div className="p-6">
                <div className="grid md:grid-cols-2 gap-6">
                  {/* Modules Section */}
                  <div>
```

```
  <h4 className="text-lg font-semibold text-gray-800 mb-3 flex items-center gap-2">
    <FileText className="w-5 h-5 text-green-600" />
    Modules
  </h4>
  <ul className="space-y-2">
    {course.modules.map((module, idx) => (
      <li key={idx} className="flex items-start gap-2 text-gray-700">
        <span className="text-green-600 font-bold mt-1">{idx + 1}.</span>
        <span className="text-sm">{module}</span>
      </li>
    ))}
  </ul>
</div>

{/* Beneficiaries Section */}
<div>
  <h4 className="text-lg font-semibold text-gray-800 mb-3 flex items-center gap-2">
    <GraduationCap className="w-5 h-5 text-blue-600" />
    Who Can Benefit
  </h4>
  <div className="space-y-2">
    {course.beneficiaries.map((beneficiary, idx) => (
      <div key={idx} className="flex items-center gap-2">
        <div className="w-2 h-2 bg-blue-500 rounded-full"></div>
        <span className="text-sm text-gray-700">{beneficiary}</span>
      </div>
    ))}
  </div>
 </div>
</div>
```

```
                {/* Action Buttons */}
                <div className="mt-6 pt-6 border-t flex gap-3">
                  <button className="flex-1 px-4 py-2 bg-green-600 text-white rounded-lg hover:bg-green-700 transition-colors font-medium">
                    View Details
                  </button>
                  <button className="flex-1 px-4 py-2 border-2 border-green-600 text-green-600 rounded-lg hover:bg-green-50 transition-colors font-medium">
                    Enroll Students
                  </button>
                </div>
              </div>
            </div>
          ))}
        </div>
      </div>
    );

  case 'exam-score':
    return (
      <div className="bg-white rounded-lg shadow-md p-8">
        <h2 className="text-3xl font-bold text-gray-800 mb-8">Exam Scores</h2>

        <form className="space-y-6">
          {/* Batch and Trainee Row */}
          <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
            <div>
              <label className="block text-gray-700 font-medium mb-2">Batch</label>
              <select className="w-full px-4 py-3 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500 bg-white text-gray-500">
                <option>Select Batch</option>
                <option>Batch 1</option>
```

```
              <option>Batch 2</option>
              <option>Batch 3</option>
            </select>
          </div>
          <div>
            <label className="block text-gray-700 font-medium mb-2">Trainee</label>
            <select className="w-full px-4 py-3 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500 bg-white text-gray-500">
              <option>Select Trainee</option>
              <option>Trainee 1</option>
              <option>Trainee 2</option>
              <option>Trainee 3</option>
            </select>
          </div>
        </div>

        {/* Exam Title and Exam Type Row */}
        <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
          <div>
            <label className="block text-gray-700 font-medium mb-2">Exam Title</label>
            <input
              type="text"
              className="w-full px-4 py-3 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500"
              placeholder="Enter exam title"
            />
          </div>
          <div>
            <label className="block text-gray-700 font-medium mb-2">Exam Type</label>
            <select className="w-full px-4 py-3 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500 bg-white text-gray-700">
              <option>Midterm</option>
```

```
            <option>Final</option>
            <option>Quiz</option>
            <option>Assignment</option>
          </select>
        </div>
      </div>

      {/* Total Marks and Obtained Marks Row */}
      <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
        <div>
          <label className="block text-gray-700 font-medium mb-2">Total Marks</label>
          <input
            type="number"
            className="w-full px-4 py-3 border border-gray-300 rounded-lg focus:outline-none
focus:ring-2 focus:ring-green-500"
            placeholder="Enter total marks"
          />
        </div>
        <div>
          <label className="block text-gray-700 font-medium mb-2">Obtained Marks</label>
          <input
            type="number"
            className="w-full px-4 py-3 border border-gray-300 rounded-lg focus:outline-none
focus:ring-2 focus:ring-green-500"
            placeholder="Enter obtained marks"
          />
        </div>
      </div>

      {/* Exam Date and Document Row */}
      <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
        <div>
```

```
            <label className="block text-gray-700 font-medium mb-2">Exam Date</label>
            <input
              type="date"
              className="w-full px-4 py-3 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500"
            />
          </div>
          <div>
            <label className="block text-gray-700 font-medium mb-2">Document</label>
            <input
              type="file"
              className="w-full px-4 py-3 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500 file:mr-4 file:py-2 file:px-4 file:rounded-md file:border-0 file:text-sm file:font-semibold file:bg-green-50 file:text-green-700 hover:file:bg-green-100"
            />
          </div>
        </div>

        {/* Remarks */}
        <div>
          <label className="block text-gray-700 font-medium mb-2">Remarks</label>
          <textarea
            rows="5"
            className="w-full px-4 py-3 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500 resize-none"
            placeholder="Enter any remarks or comments"
          ></textarea>
        </div>

        {/* Action Buttons */}
        <div className="flex justify-end gap-4 pt-4">
          <button
```

```
            type="button"
            className="px-8 py-3 bg-gray-200 text-gray-700 rounded-lg hover:bg-gray-300 transition-colors font-medium"
          >
            Cancel
          </button>
          <button
            type="submit"
            className="px-8 py-3 bg-purple-600 text-white rounded-lg hover:bg-purple-700 transition-colors font-medium"
          >
            Create Exam Score
          </button>
        </div>
      </form>
    </div>
  );

case 'settings':
  return (
    <div>
      <div className="mb-6">
        <h2 className="text-2xl font-bold text-gray-800 mb-2">Account Settings</h2>
        <p className="text-gray-600">Manage your profile and account preferences</p>
      </div>

      <div className="space-y-6">
        {/* Profile Picture Section */}
        <div className="bg-white rounded-lg shadow-md p-6">
          <h3 className="text-xl font-bold text-gray-800 mb-4 flex items-center gap-2">
            <User className="w-5 h-5 text-green-600" />
            Profile Picture
```

```
          </h3>
          <div className="flex items-center gap-6">
            <img
              src={trainerData.profilePicture || 'https://images.unsplash.com/photo-1472099645785-5658abf4ff4e?w=150&h=150&fit=crop'}
              alt="Profile"
              className="w-24 h-24 rounded-full object-cover border-4 border-green-600"
            />
            <div className="flex-1">
              <input
                type="file"
                accept="image/*"
                className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500 file:mr-4 file:py-2 file:px-4 file:rounded-md file:border-0 file:text-sm file:font-semibold file:bg-green-50 file:text-green-700 hover:file:bg-green-100"
              />
              <p className="text-sm text-gray-500 mt-2">Upload a new profile picture (JPG, PNG - Max 5MB)</p>
            </div>
          </div>
        </div>

        {/* Personal Information */}
        <div className="bg-white rounded-lg shadow-md p-6">
          <h3 className="text-xl font-bold text-gray-800 mb-4 flex items-center gap-2">
            <User className="w-5 h-5 text-green-600" />
            Personal Information
          </h3>
          <form className="space-y-4">
            <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
              <div>
                <label className="block text-gray-700 font-medium mb-2">Full Name</label>
```

```
                <input
                  type="text"
                  defaultValue={trainerData.name}
                  className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500"
                  placeholder="Enter your full name"
                />
              </div>
              <div>
                <label className="block text-gray-700 font-medium mb-2">Email</label>
                <input
                  type="email"
                  defaultValue={trainerData.email}
                  className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500"
                  placeholder="Enter your email"
                />
              </div>
            </div>
            <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
              <div>
                <label className="block text-gray-700 font-medium mb-2">Phone Number</label>
                <input
                  type="tel"
                  defaultValue={trainerData.phone}
                  className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500"
                  placeholder="Enter your phone number"
                />
              </div>
              <div>
                <label className="block text-gray-700 font-medium mb-2">Location</label>
```

```
                <input
                  type="text"
                  defaultValue={trainerData.location}
                  className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500"
                  placeholder="Enter your location"
                />
              </div>
            </div>
          </form>
        </div>

        {/* Expertise & Experience */}
        <div className="bg-white rounded-lg shadow-md p-6">
          <h3 className="text-xl font-bold text-gray-800 mb-4 flex items-center gap-2">
            <BookOpen className="w-5 h-5 text-green-600" />
            Expertise & Experience
          </h3>
          <form className="space-y-4">
            <div>
              <label className="block text-gray-700 font-medium mb-2">Specialization</label>
              <input
                type="text"
                defaultValue={trainerData.specialization}
                className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500"
                placeholder="e.g., Post-harvest Management, Cold Chain Operations"
              />
            </div>

            <div>
              <label className="block text-gray-700 font-medium mb-2">Courses Expertise</label>
```

```
            <div className="space-y-2">
              {coursesData.map((course, idx) => (
                <label key={idx} className="flex items-center gap-2 p-3 border border-gray-200 rounded-lg hover:bg-gray-50 cursor-pointer">
                  <input type="checkbox" className="w-4 h-4 text-green-600 rounded focus:ring-green-500" />
                  <span className="text-sm text-gray-700">{course.title}</span>
                </label>
              ))}
            </div>
          </div>

          <div>
            <label className="block text-gray-700 font-medium mb-2">Years of Experience</label>
            <input
              type="number"
              className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500"
              placeholder="Enter years of experience"
            />
          </div>

          <div>
            <label className="block text-gray-700 font-medium mb-2">Bio / About</label>
            <textarea
              rows="4"
              className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500 resize-none"
              placeholder="Tell us about your expertise and experience..."
            ></textarea>
          </div>
```

```
            <div>
              <label className="block text-gray-700 font-medium mb-2">Certifications</label>
              <input
                type="text"
                className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500"
                placeholder="Add certifications (comma separated)"
              />
              {trainerData.certifications && trainerData.certifications.length > 0 && (
                <div className="flex flex-wrap gap-2 mt-3">
                  {trainerData.certifications.map((cert, idx) => (
                    <span key={idx} className="px-3 py-1 bg-green-100 text-green-700 rounded-full text-xs font-medium flex items-center gap-2">
                      {cert}
                      <button type="button" className="hover:text-green-900">×</button>
                    </span>
                  ))}
                </div>
              )}
            </div>

            <div className="flex justify-end pt-2">
              <button
                type="submit"
                className="px-6 py-2 bg-green-600 text-white rounded-lg hover:bg-green-700 transition-colors font-medium"
              >
                Save Changes
              </button>
            </div>
          </form>
        </div>
```

```
{/* Password Settings */}
<div className="bg-white rounded-lg shadow-md p-6">
 <h3 className="text-xl font-bold text-gray-800 mb-4 flex items-center gap-2">
  <Settings className="w-5 h-5 text-green-600" />
  Password & Security
 </h3>
 <form className="space-y-4">
  <div>
   <label className="block text-gray-700 font-medium mb-2">Current Password</label>
   <input
    type="password"
    className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none
focus:ring-2 focus:ring-green-500"
    placeholder="Enter current password"
   />
  </div>
  <div>
   <label className="block text-gray-700 font-medium mb-2">New Password</label>
   <input
    type="password"
    className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none
focus:ring-2 focus:ring-green-500"
    placeholder="Enter new password"
   />
  </div>
  <div>
   <label className="block text-gray-700 font-medium mb-2">Confirm New
Password</label>
   <input
    type="password"
```

```
                className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-green-500"
                placeholder="Confirm new password"
              />
            </div>
            <div className="bg-blue-50 border border-blue-200 rounded-lg p-4">
              <p className="text-sm text-blue-800 mb-2">
                <strong>Password Requirements:</strong>
              </p>
              <ul className="text-xs text-blue-700 space-y-1 ml-4 list-disc">
                <li>At least 8 characters long</li>
                <li>Include uppercase and lowercase letters</li>
                <li>Include at least one number</li>
                <li>Include at least one special character</li>
              </ul>
            </div>
            <div className="flex justify-between items-center pt-2">
              <button
                type="button"
                className="text-blue-600 hover:text-blue-800 text-sm font-medium"
              >
                Forgot Password?
              </button>
              <button
                type="submit"
                className="px-6 py-2 bg-green-600 text-white rounded-lg hover:bg-green-700 transition-colors font-medium"
              >
                Update Password
              </button>
            </div>
          </form>
```

```
        </div>


        {/* Account Actions */}
        <div className="bg-white rounded-lg shadow-md p-6 border-2 border-red-200">
          <h3 className="text-xl font-bold text-gray-800 mb-4 flex items-center gap-2 text-red-600">
            <Award className="w-5 h-5" />
            Danger Zone
          </h3>
          <div className="space-y-3">
            <div className="flex justify-between items-center p-4 bg-red-50 rounded-lg">
              <div>
                <p className="font-medium text-gray-800">Deactivate Account</p>
                <p className="text-sm text-gray-600">Temporarily disable your account</p>
              </div>
              <button className="px-4 py-2 bg-red-100 text-red-700 rounded-lg hover:bg-red-200
transition-colors font-medium text-sm">
                Deactivate
              </button>
            </div>
            <div className="flex justify-between items-center p-4 bg-red-50 rounded-lg">
              <div>
                <p className="font-medium text-gray-800">Delete Account</p>
                <p className="text-sm text-gray-600">Permanently delete your account and all
data</p>
              </div>
              <button className="px-4 py-2 bg-red-600 text-white rounded-lg hover:bg-red-700
transition-colors font-medium text-sm">
                Delete
              </button>
            </div>
          </div>
        </div>
```

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

case 'quiz':
  return (
    <div>
      <div className="mb-6 flex items-center justify-between">
        <div>
          <h2 className="text-2xl font-bold text-gray-800 mb-2">Quiz Management</h2>
          <p className="text-gray-600">Create and manage quizzes for your courses</p>
        </div>
        <button
          onClick={() => setShowCreateQuiz(true)}
          className="flex items-center gap-2 px-6 py-3 bg-purple-600 text-white rounded-lg hover:bg-purple-700 transition-colors font-medium shadow-md"
        >
          <Plus className="w-5 h-5" />
          Create Quiz
        </button>
      </div>

      {/* Create Quiz Form */}
      {showCreateQuiz && (
        <div className="bg-white rounded-lg shadow-md p-6 mb-6 border-2 border-purple-200">
          <h3 className="text-xl font-bold text-gray-800 mb-4">Create New Quiz</h3>
          <form onSubmit={handleCreateQuiz} className="space-y-4">
            <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
              <div>
                <label className="block text-gray-700 font-medium mb-2">Quiz Title *</label>
                <input
```

```
                type="text"
                required
                value={newQuiz.title}
                onChange={(e) => setNewQuiz({ ...newQuiz, title: e.target.value })}
                className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-purple-500"
                placeholder="Enter quiz title"
              />
            </div>
            <div>
              <label className="block text-gray-700 font-medium mb-2">Select Course *</label>
              <select
                required
                value={newQuiz.course}
                onChange={(e) => setNewQuiz({ ...newQuiz, course: e.target.value })}
                className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-purple-500"
              >
                <option value="">Select a course</option>
                {coursesData.map((course) => (
                  <option key={course.id} value={course.title}>{course.title}</option>
                ))}
              </select>
            </div>
          </div>

          <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
            <div>
              <label className="block text-gray-700 font-medium mb-2">Duration *</label>
              <input
                type="text"
                required
```

```
                value={newQuiz.duration}
                onChange={(e) => setNewQuiz({ ...newQuiz, duration: e.target.value })}
                className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-purple-500"
                placeholder="e.g., 30 minutes"
              />
            </div>
            <div>
              <label className="block text-gray-700 font-medium mb-2">Total Marks *</label>
              <input
                type="number"
                required
                value={newQuiz.totalMarks}
                onChange={(e) => setNewQuiz({ ...newQuiz, totalMarks: e.target.value })}
                className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-purple-500"
                placeholder="100"
              />
            </div>
            <div>
              <label className="block text-gray-700 font-medium mb-2">Passing Marks *</label>
              <input
                type="number"
                required
                value={newQuiz.passingMarks}
                onChange={(e) => setNewQuiz({ ...newQuiz, passingMarks: e.target.value })}
                className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-purple-500"
                placeholder="50"
              />
            </div>
          </div>
```

```
            <div className="border-t pt-4">
              <h4 className="text-lg font-semibold text-gray-800 mb-3">Questions</h4>
              {newQuiz.questions.map((q, qIndex) => (
                <div key={qIndex} className="mb-6 p-4 border border-gray-200 rounded-lg bg-gray-
50">
                  <div className="flex justify-between items-center mb-3">
                    <h5 className="font-semibold text-gray-700">Question {qIndex + 1}</h5>
                    {newQuiz.questions.length > 1 && (
                      <button
                        type="button"
                        onClick={() => handleRemoveQuestion(qIndex)}
                        className="px-3 py-1 bg-red-100 text-red-600 rounded-lg hover:bg-red-200 text-
sm"
                      >
                        <Trash2 className="w-4 h-4" />
                      </button>
                    )}
                  </div>
                  <div className="space-y-3">
                    <input
                      type="text"
                      value={q.question}
                      onChange={(e) => handleQuestionChange(qIndex, 'question', e.target.value)}
                      className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none
focus:ring-2 focus:ring-purple-500"
                      placeholder="Enter question"
                    />
                    <div className="grid grid-cols-1 md:grid-cols-2 gap-2">
                      {q.options.map((opt, optIndex) => (
                        <input
                          key={optIndex}
```

```
                    type="text"
                    value={opt}
                    onChange={(e) => handleOptionChange(qIndex, optIndex, e.target.value)}
                    className="px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-purple-500"
                    placeholder={`Option ${optIndex + 1}`}
                  />
                ))}
              </div>
              <select
                value={q.correctAnswer}
                onChange={(e) => handleQuestionChange(qIndex, 'correctAnswer', e.target.value)}
                className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-purple-500"
              >
                <option value="">Select correct answer</option>
                {q.options.map((opt, idx) => (
                  <option key={idx} value={opt}>{opt || `Option ${idx + 1}`}</option>
                ))}
              </select>
            </div>
          </div>
        ))}
        <button
          type="button"
          onClick={handleAddQuestion}
          className="flex items-center gap-2 px-4 py-2 bg-purple-50 text-purple-600 rounded-lg hover:bg-purple-100"
        >
          <Plus className="w-4 h-4" />
          Add Question
        </button>
```

```
            </div>


            <div className="flex justify-end gap-3 pt-4 border-t">
              <button
                type="button"
                onClick={handleCancelQuiz}
                className="px-6 py-2 bg-gray-200 text-gray-700 rounded-lg hover:bg-gray-300
transition-colors"
              >
                Cancel
              </button>
              <button
                type="submit"
                className="px-6 py-2 bg-purple-600 text-white rounded-lg hover:bg-purple-700
transition-colors"
              >
                Create Quiz
              </button>
            </div>
          </form>
        </div>
      )}


      {/* Quiz List with Attempts */}
      <div className="space-y-6">
        {quizzes.map((quiz) => (
          <div key={quiz.id} className="bg-white rounded-lg shadow-md overflow-hidden">
            {/* Quiz Header */}
            <div className="bg-gradient-to-r from-purple-500 to-purple-600 p-6 text-white">
              <div className="flex items-start justify-between">
                <div className="flex-1">
                  <h3 className="text-xl font-bold mb-2">{quiz.title}</h3>
```

```
      <p className="text-purple-100 text-sm mb-3">{quiz.course}</p>
      <div className="flex flex-wrap gap-4 text-sm">
        <div className="flex items-center gap-1">
          <Clock className="w-4 h-4" />
          <span>{quiz.duration}</span>
        </div>
        <div className="flex items-center gap-1">
          <FileText className="w-4 h-4" />
          <span>{quiz.totalQuestions} Questions</span>
        </div>
        <div className="flex items-center gap-1">
          <Award className="w-4 h-4" />
          <span>{quiz.totalMarks} Marks</span>
        </div>
        <div className="flex items-center gap-1">
          <BarChart3 className="w-4 h-4" />
          <span>Pass: {quiz.passingMarks}</span>
        </div>
      </div>
    </div>
    <ClipboardList className="w-12 h-12 opacity-20" />
  </div>
</div>

{/* Trainee Attempts */}
<div className="p-6">
  <h4 className="text-lg font-semibold text-gray-800 mb-4">Trainee Attempts</h4>
  {quiz.attempts.length > 0 ? (
    <div className="overflow-x-auto">
      <table className="w-full">
        <thead>
```

```
                    <tr className="bg-gray-50 border-b">
                      <th className="px-4 py-3 text-left text-sm font-semibold text-gray-700">Trainee Name</th>
                      <th className="px-4 py-3 text-center text-sm font-semibold text-gray-700">Attempts</th>
                      <th className="px-4 py-3 text-center text-sm font-semibold text-gray-700">Last Score</th>
                      <th className="px-4 py-3 text-center text-sm font-semibold text-gray-700">Status</th>
                      <th className="px-4 py-3 text-center text-sm font-semibold text-gray-700">Actions</th>
                    </tr>
                  </thead>
                  <tbody>
                    {quiz.attempts.map((attempt, idx) => (
                      <tr key={idx} className="border-b hover:bg-gray-50">
                        <td className="px-4 py-3 text-sm text-gray-800">{attempt.traineeName}</td>
                        <td className="px-4 py-3 text-center">
                          <span className="inline-flex items-center justify-center w-8 h-8 bg-blue-100 text-blue-700 rounded-full font-semibold text-sm">
                            {attempt.attemptCount}
                          </span>
                        </td>
                        <td className="px-4 py-3 text-center">
                          <span className="font-semibold text-gray-800">{attempt.lastScore}/{quiz.totalMarks}</span>
                        </td>
                        <td className="px-4 py-3 text-center">
                          <span className={`px-3 py-1 rounded-full text-xs font-semibold ${
                            attempt.status === 'Passed'
                              ? 'bg-green-100 text-green-700'
                              : 'bg-red-100 text-red-700'
                          }`}>
```

```
                        {attempt.status}
                      </span>
                    </td>
                    <td className="px-4 py-3 text-center">
                      <button className="text-purple-600 hover:text-purple-800 text-sm font-
medium">
                        View Details
                      </button>
                    </td>
                  </tr>
                ))}
              </tbody>
            </table>
          </div>
        ) : (
          <div className="text-center py-8 text-gray-500">
            <Users className="w-12 h-12 mx-auto mb-3 opacity-50" />
            <p>No attempts yet</p>
          </div>
        )}
      </div>
    </div>
  ));

default:
  // Instructor submenu items
  const sections = {
    'batch': { title: 'Batches', icon: Users }
```

```
    };
    if (sections[activeMenu]) {
      const SectionIcon = sections[activeMenu].icon;
      return (
        <div className="bg-white rounded-lg shadow-md p-6">
          <div className="flex items-center gap-3 mb-6">
            <SectionIcon className="w-6 h-6 text-green-600" />
            <h2 className="text-2xl font-bold text-gray-800">{sections[activeMenu].title}</h2>
          </div>
          <p className="text-gray-600">Content for {sections[activeMenu].title} will be displayed here.</p>
        </div>
      );
    }

    // My Courses - Allotted by Admin
    if (activeMenu === 'my-course') {
      const allottedCourses = [
        {
          id: 1,
          courseName: "Post-harvest management of horticulture produce",
          batch: "Batch A - 2024",
          batchCode: "PH-A-2024",
          startDate: "2024-11-01",
          endDate: "2024-11-05",
          schedule: "Monday to Friday",
          timeSlot: "09:00 AM - 05:00 PM",
          totalStudents: 30,
          enrolledStudents: 28,
          location: "Training Center - Hall 1",
          status: "Ongoing",
```

```
  progress: 60,
  color: "from-green-500 to-green-600"
 },
 {
  id: 2,
  courseName: "Cold chain Operations",
  batch: "Batch B - 2024",
  batchCode: "CC-B-2024",
  startDate: "2024-11-15",
  endDate: "2024-11-19",
  schedule: "Monday to Friday",
  timeSlot: "10:00 AM - 06:00 PM",
  totalStudents: 25,
  enrolledStudents: 25,
  location: "Training Center - Hall 2",
  status: "Upcoming",
  progress: 0,
  color: "from-blue-500 to-blue-600"
 },
 {
  id: 3,
  courseName: "Agri logistics and retail distributions",
  batch: "Batch C - 2024",
  batchCode: "AL-C-2024",
  startDate: "2024-10-01",
  endDate: "2024-10-03",
  schedule: "Monday to Wednesday",
  timeSlot: "02:00 PM - 06:00 PM",
  totalStudents: 20,
  enrolledStudents: 20,
  location: "Online",
```

```
    status: "Completed",
    progress: 100,
    color: "from-orange-500 to-orange-600"
  },
  {
    id: 4,
    courseName: "Cold chain support schemes and incentives",
    batch: "Batch D - 2024",
    batchCode: "CS-D-2024",
    startDate: "2024-12-01",
    endDate: "2024-12-03",
    schedule: "Monday to Wednesday",
    timeSlot: "09:00 AM - 03:00 PM",
    totalStudents: 35,
    enrolledStudents: 15,
    location: "Training Center - Hall 3",
    status: "Enrollment Open",
    progress: 0,
    color: "from-purple-500 to-purple-600"
  }
];

return (
  <div>
    <div className="mb-6">
      <h2 className="text-2xl font-bold text-gray-800 mb-2">My Courses</h2>
      <p className="text-gray-600">Courses allotted to you by the administrator</p>
    </div>

    {/* Filter Tabs */}
    <div className="flex gap-2 mb-6 flex-wrap">
```

```
          {['All', 'Ongoing', 'Upcoming', 'Completed', 'Enrollment Open'].map((filter) => (
            <button
              key={filter}
              className={`px-4 py-2 rounded-lg font-medium transition-colors ${
                filter === 'All'
                  ? 'bg-green-600 text-white'
                  : 'bg-white text-gray-700 border border-gray-300 hover:bg-gray-50'
              }`}
            >
              {filter}
            </button>
          ))}
        </div>

        {/* Courses List */}
        <div className="space-y-6">
          {allottedCourses.map((course) => (
            <div key={course.id} className="bg-white rounded-lg shadow-md overflow-hidden hover:shadow-lg transition-shadow">
              {/* Course Header */}
              <div className={`bg-gradient-to-r ${course.color} p-6 text-white`}>
                <div className="flex items-start justify-between mb-4">
                  <div className="flex-1">
                    <div className="flex items-center gap-3 mb-2">
                      <span className="bg-white/20 px-3 py-1 rounded-full text-sm font-semibold">
                        {course.batchCode}
                      </span>
                      <span className={`px-3 py-1 rounded-full text-sm font-semibold ${
                        course.status === 'Ongoing' ? 'bg-green-500/30' :
                        course.status === 'Upcoming' ? 'bg-blue-500/30' :
                        course.status === 'Completed' ? 'bg-gray-500/30' :
```

```
          'bg-yellow-500/30'
        }`}>
          {course.status}
        </span>
      </div>
      <h3 className="text-xl font-bold mb-2">{course.courseName}</h3>
      <p className="text-white/90 text-sm">{course.batch}</p>
    </div>
    <BookOpen className="w-12 h-12 opacity-20" />
  </div>

  {/* Progress Bar for Ongoing Courses */}
  {course.status === 'Ongoing' && (
    <div className="mt-4">
      <div className="flex justify-between text-sm mb-1">
        <span>Course Progress</span>
        <span>{course.progress}%</span>
      </div>
      <div className="w-full bg-white/20 rounded-full h-2">
        <div
          className="bg-white rounded-full h-2 transition-all"
          style={{ width: `${course.progress}%` }}
        ></div>
      </div>
    </div>
  )}
</div>

{/* Course Details */}
<div className="p-6">
  <div className="grid md:grid-cols-2 gap-6">
```

```
{/* Schedule Details */}
<div>
 <h4 className="text-lg font-semibold text-gray-800 mb-3 flex items-center gap-2">
  <Clock className="w-5 h-5 text-green-600" />
  Schedule Details
 </h4>
 <div className="space-y-3">
  <div className="flex items-start gap-3">
   <div className="w-32 text-sm font-medium text-gray-600">Start Date:</div>
   <div className="text-sm text-gray-800 font-semibold">{course.startDate}</div>
  </div>
  <div className="flex items-start gap-3">
   <div className="w-32 text-sm font-medium text-gray-600">End Date:</div>
   <div className="text-sm text-gray-800 font-semibold">{course.endDate}</div>
  </div>
  <div className="flex items-start gap-3">
   <div className="w-32 text-sm font-medium text-gray-600">Schedule:</div>
   <div className="text-sm text-gray-800">{course.schedule}</div>
  </div>
  <div className="flex items-start gap-3">
   <div className="w-32 text-sm font-medium text-gray-600">Time Slot:</div>
   <div className="text-sm text-gray-800 font-semibold">{course.timeSlot}</div>
  </div>
  <div className="flex items-start gap-3">
   <div className="w-32 text-sm font-medium text-gray-600">Location:</div>
   <div className="text-sm text-gray-800">{course.location}</div>
  </div>
 </div>
</div>

{/* Batch Information */}
```

```
                <div>
                  <h4 className="text-lg font-semibold text-gray-800 mb-3 flex items-center gap-2">
                    <Users className="w-5 h-5 text-blue-600" />
                    Batch Information
                  </h4>
                  <div className="space-y-3">
                    <div className="flex items-start gap-3">
                      <div className="w-32 text-sm font-medium text-gray-600">Batch Code:</div>
                      <div className="text-sm text-gray-800 font-semibold">{course.batchCode}</div>
                    </div>
                    <div className="flex items-start gap-3">
                      <div className="w-32 text-sm font-medium text-gray-600">Total Capacity:</div>
                      <div className="text-sm text-gray-800">{course.totalStudents} students</div>
                    </div>
                    <div className="flex items-start gap-3">
                      <div className="w-32 text-sm font-medium text-gray-600">Enrolled:</div>
                      <div className="text-sm text-gray-800">
                        <span className="font-semibold">{course.enrolledStudents}</span> /
{course.totalStudents}
                      </div>
                    </div>
                    <div className="flex items-start gap-3">
                      <div className="w-32 text-sm font-medium text-gray-600">Availability:</div>
                      <div className="text-sm">
                        {course.enrolledStudents < course.totalStudents ? (
                          <span className="px-2 py-1 bg-green-100 text-green-700 rounded text-xs font-
semibold">
                            {course.totalStudents - course.enrolledStudents} seats available
                          </span>
                        ) : (
                          <span className="px-2 py-1 bg-red-100 text-red-700 rounded text-xs font-
semibold">
```

```
                Fully Booked
              </span>
            )}
          </div>
        </div>
        <div className="flex items-start gap-3">
          <div className="w-32 text-sm font-medium text-gray-600">Status:</div>
          <div className="text-sm">
            <span className={`px-3 py-1 rounded-full text-xs font-semibold ${
              course.status === 'Ongoing' ? 'bg-green-100 text-green-700' :
              course.status === 'Upcoming' ? 'bg-blue-100 text-blue-700' :
              course.status === 'Completed' ? 'bg-gray-100 text-gray-700' :
              'bg-yellow-100 text-yellow-700'
            }`}>
              {course.status}
            </span>
          </div>
        </div>
      </div>
    </div>

    {/* Action Buttons */}
    <div className="mt-6 pt-6 border-t flex gap-3 flex-wrap">
      <button className="px-4 py-2 bg-green-600 text-white rounded-lg hover:bg-green-700 transition-colors font-medium text-sm">
        View Course Details
      </button>
      <button className="px-4 py-2 bg-blue-600 text-white rounded-lg hover:bg-blue-700 transition-colors font-medium text-sm">
        View Students
      </button>
```

```
                <button className="px-4 py-2 border-2 border-purple-600 text-purple-600 rounded-lg hover:bg-purple-50 transition-colors font-medium text-sm">
                  Attendance
                </button>
                <button className="px-4 py-2 border-2 border-orange-600 text-orange-600 rounded-lg hover:bg-orange-50 transition-colors font-medium text-sm">
                  Materials
                </button>
                {course.status === 'Completed' && (
                  <button className="px-4 py-2 border-2 border-gray-600 text-gray-600 rounded-lg hover:bg-gray-50 transition-colors font-medium text-sm">
                    Course Report
                  </button>
                )}
              </div>
            </div>
          </div>
        ))}
      </div>
    </div>
  );
}

// Certificate Management
if (activeMenu === 'certificate') {
  return (
    <div>
      <div className="mb-6">
        <h2 className="text-2xl font-bold text-gray-800 mb-2">Certificate Management</h2>
        <p className="text-gray-600">Select a template and issue certificates to trainees</p>
      </div>
```

```
        {/* Issue Certificate Form */}
        {showIssueCertificate && selectedTemplate && (
          <div className="bg-white rounded-lg shadow-md p-6 mb-6 border-2 border-amber-200">
            <h3 className="text-xl font-bold text-gray-800 mb-4">Issue Certificate - {selectedTemplate.name}</h3>
            <form onSubmit={handleIssueCertificate} className="space-y-4">
              <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                <div>
                  <label className="block text-gray-700 font-medium mb-2">Trainee Name *</label>
                  <input
                    type="text"
                    required
                    value={certificateForm.traineeName}
                    onChange={(e) => setCertificateForm({ ...certificateForm, traineeName: e.target.value })}
                    className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-amber-500"
                    placeholder="Enter trainee name"
                  />
                </div>
                <div>
                  <label className="block text-gray-700 font-medium mb-2">Course *</label>
                  <select
                    required
                    value={certificateForm.course}
                    onChange={(e) => setCertificateForm({ ...certificateForm, course: e.target.value })}
                    className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-amber-500"
                  >
                    <option value="">Select course</option>
                    {coursesData.map((course) => (
                      <option key={course.id} value={course.title}>{course.title}</option>
```

```
                ))}
              </select>
            </div>
          </div>

          <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
            <div>
              <label className="block text-gray-700 font-medium mb-2">Completion Date *</label>
              <input
                type="date"
                required
                value={certificateForm.completionDate}
                onChange={(e) => setCertificateForm({ ...certificateForm, completionDate:
e.target.value })}
                className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none
focus:ring-2 focus:ring-amber-500"
              />
            </div>
            <div>
              <label className="block text-gray-700 font-medium mb-2">Grade *</label>
              <input
                type="text"
                required
                value={certificateForm.grade}
                onChange={(e) => setCertificateForm({ ...certificateForm, grade: e.target.value })}
                className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none
focus:ring-2 focus:ring-amber-500"
                placeholder="e.g., A, B+, 85%"
              />
            </div>
          </div>
```

```
{/* Certificate Preview */}
<div className="border-t pt-4">
  <h4 className="text-lg font-semibold text-gray-800 mb-3">Certificate Preview</h4>
  <div className={`bg-gradient-to-br ${selectedTemplate.color} border-4
${selectedTemplate.borderColor} rounded-lg p-8 text-center`}>
    <div className="mb-6">
      <Award className={`w-16 h-16 mx-auto mb-4 ${selectedTemplate.accentColor}`} />
      <h2 className={`text-3xl font-bold ${selectedTemplate.accentColor} mb-
2`}>Certificate of Completion</h2>
      <div className={`w-32 h-1 ${selectedTemplate.borderColor} bg-current mx-
auto`}></div>
    </div>
    <p className="text-gray-600 text-lg mb-4">This is to certify that</p>
    <h3 className={`text-2xl font-bold ${selectedTemplate.accentColor} mb-4`}>
      {certificateForm.traineeName || '[Trainee Name]'}
    </h3>
    <p className="text-gray-600 mb-4">has successfully completed the course</p>
    <h4 className={`text-xl font-semibold ${selectedTemplate.accentColor} mb-6`}>
      {certificateForm.course || '[Course Name]'}
    </h4>
    <div className="flex justify-around items-center text-sm text-gray-600 mt-8">
      <div>
        <p className="font-semibold">Date</p>
        <p>{certificateForm.completionDate || '[Date]'}</p>
      </div>
      <div>
        <p className="font-semibold">Grade</p>
        <p>{certificateForm.grade || '[Grade]'}</p>
      </div>
      <div>
        <p className="font-semibold">Certificate ID</p>
        <p>CERT{String(issuedCertificates.length + 1).padStart(3, '0')}</p>
```

```
                </div>
              </div>
            </div>
          </div>

          <div className="flex justify-end gap-3 pt-4 border-t">
            <button
              type="button"
              onClick={handleCancelCertificate}
              className="px-6 py-2 bg-gray-200 text-gray-700 rounded-lg hover:bg-gray-300 transition-colors"
            >
              Cancel
            </button>
            <button
              type="submit"
              className="px-6 py-2 bg-amber-600 text-white rounded-lg hover:bg-amber-700 transition-colors"
            >
              Issue Certificate
            </button>
          </div>
        </form>
      </div>
    )}

    {/* Certificate Templates */}
    {!showIssueCertificate && (
      <>
        <div className="mb-6">
          <h3 className="text-xl font-bold text-gray-800 mb-4">Choose a Certificate Template</h3>
```

```
          <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4">
            {certificateTemplates.map((template) => (
              <div
                key={template.id}
                className="bg-white rounded-lg shadow-md overflow-hidden hover:shadow-lg transition-shadow cursor-pointer"
                onClick={() => handleSelectTemplate(template)}
              >
                <div className={`bg-gradient-to-br ${template.color} border-4 ${template.borderColor} p-6 h-48 flex flex-col items-center justify-center`}>
                  <Award className={`w-12 h-12 ${template.accentColor} mb-3`} />
                  <h4 className={`text-lg font-bold ${template.accentColor} text-center`}>{template.name}</h4>
                </div>
                <div className="p-4">
                  <p className="text-sm text-gray-600 text-center mb-3">{template.description}</p>
                  <button className="w-full px-4 py-2 bg-amber-600 text-white rounded-lg hover:bg-amber-700 transition-colors text-sm font-medium">
                    Use Template
                  </button>
                </div>
              </div>
            ))}
          </div>
        </div>

        {/* Issued Certificates */}
        <div className="mt-8">
          <h3 className="text-xl font-bold text-gray-800 mb-4">Issued Certificates</h3>
          <div className="bg-white rounded-lg shadow-md overflow-hidden">
            {issuedCertificates.length > 0 ? (
              <div className="overflow-x-auto">
```

```
<table className="w-full">
 <thead>
  <tr className="bg-gray-50 border-b">
   <th className="px-4 py-3 text-left text-sm font-semibold text-gray-700">Certificate ID</th>
   <th className="px-4 py-3 text-left text-sm font-semibold text-gray-700">Trainee Name</th>
   <th className="px-4 py-3 text-left text-sm font-semibold text-gray-700">Course</th>
   <th className="px-4 py-3 text-left text-sm font-semibold text-gray-700">Template</th>
   <th className="px-4 py-3 text-center text-sm font-semibold text-gray-700">Grade</th>
   <th className="px-4 py-3 text-center text-sm font-semibold text-gray-700">Issued Date</th>
   <th className="px-4 py-3 text-center text-sm font-semibold text-gray-700">Status</th>
   <th className="px-4 py-3 text-center text-sm font-semibold text-gray-700">Actions</th>
  </tr>
 </thead>
 <tbody>
  {issuedCertificates.map((cert, idx) => {
   const template = certificateTemplates.find(t => t.id === cert.template);
   return (
    <tr key={idx} className="border-b hover:bg-gray-50">
     <td className="px-4 py-3 text-sm font-semibold text-gray-800">{cert.id}</td>
     <td className="px-4 py-3 text-sm text-gray-800">{cert.traineeName}</td>
     <td className="px-4 py-3 text-sm text-gray-600">{cert.course}</td>
     <td className="px-4 py-3 text-sm">
      <span className={`px-2 py-1 rounded text-xs font-medium ${template?.color} ${template?.borderColor} border`}>
       {template?.name}
      </span>
```

```
                    </td>
                    <td className="px-4 py-3 text-center text-sm font-semibold text-gray-800">{cert.grade}</td>
                    <td className="px-4 py-3 text-center text-sm text-gray-600">{cert.issuedDate}</td>
                    <td className="px-4 py-3 text-center">
                      <span className="px-3 py-1 rounded-full text-xs font-semibold bg-green-100 text-green-700">
                        {cert.status}
                      </span>
                    </td>
                    <td className="px-4 py-3 text-center">
                      <button className="text-amber-600 hover:text-amber-800 text-sm font-medium mr-2">
                        View
                      </button>
                      <button className="text-blue-600 hover:text-blue-800 text-sm font-medium">
                        Download
                      </button>
                    </td>
                  </tr>
                );
              })}
            </tbody>
          </table>
        </div>
      ) : (
        <div className="text-center py-8 text-gray-500">
          <Award className="w-12 h-12 mx-auto mb-3 opacity-50" />
          <p>No certificates issued yet</p>
        </div>
      )}
```

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

  return null;
 }
};

if (loading) {
 return (
  <div className="min-h-screen flex items-center justify-center bg-gray-50">
   <div className="text-center">
    <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-green-600 mx-auto"></div>
    <p className="mt-4 text-gray-600">Loading your dashboard...</p>
   </div>
  </div>
 );
}

if (!trainerData) return null;

return (
 <div className="min-h-screen bg-gray-50 flex">
  {/* Sidebar */}
  <div className={`${sidebarOpen ? 'w-64' : 'w-20'} bg-white shadow-lg transition-all duration-300 flex flex-col`}>
   <div className="p-4 border-b flex items-center justify-between">
```

```
        {sidebarOpen && (
          <div className="flex items-center gap-2">
            <div className="bg-green-600 rounded-full p-2">
              <svg className="w-6 h-6 text-white" viewBox="0 0 24 24" fill="currentColor">
                <path d="M12 2L4 7v6c0 5.55 3.84 10.74 9 12 5.16-1.26 9-6.45 9-12V7l-8-5z"/>
              </svg>
            </div>
            <span className="text-xl font-bold text-gray-800">AgriValue</span>
          </div>
        )}
        <button onClick={() => setSidebarOpen(!sidebarOpen)} className="p-2 hover:bg-gray-100 rounded-lg">
          {sidebarOpen ? <X className="w-5 h-5" /> : <Menu className="w-5 h-5" />}
        </button>
      </div>

      <nav className="flex-1 p-4 overflow-y-auto">
        {/* Dashboard */}
        <button
          onClick={() => { setActiveMenu('dashboard'); setShowInstructorSubmenu(false); setShowUserSubmenu(false); }}
          className={`w-full flex items-center gap-3 px-4 py-3 rounded-lg mb-2 transition-colors ${activeMenu === 'dashboard' ? 'bg-green-50 text-green-600' : 'text-gray-700 hover:bg-gray-50'}`}
        >
          <BarChart3 className="w-5 h-5" />
          {sidebarOpen && <span className="font-medium">Dashboard</span>}
        </button>

        {/* Noticeboard - Updated with navigation */}
        <button
          onClick={handleNoticeboardClick}
```

```
          className="w-full flex items-center gap-3 px-4 py-3 rounded-lg mb-2 transition-colors text-gray-700 hover:bg-gray-50"
        >
          <Bell className="w-5 h-5" />
          {sidebarOpen && <span className="font-medium">Noticeboard</span>}
        </button>

        {/* Training Courses */}
        <button
          onClick={() => { setActiveMenu('training-course'); setShowInstructorSubmenu(false); setShowUserSubmenu(false); }}
          className={`w-full flex items-center gap-3 px-4 py-3 rounded-lg mb-2 transition-colors ${activeMenu === 'training-course' ? 'bg-green-50 text-green-600' : 'text-gray-700 hover:bg-gray-50'}`}
        >
          <BookOpen className="w-5 h-5" />
          {sidebarOpen && <span className="font-medium">Training Courses</span>}
        </button>

        {/* Instructor Submenu */}
        <div className="mb-2">
          <button
            onClick={() => setShowInstructorSubmenu(!showInstructorSubmenu)}
            className="w-full flex items-center gap-3 px-4 py-3 rounded-lg text-gray-700 hover:bg-gray-50 transition-colors"
          >
            <BookOpen className="w-5 h-5" />
            {sidebarOpen && <span className="font-medium flex-1 text-left">Instructor</span>}
            {sidebarOpen && <span className="text-xs">{showInstructorSubmenu ? '▼' : '▶'}</span>}
          </button>
          {showInstructorSubmenu && sidebarOpen && (
            <div className="ml-4 mt-1 space-y-1">
              {[
```

```
            { id: 'my-course', label: 'My Courses', icon: BookOpen },
            { id: 'exam-score', label: 'Exam Score', icon: BarChart3 },
            { id: 'quiz', label: 'Quiz', icon: ClipboardList },
            { id: 'certificate', label: 'Certificate', icon: Award }
          ].map((item) => {
            const Icon = item.icon;
            return (
              <button
                key={item.id}
                onClick={() => setActiveMenu(item.id)}
                className={`w-full flex items-center gap-2 px-4 py-2 rounded-lg text-sm transition-colors ${activeMenu === item.id ? 'bg-green-50 text-green-600' : 'text-gray-600 hover:bg-gray-50'}`}
              >
                <Icon className="w-4 h-4" />
                {item.label}
              </button>
            );
          })}
        </div>
      )}
    </div>

    {/* User Submenu */}
    <div>
      <button
        onClick={() => setShowUserSubmenu(!showUserSubmenu)}
        className="w-full flex items-center gap-3 px-4 py-3 rounded-lg text-gray-700 hover:bg-gray-50 transition-colors"
      >
        <User className="w-5 h-5" />
        {sidebarOpen && <span className="font-medium flex-1 text-left">User</span>}
        {sidebarOpen && <span className="text-xs">{showUserSubmenu ? '▼' : '▶'}</span>}
```

```
          </button>
          {showUserSubmenu && sidebarOpen && (
            <div className="ml-4 mt-1 space-y-1">
              <button
                onClick={() => setActiveMenu('settings')}
                className={`w-full flex items-center gap-2 px-4 py-2 rounded-lg text-sm transition-colors ${activeMenu === 'settings' ? 'bg-green-50 text-green-600' : 'text-gray-600 hover:bg-gray-50'}`}
              >
                <Settings className="w-4 h-4" />
                Settings
              </button>
              <button
                onClick={handleLogout}
                className="w-full flex items-center gap-2 px-4 py-2 rounded-lg text-sm text-red-600 hover:bg-red-50 transition-colors"
              >
                <LogOut className="w-4 h-4" />
                Logout
              </button>
            </div>
          )}
        </div>
      </nav>
    </div>

    {/* Main Content */}
    <div className="flex-1 flex flex-col">
      {/* Top Bar */}
      <div className="bg-white shadow-sm p-4 flex items-center justify-between">
        <div className="flex items-center gap-4">
          <img
```

```
          src={trainerData.profilePicture || 'https://images.unsplash.com/photo-1472099645785-5658abf4ff4e?w=150&h=150&fit=crop'}
          alt="Profile"
          className="w-12 h-12 rounded-full object-cover border-2 border-green-600"
        />
        <div>
          <p className="text-lg font-semibold text-gray-800">Hello, {trainerData.name}!</p>
          <p className="text-sm text-gray-600">Welcome back to your dashboard</p>
        </div>
      </div>
      <div className="flex items-center gap-4">
        <button className="relative p-2 hover:bg-gray-100 rounded-full">
          <Bell className="w-6 h-6 text-gray-600" />
          <span className="absolute top-1 right-1 w-2 h-2 bg-red-500 rounded-full"></span>
        </button>
      </div>
    </div>

    {/* Content Area */}
    <div className="flex-1 p-6 overflow-auto">
      <div className="grid grid-cols-1 lg:grid-cols-3 gap-6">
        <div className="lg:col-span-2">{renderContent()}</div>

        {/* Trainer Details Sidebar */}
        <div className="lg:col-span-1">
          <div className="bg-white rounded-lg shadow-md p-6 sticky top-6">
            <h3 className="text-xl font-bold text-gray-800 mb-4">Trainer Details</h3>
            <div className="flex flex-col items-center mb-6">
              <img
                src={trainerData.profilePicture || 'https://images.unsplash.com/photo-1472099645785-5658abf4ff4e?w=150&h=150&fit=crop'}
                alt="Profile"
```

```
            className="w-24 h-24 rounded-full object-cover border-4 border-green-600 mb-3"
          />
          <h4 className="text-lg font-bold text-gray-800">{trainerData.name}</h4>
          <p className="text-sm text-gray-600">{trainerData.role || 'Trainer'}</p>
        </div>
        <div className="space-y-4">
          <div className="border-t pt-4">
            <p className="text-sm text-gray-500 mb-1">Email</p>
            <p className="text-sm font-medium text-gray-800 break-all">{trainerData.email}</p>
          </div>
          {trainerData.phone && <div><p className="text-sm text-gray-500 mb-1">Phone</p><p
className="text-sm font-medium text-gray-800">{trainerData.phone}</p></div>}
          {trainerData.location && <div><p className="text-sm text-gray-500 mb-
1">Location</p><p className="text-sm font-medium text-gray-
800">{trainerData.location}</p></div>}
          {trainerData.joinDate && <div><p className="text-sm text-gray-500 mb-1">Joined</p><p
className="text-sm font-medium text-gray-800">{trainerData.joinDate}</p></div>}
          {trainerData.specialization && <div><p className="text-sm text-gray-500 mb-
2">Specialization</p><p className="text-sm font-medium text-gray-
800">{trainerData.specialization}</p></div>}
          {trainerData.certifications && trainerData.certifications.length > 0 && (
            <div>
              <p className="text-sm text-gray-500 mb-2">Certifications</p>
              <div className="flex flex-wrap gap-2">
                {trainerData.certifications.map((cert, idx) => (
                  <span key={idx} className="px-3 py-1 bg-green-100 text-green-700 rounded-full text-
xs font-medium">{cert}</span>
                ))}
              </div>
            </div>
          )}
        </div>
      </div>
```

```
          </div>


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