```
import { useEffect, useState } from 'react';
import { useParams, useNavigate, Link } from 'react-router-dom';
import axios from 'axios';
import {
  BookOpenIcon,
  ClipboardDocumentListIcon,
  DocumentTextIcon,
  VideoCameraIcon,
  ArrowLeftIcon,
  ClockIcon,
  AcademicCapIcon,
  CheckCircleIcon,
  ExclamationTriangleIcon,
  PlusCircleIcon,
  XCircleIcon,
  PencilIcon,
  TrashIcon,
  PlusIcon,
  ChevronDownIcon,
  ChevronUpIcon,
  CalendarIcon,
  QuestionMarkCircleIcon,
  XMarkIcon,
  SpeakerWaveIcon
} from '@heroicons/react/24/outline';
import { PuzzlePieceIcon } from '@heroicons/react/24/solid';
import { Tab } from '@headlessui/react';
import { motion } from 'framer-motion';
import { useContext } from 'react';
```

```
import { AuthContext } from '../context/AuthContext.js';




const CourseDetails = () => {
  const { courseId } = useParams();
  const navigate = useNavigate();
  const [course, setCourse] = useState(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const userRole = localStorage.getItem('userRole');



const { user } = useContext(AuthContext);

function CourseDetails() {
  const { user } = useContext(AuthContext); // ✅ هذا هو المطلوب
}



  useEffect(() => {
    const fetchCourseDetails = async () => {
      try {
        const token = localStorage.getItem('accessToken');
        const response = await axios.get(`http://localhost:8080/courses/${courseId}`, {
          headers: {
            Authorization: `Bearer ${token}`
```

```
      }
    });
    setCourse(response.data);
   } catch (err) {
    setError('Failed to load course details');
    console.error('Error fetching course:', err);
   } finally {
    setLoading(false);
   }
  };

  fetchCourseDetails();
}, [courseId]);

const fetchLessons = async () => {
  try {
   const token = localStorage.getItem('accessToken');
   const response = await axios.get(`http://localhost:8080/lessons/course/${courseId}`, {
    headers: {
     Authorization: `Bearer ${token}`
    }
   });
   return response.data;
  } catch (err) {
   console.error('Error fetching lessons:', err);
   return [];
  }
};
```

```
const fetchAssignments = async () => {
  try {
    const token = localStorage.getItem('accessToken');
    const response = await axios.get(`http://localhost:8080/assignments/course/${courseId}`, {
      headers: {
        Authorization: `Bearer ${token}`
      }
    });
    return response.data;
  } catch (err) {
    console.error('Error fetching assignments:', err);
    return [];
  }
};

const fetchQuizzes = async () => {
  try {
    const token = localStorage.getItem('accessToken');
    const response = await axios.get(`http://localhost:8080/api/quizzes/course/${courseId}`, {
      headers: {
        Authorization: `Bearer ${token}`
      }
    });
    return response.data;
  } catch (err) {
    console.error('Error fetching quizzes:', err);
    return [];
  }
};
```

```
 if (loading) {
  return (
   <div className="flex justify-center items-center h-screen">
    <div className="animate-spin rounded-full h-12 w-12 border-t-2 border-b-2 border-blue-500"></div>
   </div>
  );
 }

 if (error) {
  return (
   <div className="flex justify-center items-center h-screen">
    <motion.div
     initial={{ opacity: 0, y: -20 }}
     animate={{ opacity: 1, y: 0 }}
     className="bg-red-100 border-l-4 border-red-500 text-red-700 p-4 max-w-md"
    >
     <p>{error}</p>
    </motion.div>
   </div>
  );
 }

 return (
  <div className="min-h-screen bg-gradient-to-br from-gray-50 to-gray-100 py-8 px-4 sm:px-6 lg:px-8">
   <div className="max-w-7xl mx-auto">
    <motion.button
```

```
        onClick={() => navigate(-1)}
        className="flex items-center text-gray-600 hover:text-gray-900 mb-6 transition-colors duration-200"
        whileHover={{ x: -3 }}
      >
        <ArrowLeftIcon className="h-5 w-5 mr-2" />
        Back to Courses
      </motion.button>

      <motion.div
        initial={{ opacity: 0, y: 20 }}
        animate={{ opacity: 1, y: 0 }}
        transition={{ duration: 0.5 }}
        className="bg-white rounded-xl shadow-lg overflow-hidden mb-8 border border-gray-100"
      >
        <div className="p-6 md:p-8">
          <div className="flex flex-col md:flex-row md:items-center md:justify-between">
            <div>
              <h1 className="text-3xl font-bold text-gray-900 mb-2">{course.title}</h1>
              <p className="text-gray-600 mb-4 max-w-3xl">{course.description}</p>
            </div>
            <div className="flex items-center space-x-2 bg-blue-50 px-4 py-2 rounded-lg">
              <AcademicCapIcon className="h-5 w-5 text-blue-600" />
              <span className="text-blue-800 font-medium">{course.instructorName}</span>
            </div>
          </div>
          <div className="flex flex-wrap items-center text-sm text-gray-500 mt-4 gap-4">
            <div className="flex items-center">
              <ClockIcon className="h-4 w-4 mr-1 text-gray-400" />
```

```
            <span>{course.duration} hours</span>
          </div>
          <div className="flex items-center">
            <CheckCircleIcon className="h-4 w-4 mr-1 text-green-500" />
            <span>Enrolled</span>
          </div>
        </div>
      </div>
    </motion.div>

    <Tab.Group>
      <Tab.List className="flex space-x-1 rounded-xl bg-white p-1 mb-8 shadow-sm border border-gray-200">
        {['lessons', 'assignments', 'quizzes'].map((tab) => (
          <Tab
            key={tab}
            className={({ selected }) =>
              `py-3 px-6 rounded-lg text-sm font-medium transition-all duration-200 flex items-center ${
                selected
                  ? 'bg-blue-600 text-white shadow-md'
                  : 'text-gray-600 hover:bg-gray-100 hover:text-gray-900'
              }`
            }
          >
            {tab === 'lessons' && (
              <>
                <VideoCameraIcon className="h-5 w-5 mr-2" />
                Lessons
              </>
```

```
          )}
          {tab === 'assignments' && (
            <>
              <DocumentTextIcon className="h-5 w-5 mr-2" />
              Assignments
            </>
          )}
          {tab === 'quizzes' && (
            <>
              <ClipboardDocumentListIcon className="h-5 w-5 mr-2" />
              Quizzes
            </>
          )}
        </Tab>
      ))}
    </Tab.List>

  <Tab.Panels>
    <Tab.Panel>
      <LessonsTab courseId={courseId} userRole={user?.role} />
    </Tab.Panel>
    <Tab.Panel>
      <AssignmentsTab courseId={courseId} userRole={user?.role} />
    </Tab.Panel>
    <Tab.Panel>
<QuizzesTab
  courseId={courseId}
  userRole={user?.role}
  fetchQuizzes={fetchQuizzes} // ← أضف هذا السطر
```

```
 />
</Tab.Panel>
        </Tab.Panels>
       </Tab.Group>
     </div>
   </div>
  );
};


const LessonsTab = ({ courseId, userRole }) => {
  const [lessons, setLessons] = useState([]);
  const [resources, setResources] = useState({});
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [expandedLesson, setExpandedLesson] = useState(null);
  const [currentVideo, setCurrentVideo] = useState(null);
  const [currentVideoTitle, setCurrentVideoTitle] = useState('');
  const [showCreateLessonModal, setShowCreateLessonModal] = useState(false);
  const [showAddResourceModal, setShowAddResourceModal] = useState(false);
  const [selectedLessonForResource, setSelectedLessonForResource] = useState(null);

  const toggleLesson = (lessonId) => {
    setExpandedLesson(expandedLesson === lessonId ? null : lessonId);
    setCurrentVideo(null);
  };

  const handleViewResource = async (resource) => {
    try {
```

```
const token = localStorage.getItem('accessToken');
const response = await axios.get(
  `http://localhost:8080/resources/download/${resource.id}`,
  {
    headers: {
      Authorization: `Bearer ${token}`
    },
    responseType: 'blob'
  }
);

const contentType = response.headers['content-type'];

if (contentType.startsWith('video/')) {
  const videoUrl = URL.createObjectURL(new Blob([response.data], { type: contentType }));
  setCurrentVideo(videoUrl);
  setCurrentVideoTitle(resource.title);
} else {
  const url = window.URL.createObjectURL(new Blob([response.data], { type: contentType }));
  const link = document.createElement('a');
  link.href = url;

  let extension = '.txt';
  if (contentType.includes('pdf')) extension = '.pdf';
  else if (contentType.includes('mpeg')) extension = '.mp3';
  else if (contentType.includes('jpeg')) extension = '.jpg';
  else if (contentType.includes('png')) extension = '.png';

  link.setAttribute('download', resource.title + extension);
```

```
      document.body.appendChild(link);
      link.click();
      link.remove();
    }
  } catch (err) {
    console.error('Error handling resource:', err);
    alert('Failed to process resource: ' + err.message);
  }
};

const fetchData = async () => {
  try {
    const token = localStorage.getItem('accessToken');
    if (!token) {
      throw new Error('Authentication token not found');
    }

    // Fetch lessons
    const lessonsResponse = await axios.get(
      `http://localhost:8080/lessons/course/${courseId}`,
      {
        headers: {
          Authorization: `Bearer ${token}`,
          'Content-Type': 'application/json'
        }
      }
    );

    setLessons(lessonsResponse.data);
```

```
    // Fetch resources for each lesson
    const resourcesData = {};
    await Promise.all(lessonsResponse.data.map(async (lesson) => {
      try {
        const resResponse = await axios.get(
          `http://localhost:8080/resources/lesson/${lesson.id}`,
          {
            headers: {
              Authorization: `Bearer ${token}`,
              'Content-Type': 'application/json'
            }
          }
        );
        resourcesData[lesson.id] = resResponse.data;
      } catch (resErr) {
        console.error(`Error fetching resources for lesson ${lesson.id}:`, resErr);
        resourcesData[lesson.id] = [];
      }
    }));

    setResources(resourcesData);
  } catch (err) {
    setError(err.response?.data?.message || err.message || 'Failed to load data. Please try again.');
  } finally {
    setLoading(false);
  }
};
```

```
  useEffect(() => {
    fetchData();
  }, [courseId]);

  const handleCreateLesson = () => {
    setShowCreateLessonModal(true);
  };

  const handleAddResource = (lessonId) => {
    setSelectedLessonForResource(lessonId);
    setShowAddResourceModal(true);
  };

  const handleLessonCreated = () => {
    setShowCreateLessonModal(false);
    fetchData();
  };

  const handleResourceAdded = () => {
    setShowAddResourceModal(false);
    fetchData();
  };

  if (loading) {
    return (
      <div className="flex justify-center py-12">
        <div className="animate-spin rounded-full h-10 w-10 border-t-2 border-b-2 border-blue-
500"></div>
      </div>
```

```
  );
}

if (error) {
  return (
    <div className="bg-red-50 border-l-4 border-red-500 p-4 mb-4 rounded">
      <div className="flex items-center">
        <ExclamationTriangleIcon className="h-5 w-5 text-red-500 mr-3" />
        <div>
          <p className="font-medium text-red-700">Error loading data</p>
          <p className="text-sm text-red-600">{error}</p>
          <button
            onClick={() => window.location.reload()}
            className="mt-2 text-sm text-blue-600 hover:text-blue-800"
          >
            Refresh Page
          </button>
        </div>
      </div>
    </div>
  );
}

return (
  <div className="space-y-4">
    <div className="flex justify-between items-center mb-4">
      <h2 className="text-xl font-semibold text-gray-800 flex items-center">
        <VideoCameraIcon className="h-5 w-5 mr-2 text-blue-500" />
        Lessons
```

```
      </h2>

      {(userRole === 'INSTRUCTOR' || userRole === 'ADMIN') && (
        <button
          onClick={handleCreateLesson}
          className="px-4 py-2 bg-blue-600 text-white rounded-md hover:bg-blue-700 flex items-center text-sm"
        >
          <PlusIcon className="h-4 w-4 mr-2" />
          Create Lesson
        </button>
      )}
    </div>

    {showCreateLessonModal && (
      <CreateLessonModal
        courseId={courseId}
        onClose={() => setShowCreateLessonModal(false)}
        onLessonCreated={handleLessonCreated}
      />
    )}

    {showAddResourceModal && (
      <AddResourceModal
        lessonId={selectedLessonForResource}
        onClose={() => setShowAddResourceModal(false)}
        onResourceAdded={handleResourceAdded}
      />
    )}
```

```
{lessons.length === 0 ? (
  <div className="bg-yellow-50 border-l-4 border-yellow-400 p-4 rounded">
    <div className="flex items-center">
      <ExclamationTriangleIcon className="h-5 w-5 text-yellow-500 mr-3" />
      <p className="text-yellow-700">No lessons available for this course.</p>
    </div>
  </div>
) : (
  lessons.map((lesson) => (
    <motion.div
      key={lesson.id}
      initial={{ opacity: 0, y: 20 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ duration: 0.3 }}
      className="bg-white rounded-lg shadow-md overflow-hidden border border-gray-100"
    >
      <button
        onClick={() => toggleLesson(lesson.id)}
        className="w-full p-5 text-left flex justify-between items-center hover:bg-gray-50 transition-colors duration-200"
      >
        <div className="flex items-start">
          <div className="flex-shrink-0 bg-blue-100 p-3 rounded-lg mr-4">
            <VideoCameraIcon className="h-6 w-6 text-blue-600" />
          </div>
          <div>
            <h3 className="text-lg font-semibold text-gray-800 mb-1">{lesson.title}</h3>
            <p className="text-gray-600 text-sm">{lesson.description}</p>
```

```
    </div>
  </div>
  <div className="text-gray-400">
    {expandedLesson === lesson.id ? '▲' : '▼'}
  </div>
</button>

{expandedLesson === lesson.id && (
  <motion.div
    initial={{ opacity: 0, height: 0 }}
    animate={{ opacity: 1, height: 'auto' }}
    transition={{ duration: 0.3 }}
    className="px-5 pb-5"
  >
    {lesson.videoUrl && (
      <div className="mb-6">
        <h4 className="font-medium text-gray-700 mb-3 flex items-center">
          <VideoCameraIcon className="h-5 w-5 mr-2 text-red-500" />
          Lesson Video
        </h4>
        <div className="rounded-lg overflow-hidden bg-black">
          <video
            controls
            className="w-full"
            src={lesson.videoUrl}
          >
            Your browser does not support the video tag.
          </video>
        </div>
```

```
  </div>
)}

{currentVideo && (
  <div className="mb-6">
    <h4 className="font-medium text-gray-700 mb-3 flex items-center">
      <VideoCameraIcon className="h-5 w-5 mr-2 text-red-500" />
      {currentVideoTitle}
    </h4>
    <div className="rounded-lg overflow-hidden bg-black">
      <video
        controls
        autoPlay
        className="w-full"
        src={currentVideo}
        onError={(e) => {
          console.error('Video playback error:', e);
          alert('Cannot play video. Please try downloading it.');
          setCurrentVideo(null);
        }}
      >
        Your browser does not support the video tag.
      </video>
    </div>
    <div className="mt-2 flex space-x-2">
      <button
        onClick={() => setCurrentVideo(null)}
        className="px-3 py-1 bg-gray-200 text-gray-700 rounded text-sm hover:bg-gray-300"
      >
```

```
        Close Video
      </button>
      <button
        onClick={() => {
          const link = document.createElement('a');
          link.href = currentVideo;
          link.setAttribute('download', currentVideoTitle + '.mp4');
          document.body.appendChild(link);
          link.click();
          link.remove();
        }}
        className="px-3 py-1 bg-blue-200 text-blue-700 rounded text-sm hover:bg-blue-300"
      >
        Download Video
      </button>
    </div>
  </div>
)}

<div className="border-t border-gray-100 pt-4">
  <div className="flex justify-between items-center mb-3">
    <h4 className="font-medium text-gray-700 flex items-center">
      <PuzzlePieceIcon className="h-5 w-5 mr-2 text-blue-500" />
      Additional Resources
    </h4>
    {(userRole === 'INSTRUCTOR' || userRole === 'ADMIN') && (
      <button
        onClick={() => handleAddResource(lesson.id)}
```

```
                  className="px-3 py-1 bg-green-600 text-white rounded text-sm hover:bg-green-700 flex
items-center"
                >
                  <PlusIcon className="h-3 w-3 mr-1" />
                  Add Resource
                </button>
              )}
            </div>

            {resources[lesson.id]?.length > 0 ? (
              <div className="grid grid-cols-1 md:grid-cols-2 gap-3">
                {resources[lesson.id].map((resource) => {
                  const type = resource.type?.toUpperCase();
                  let icon, actionLabel;

                  switch(type) {
                    case 'VIDEO':
                      icon = <VideoCameraIcon className="h-5 w-5 text-red-500" />;
                      actionLabel = 'View Video';
                      break;
                    case 'PDF':
                      icon = <DocumentTextIcon className="h-5 w-5 text-blue-500" />;
                      actionLabel = 'Download PDF';
                      break;
                    case 'AUDIO':
                      icon = <SpeakerWaveIcon className="h-5 w-5 text-purple-500" />;
                      actionLabel = 'Download Audio';
                      break;
                    case 'QUIZ':
```

```
                    icon = <PuzzlePieceIcon className="h-5 w-5 text-green-500" />;
                    actionLabel = 'Take Quiz';
                    break;
                  default:
                    icon = <DocumentTextIcon className="h-5 w-5 text-gray-500" />;
                    actionLabel = 'Download Resource';
                }

                return (
                  <div key={resource.id} className="bg-gray-50 rounded-lg p-3 hover:bg-gray-100
transition-colors">
                    <div className="flex items-start">
                      <div className="flex-shrink-0 mt-1">
                        {icon}
                      </div>
                      <div className="ml-3">
                        <h6 className="font-medium text-gray-800">{resource.title}</h6>
                        <button
                          onClick={() => handleViewResource(resource)}
                          className="text-sm text-blue-600 hover:underline flex items-center mt-1"
                        >
                          {actionLabel}
                        </button>
                      </div>
                    </div>
                  </div>
                );
              })}
            </div>
```

```
              ) : (
                <div className="text-sm text-gray-500 mt-2">
                  No additional resources available for this lesson.
                </div>
              )}
            </div>
          </motion.div>
        )}
      </motion.div>
    ))
  )}
 </div>
);
};

const CreateLessonModal = ({ courseId, onClose, onLessonCreated }) => {
 const [formData, setFormData] = useState({
  title: '',
  description: '',
  videoUrl: '',
  courseId: courseId
 });
 const [isSubmitting, setIsSubmitting] = useState(false);
 const [error, setError] = useState('');

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

```
const handleSubmit = async (e) => {
  e.preventDefault();
  setIsSubmitting(true);
  setError('');

  try {
    const token = localStorage.getItem('accessToken');

    const response = await axios.post(
      'http://localhost:8080/lessons/create',
      formData,
      {
        headers: {
          Authorization: `Bearer ${token}`,
          'Content-Type': 'application/json'
        }
      }
    );

    onLessonCreated();
    onClose();
  } catch (err) {
    console.error('Error creating lesson:', err);
    setError(err.response?.data?.message || 'Failed to create lesson');
  } finally {
    setIsSubmitting(false);
  }
};
```

```
return (
  <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center p-4 z-50">
    <div className="bg-white rounded-lg shadow-xl w-full max-w-2xl">
      <div className="flex justify-between items-center border-b p-4">
        <h3 className="text-lg font-semibold">Create New Lesson</h3>
        <button onClick={onClose} className="text-gray-500 hover:text-gray-700">
          <XMarkIcon className="h-6 w-6" />
        </button>
      </div>

      <form onSubmit={handleSubmit} className="p-4 space-y-4">
        {error && (
          <div className="bg-red-100 border-l-4 border-red-500 text-red-700 p-4">
            <p>{error}</p>
          </div>
        )}

        <div>
          <label className="block text-sm font-medium text-gray-700 mb-1">Lesson Title</label>
          <input
            type="text"
            name="title"
            value={formData.title}
            onChange={handleInputChange}
            className="w-full px-3 py-2 border border-gray-300 rounded-md"
            required
          />
        </div>
```

```
        <div>
          <label className="block text-sm font-medium text-gray-700 mb-1">Description</label>
          <textarea
            name="description"
            value={formData.description}
            onChange={handleInputChange}
            className="w-full px-3 py-2 border border-gray-300 rounded-md"
            rows="3"
          />
        </div>




        <div className="flex justify-end space-x-3 pt-4">
          <button
            type="button"
            onClick={onClose}
            className="px-4 py-2 border border-gray-300 rounded-md text-sm font-medium text-gray-700 hover:bg-gray-50"
          >
            Cancel
          </button>
          <button
            type="submit"
            disabled={isSubmitting}
            className="px-4 py-2 bg-blue-600 text-white rounded-md text-sm font-medium hover:bg-blue-700 disabled:opacity-50"
          >
```

```
            {isSubmitting ? 'Creating...' : 'Create Lesson'}
          </button>
        </div>
      </form>
    </div>
  </div>
 );
};

const AddResourceModal = ({ lessonId, onClose, onResourceAdded }) => {
 const [resourceFile, setResourceFile] = useState(null);
 const [resourceType, setResourceType] = useState('VIDEO');
 const [resourceTitle, setResourceTitle] = useState('');
 const [isSubmitting, setIsSubmitting] = useState(false);
 const [error, setError] = useState('');

 const handleFileChange = (e) => {
  setResourceFile(e.target.files[0]);
 };

 const handleSubmit = async (e) => {
  e.preventDefault();
  setIsSubmitting(true);
  setError('');

  try {
   const token = localStorage.getItem('accessToken');

   const formData = new FormData();
```

```
    formData.append('file', resourceFile);
    formData.append('lessonId', lessonId);
    formData.append('title', resourceTitle);
    formData.append('type', resourceType);

    await axios.post(
      'http://localhost:8080/resources/upload',
      formData,
      {
        headers: {
          Authorization: `Bearer ${token}`,
          'Content-Type': 'multipart/form-data'
        }
      }
    );

    onResourceAdded();
    onClose();
  } catch (err) {
    console.error('Error adding resource:', err);
    setError(err.response?.data?.message || 'Failed to add resource');
  } finally {
    setIsSubmitting(false);
  }
};

return (
  <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center p-4 z-50">
    <div className="bg-white rounded-lg shadow-xl w-full max-w-md">
```

```
<div className="flex justify-between items-center border-b p-4">
 <h3 className="text-lg font-semibold">Add Resource</h3>
 <button onClick={onClose} className="text-gray-500 hover:text-gray-700">
  <XMarkIcon className="h-6 w-6" />
 </button>
</div>

<form onSubmit={handleSubmit} className="p-4 space-y-4">
 {error && (
  <div className="bg-red-100 border-l-4 border-red-500 text-red-700 p-4">
   <p>{error}</p>
  </div>
 )}

 <div>
  <label className="block text-sm font-medium text-gray-700 mb-1">Resource Title</label>
  <input
   type="text"
   value={resourceTitle}
   onChange={(e) => setResourceTitle(e.target.value)}
   className="w-full px-3 py-2 border border-gray-300 rounded-md"
   required
  />
 </div>

 <div>
  <label className="block text-sm font-medium text-gray-700 mb-1">Resource Type</label>
  <select
   value={resourceType}
```

```
    onChange={(e) => setResourceType(e.target.value)}
    className="w-full px-3 py-2 border border-gray-300 rounded-md"
  >
    <option value="VIDEO">Video</option>
    <option value="PDF">PDF</option>
    <option value="AUDIO">Audio</option>

  </select>
</div>

<div>
  <label className="block text-sm font-medium text-gray-700 mb-1">File</label>
  <input
    type="file"
    onChange={handleFileChange}
    className="block w-full text-sm text-gray-500
      file:mr-4 file:py-2 file:px-4
      file:rounded-md file:border-0
      file:text-sm file:font-semibold
      file:bg-blue-50 file:text-blue-700
      hover:file:bg-blue-100"
    required
  />
</div>

<div className="flex justify-end space-x-3 pt-4">
  <button
    type="button"
    onClick={onClose}
```

```
            className="px-4 py-2 border border-gray-300 rounded-md text-sm font-medium text-gray-700 hover:bg-gray-50"
          >
            Cancel
          </button>
          <button
            type="submit"
            disabled={isSubmitting}
            className="px-4 py-2 bg-blue-600 text-white rounded-md text-sm font-medium hover:bg-blue-700 disabled:opacity-50"
          >
            {isSubmitting ? 'Adding...' : 'Add Resource'}
          </button>
        </div>
      </form>
    </div>
  </div>
 );
};



const AssignmentsTab = ({ courseId, userRole }) => {
 const [assignments, setAssignments] = useState([]);
 const [loading, setLoading] = useState(true);
 const [error, setError] = useState(null);
 const navigate = useNavigate();

 useEffect(() => {
  const fetchAssignments = async () => {
```

```
    try {
      const token = localStorage.getItem('accessToken');
      const response = await axios.get(`http://localhost:8080/assignments/course/${courseId}`, {
        headers: {
          Authorization: `Bearer ${token}`
        }
      });
      setAssignments(response.data);
    } catch (err) {
      setError('Failed to load assignments');
      console.error('Error fetching assignments:', err);
    } finally {
      setLoading(false);
    }
  };

  fetchAssignments();
 }, [courseId]);

 const handleViewAssignment = (assignmentId) => {
  navigate(`/assignments/${assignmentId}`);
 };

 if (loading) {
  return (
    <div className="flex justify-center py-12">
      <div className="animate-spin rounded-full h-10 w-10 border-t-2 border-b-2 border-blue-
500"></div>
    </div>
```

```
  );
 }

 if (error) {
  return (
   <div className="bg-red-50 border-l-4 border-red-500 p-4 mb-4 rounded">
    <div className="flex items-center">
     <ExclamationTriangleIcon className="h-5 w-5 text-red-500 mr-3" />
     <p className="text-red-700">{error}</p>
    </div>
   </div>
  );
 }

 return (
  <div className="space-y-6">
   <div className="flex justify-between items-center mb-4">
    <h2 className="text-xl font-semibold text-gray-800 flex items-center">
     <DocumentTextIcon className="h-5 w-5 mr-2 text-blue-500" />
     Assignments
    </h2>

    {(userRole === 'INSTRUCTOR' || userRole === 'ADMIN') && (
     <Link
      to={`/courses/${courseId}/create-assignment`}
      className="px-4 py-2 bg-blue-600 text-white rounded-md hover:bg-blue-700 flex items-center
text-sm"
     >
      <DocumentTextIcon className="h-4 w-4 mr-2" />
```

```
          Create Assignment
        </Link>
      )}
    </div>

    {assignments.length === 0 ? (
      <div className="bg-yellow-50 border-l-4 border-yellow-400 p-4 rounded">
        <div className="flex items-center">
          <ExclamationTriangleIcon className="h-5 w-5 text-yellow-500 mr-3" />
          <p className="text-yellow-700">No assignments available for this course.</p>
        </div>
      </div>
    ) : (
      <div className="space-y-4">
        {assignments.map((assignment, index) => (
          <motion.div
            key={assignment.id}
            initial={{ opacity: 0, y: 20 }}
            animate={{ opacity: 1, y: 0 }}
            transition={{ delay: index * 0.1 }}
            className="bg-white rounded-lg shadow-md p-5 border border-gray-100 hover:shadow-lg transition-shadow duration-300"
          >
            <div className="flex justify-between items-start">
              <div className="flex-1">
                <h3 className="text-lg font-semibold text-gray-800 mb-2">{assignment.title}</h3>
                <p className="text-gray-600 text-sm mb-3">{assignment.description}</p>

                <div className="flex items-center text-sm text-gray-500 mb-2">
```

```
              <ClockIcon className="h-4 w-4 mr-1" />
              <span>Due: {new Date(assignment.dueDate).toLocaleDateString()}</span>
              <span className="mx-2">•</span>
              <span>Total Marks: {assignment.totalMarks}</span>
            </div>
          </div>

          <div className={`px-3 py-1 rounded-full text-xs font-medium ml-4 ${
            assignment.status === 'SUBMITTED' ? 'bg-green-100 text-green-800' :
            assignment.status === 'LATE' ? 'bg-red-100 text-red-800' :
            'bg-blue-100 text-blue-800'
          }`}>
            {assignment.status || 'PENDING'}
          </div>
        </div>

        <div className="flex justify-end mt-4">
          <motion.button
            onClick={() => handleViewAssignment(assignment.id)}
            whileHover={{ scale: 1.05 }}
            whileTap={{ scale: 0.95 }}
            className="px-4 py-2 bg-gradient-to-r from-blue-600 to-blue-500 text-white rounded-md text-sm font-medium shadow-md hover:shadow-lg transition-all duration-200"
          >
            {userRole === 'STUDENT' ? 'Submit Assignment' : 'View Details'}
          </motion.button>
        </div>
      </motion.div>
    ))}
```

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

const QuizzesTab = ({ courseId, fetchQuizzes }) => {
  const [userRole, setUserRole] = useState(null);
  const [quizzes, setQuizzes] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [expandedQuiz, setExpandedQuiz] = useState(null);
  const [quizScores, setQuizScores] = useState({});
  const [editingQuestion, setEditingQuestion] = useState(null);
  const [editedText, setEditedText] = useState('');
  const [editedCorrectAnswer, setEditedCorrectAnswer] = useState('');
  const [editedOptions, setEditedOptions] = useState([]);
  const [newOption, setNewOption] = useState('');
  const [submissionStatus, setSubmissionStatus] = useState({});
  const [showCreateQuizModal, setShowCreateQuizModal] = useState(false);
  const [newQuiz, setNewQuiz] = useState({
    title: '',
    description: '',
    questions: [],
    startTime: '',
    endTime: ''
  });
  const [newQuestion, setNewQuestion] = useState({
    text: '',
```

```
  options: ['', ''],
  correctAnswer: ''
});
const navigate = useNavigate();

useEffect(() => {
  const token = localStorage.getItem('accessToken');
  if (token) {
    const payload = JSON.parse(atob(token.split('.')[1]));
    setUserRole(payload.role);
  } else {
    setUserRole('UNKNOWN');
  }
}, []);

useEffect(() => {
  const loadQuizzes = async () => {
    try {
      const data = await fetchQuizzes();
      setQuizzes(data);

      if (userRole === 'STUDENT') {
        const statuses = {};
        for (const quiz of data) {
          try {
            const token = localStorage.getItem('accessToken');
            const response = await axios.get(
              `http://localhost:8080/api/quizzes/${quiz.id}/submissions/check`,
              {
```

```
            headers: {
              Authorization: `Bearer ${token}`
            }
          }
        );
        statuses[quiz.id] = response.data;
      } catch (err) {
        console.error(`Error checking submission status for quiz ${quiz.id}:`, err);
        statuses[quiz.id] = false;
      }
    }
    setSubmissionStatus(statuses);
  }

  setError(null);
 } catch (err) {
  setError('Failed to load quizzes');
  console.error(err);
 } finally {
  setLoading(false);
 }
};

 if (userRole === 'ADMIN' || userRole === 'INSTRUCTOR' || userRole === 'STUDENT') {
  loadQuizzes();
 }
}, [courseId, fetchQuizzes, userRole]);

const toggleQuestions = (quizId) => {
```

```
  setExpandedQuiz(expandedQuiz === quizId ? null : quizId);
};

const handleEditQuestion = (quizId, questionIndex, question) => {
  setEditingQuestion({ quizId, questionIndex });
  setEditedText(question.text);
  setEditedCorrectAnswer(question.correctAnswer);
  setEditedOptions([...question.options]);
  setNewOption('');
};

const handleUpdateCorrectAnswer = async (questionId, newCorrectAnswer, quizId) => {
  try {
    const token = localStorage.getItem('accessToken');
    await axios.patch(
      `http://localhost:8080/api/quizzes/questions/${questionId}/correct-answer`,
      newCorrectAnswer,
      {
        headers: {
          Authorization: `Bearer ${token}`,
          'Content-Type': 'text/plain'
        }
      }
    );

    const updatedQuizzes = await fetchQuizzes();
    setQuizzes(updatedQuizzes);
    return true;
  } catch (error) {
```

```
    console.error('Error updating correct answer:', error);
    alert('Failed to update correct answer');
    return false;
  }
};

const handleSaveQuestion = async (quizId, questionIndex) => {
  try {
    const token = localStorage.getItem('accessToken');
    if (!token) throw new Error('Authentication token not found');

    if (!editedText.trim()) throw new Error('Question text cannot be empty');
    if (editedOptions.some(opt => !opt.trim())) throw new Error('All options must be filled');
    if (!editedOptions.includes(editedCorrectAnswer)) {
      throw new Error('Correct answer must be one of the options');
    }

    const currentQuiz = quizzes.find(q => q.id === quizId);
    if (!currentQuiz) throw new Error('Quiz not found');

    const questionId = currentQuiz.questions[questionIndex]?.id;
    if (!questionId) throw new Error('Question ID not found');

    const updateSuccess = await handleUpdateCorrectAnswer(
      questionId,
      editedCorrectAnswer,
      quizId
    );
```

```
  if (!updateSuccess) return;

  const updatedQuizzes = quizzes.map(quiz => {
    if (quiz.id === quizId) {
      const questions = [...quiz.questions];
      questions[questionIndex] = {
        ...questions[questionIndex],
        text: editedText,
        options: editedOptions,
        correctAnswer: editedCorrectAnswer
      };
      return { ...quiz, questions };
    }
    return quiz;
  });

  setQuizzes(updatedQuizzes);
  setEditingQuestion(null);
  alert('Question updated successfully!');
 } catch (err) {
  console.error('Error updating question:', {
    message: err.message,
    response: err.response?.data,
    stack: err.stack
  });

  try {
    const originalData = await fetchQuizzes();
    setQuizzes(originalData);
```

```
    } catch (fetchError) {
      console.error('Failed to reload quizzes:', fetchError);
    }

    alert(`Failed to update question: ${err.response?.data?.message || err.message}`);
  }
};

const handleCancelEdit = () => {
  setEditingQuestion(null);
};

const handleAddOption = () => {
  if (newOption.trim()) {
    setEditedOptions([...editedOptions, newOption.trim()]);
    setNewOption('');
  }
};

const handleRemoveOption = (index) => {
  if (editedOptions.length <= 2) {
    alert('A question must have at least 2 options');
    return;
  }
  const newOptions = [...editedOptions];
  newOptions.splice(index, 1);
  setEditedOptions(newOptions);
  if (editedCorrectAnswer === editedOptions[index]) {
    setEditedCorrectAnswer('');
```

```
  }
};

const handleCreateQuiz = async () => {
  try {
    const token = localStorage.getItem('accessToken');
    const quizData = {
      title: newQuiz.title,
      description: newQuiz.description,
      courseId: courseId,
      questions: newQuiz.questions,
      startTime: newQuiz.startTime,
      endTime: newQuiz.endTime
    };

    const response = await axios.post(
      'http://localhost:8080/api/quizzes',
      quizData,
      {
        headers: {
          Authorization: `Bearer ${token}`,
          'Content-Type': 'application/json'
        }
      }
    );

    setQuizzes([...quizzes, response.data]);
    setShowCreateQuizModal(false);
    setNewQuiz({
```

```
      title: '',
      description: '',
      questions: [],
      startTime: '',
      endTime: ''
    });
    alert('Quiz created successfully!');
  } catch (err) {
    console.error('Error creating quiz:', err);
    alert('Failed to create quiz');
  }
};

const addQuestionToQuiz = () => {
  if (!newQuestion.text.trim() || !newQuestion.correctAnswer.trim()) {
    alert('Please fill all question fields');
    return;
  }
  if (newQuestion.options.some(opt => !opt.trim())) {
    alert('All options must be filled');
    return;
  }
  if (!newQuestion.options.includes(newQuestion.correctAnswer)) {
    alert('Correct answer must be one of the options');
    return;
  }

  setNewQuiz(prev => ({
    ...prev,
```

```
    questions: [...prev.questions, {
      text: newQuestion.text,
      options: newQuestion.options,
      correctAnswer: newQuestion.correctAnswer
    }]
  }));

  setNewQuestion({
    text: '',
    options: ['', ''],
    correctAnswer: ''
  });
};

const addOptionToQuestion = () => {
  setNewQuestion(prev => ({
    ...prev,
    options: [...prev.options, '']
  }));
};

const removeOptionFromQuestion = (index) => {
  if (newQuestion.options.length <= 2) {
    alert('A question must have at least 2 options');
    return;
  }
  const newOptions = [...newQuestion.options];
  newOptions.splice(index, 1);
  setNewQuestion(prev => ({
```

```
    ...prev,
    options: newOptions,
    correctAnswer: prev.correctAnswer === newOptions[index] ? '' : prev.correctAnswer
  }));
};

const handleCloseQuiz = async (quizId) => {
  try {
    const token = localStorage.getItem('accessToken');
    await axios.patch(
      `http://localhost:8080/api/quizzes/${quizId}/close`,
      {},
      {
        headers: {
          Authorization: `Bearer ${token}`
        }
      }
    );

    setQuizzes(quizzes.map(quiz =>
      quiz.id === quizId ? { ...quiz, isClosed: true } : quiz
    ));
  } catch (err) {
    console.error('Error closing quiz:', err);
    alert('Failed to close quiz');
  }
};

const handleDeleteQuiz = async (quizId) => {
```

```
  if (!confirm('Are you sure you want to delete this quiz?')) return;

  try {
    const token = localStorage.getItem('accessToken');
    await axios.delete(
      `http://localhost:8080/api/quizzes/${quizId}`,
      {
        headers: {
          Authorization: `Bearer ${token}`
        }
      }
    );

    setQuizzes(quizzes.filter(quiz => quiz.id !== quizId));
  } catch (err) {
    console.error('Error deleting quiz:', err);
    alert('Failed to delete quiz');
  }
};

if (loading) {
  return (
    <div className="flex justify-center py-12">
      <div className="animate-spin rounded-full h-10 w-10 border-t-2 border-b-2 border-blue-
500"></div>
    </div>
  );
}
```

```
 if (error) {
  return (
   <div className="bg-red-50 border-l-4 border-red-500 p-4 mb-4 rounded">
    <div className="flex items-center">
     <ExclamationTriangleIcon className="h-5 w-5 text-red-500 mr-3" />
     <p className="text-red-700">{error}</p>
    </div>
   </div>
  );
 }

 if (userRole === 'STUDENT') {
  return (
   <div className="space-y-8">
    <div className="bg-gradient-to-r from-blue-500 to-purple-600 rounded-xl p-6 shadow-lg">
     <h1 className="text-3xl font-bold text-white mb-2">English Quizzes</h1>
     <p className="text-blue-100 text-lg">
      Test your knowledge with our interactive quizzes and track your progress!
     </p>
    </div>

    {quizzes.map((quiz) => (
     <motion.div
      key={quiz.id}
      initial={{ opacity: 0, y: 20 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ duration: 0.3 }}
      className="bg-white rounded-xl shadow-md overflow-hidden hover:shadow-lg transition-shadow"
```

```
>
 <div className="p-6">
  <div className="flex justify-between items-start">
   <div className="flex-1">
    <h3 className="text-xl font-bold text-gray-800 mb-2">{quiz.title}</h3>
    <p className="text-gray-600 mb-4">{quiz.description}</p>

    <div className="flex items-center space-x-4 text-sm">
     <div className="flex items-center text-blue-600">
      <CalendarIcon className="h-5 w-5 mr-1" />
      <span>{new Date(quiz.startTime).toLocaleDateString()}</span>
     </div>
     <div className="flex items-center text-purple-600">
      <ClockIcon className="h-5 w-5 mr-1" />
      <span>{new Date(quiz.startTime).toLocaleTimeString()}</span>
     </div>
    </div>
   </div>

   <div className={`px-3 py-1 rounded-full text-sm font-medium ${
    quiz.isClosed ? 'bg-red-100 text-red-800' : 'bg-green-100 text-green-800'
   }`}>
    {quiz.isClosed ? 'Closed' : 'Open'}
   </div>
  </div>

  {submissionStatus[quiz.id] ? (
   <div className="mt-6 bg-blue-50 p-4 rounded-lg border border-blue-200 flex items-start">
    <CheckCircleIcon className="h-6 w-6 text-blue-500 mt-1 mr-3" />
```

```
            <div>
              <h4 className="text-blue-800 font-semibold">Quiz Completed</h4>
              <p className="text-blue-600 text-sm">You've successfully submitted this quiz</p>
            </div>
          </div>
        ) : quiz.isClosed ? (
          <div className="mt-6 bg-red-50 p-4 rounded-lg border border-red-200 flex items-start">
            <XCircleIcon className="h-6 w-6 text-red-500 mt-1 mr-3" />
            <div>
              <h4 className="text-red-800 font-semibold">Quiz Closed</h4>
              <p className="text-red-600 text-sm">The submission period has ended</p>
            </div>
          </div>
        ) : (
          <motion.button
            onClick={() => navigate(`/quiz/${quiz.id}`)}
            className="mt-6 w-full py-3 bg-gradient-to-r from-blue-500 to-blue-600 text-white rounded-lg font-medium flex items-center justify-center space-x-2 hover:from-blue-600 hover:to-blue-700 transition-all"
            whileHover={{ scale: 1.02 }}
            whileTap={{ scale: 0.98 }}
          >
            <AcademicCapIcon className="h-5 w-5" />
            <span>Start Quiz Now</span>
          </motion.button>
        )}
      </div>
    </motion.div>
  ))}
```

```
    </div>
  );
 }

 return (
  <div className="space-y-8">
   <div className="bg-gradient-to-r from-indigo-500 to-purple-600 rounded-xl p-6 shadow-lg">
    <div className="flex justify-between items-center">
     <div>
      <h1 className="text-3xl font-bold text-white mb-2">Quiz Management</h1>
      <p className="text-indigo-100 text-lg">
       Create and manage quizzes for your students
      </p>
     </div>
     <motion.button
      onClick={() => setShowCreateQuizModal(true)}
      className="px-5 py-3 bg-white text-indigo-600 rounded-lg font-medium flex items-center space-x-2 hover:bg-gray-100 shadow-md"
      whileHover={{ scale: 1.05 }}
      whileTap={{ scale: 0.95 }}
     >
      <PlusIcon className="h-5 w-5" />
      <span>New Quiz</span>
     </motion.button>
    </div>
   </div>

   {quizzes.map((quiz) => (
    <motion.div
```

```
        key={quiz.id}
        initial={{ opacity: 0, y: 20 }}
        animate={{ opacity: 1, y: 0 }}
        transition={{ duration: 0.3 }}
        className="bg-white rounded-xl shadow-md overflow-hidden hover:shadow-lg transition-shadow"
      >
        <div className="p-6">
          <div className="flex justify-between items-start">
            <div className="flex-1">
              <div className="flex items-center space-x-3 mb-2">
                <h3 className="text-xl font-bold text-gray-800">{quiz.title}</h3>
                <span className={`px-2.5 py-0.5 rounded-full text-xs font-medium ${
                  quiz.isClosed ? 'bg-red-100 text-red-800' : 'bg-green-100 text-green-800'
                }`}>
                  {quiz.isClosed ? 'Closed' : 'Open'}
                </span>
              </div>

              <p className="text-gray-600 mb-4">{quiz.description}</p>

              <div className="flex flex-wrap gap-4 text-sm">
                <div className="flex items-center text-gray-500">
                  <CalendarIcon className="h-4 w-4 mr-1" />
                  <span>Starts: {new Date(quiz.startTime).toLocaleString()}</span>
                </div>
                <div className="flex items-center text-gray-500">
                  <CalendarIcon className="h-4 w-4 mr-1" />
                  <span>Ends: {new Date(quiz.endTime).toLocaleString()}</span>
```

```
    </div>
    <div className="flex items-center text-gray-500">
      <QuestionMarkCircleIcon className="h-4 w-4 mr-1" />
      <span>{quiz.questions?.length || 0} questions</span>
    </div>
  </div>
 </div>

 <button
  onClick={() => toggleQuestions(quiz.id)}
  className="text-indigo-600 hover:text-indigo-800 flex items-center text-sm font-medium"
 >
  {expandedQuiz === quiz.id ? (
   <>
    <ChevronUpIcon className="h-5 w-5 mr-1" />
    Hide Details
   </>
  ) : (
   <>
    <ChevronDownIcon className="h-5 w-5 mr-1" />
    View Details
   </>
  )}
 </button>
</div>

{expandedQuiz === quiz.id && (
 <motion.div
  initial={{ opacity: 0, height: 0 }}
```

```
          animate={{ opacity: 1, height: 'auto' }}
          transition={{ duration: 0.3 }}
          className="mt-4 pt-4 border-t"
        >
          <h4 className="font-medium text-gray-700 mb-3">Quiz Questions</h4>

          <div className="space-y-4">
            {quiz.questions?.map((question, index) => (
              <div key={index} className="p-4 bg-gray-50 rounded-lg border border-gray-200">
                {editingQuestion?.quizId === quiz.id && editingQuestion?.questionIndex === index ? (
                  <div className="space-y-4">
                    <div>
                      <label className="block text-sm font-medium text-gray-700 mb-1">Question Text</label>
                      <input
                        type="text"
                        value={editedText}
                        onChange={(e) => setEditedText(e.target.value)}
                        className="w-full p-2.5 border rounded-lg focus:ring-2 focus:ring-indigo-500 focus:border-indigo-500"
                      />
                    </div>

                    <div>
                      <label className="block text-sm font-medium text-gray-700 mb-1">Options</label>
                      <div className="space-y-2">
                        {editedOptions.map((option, optIndex) => (
                          <div key={optIndex} className="flex items-center">
                            <input
```

```
                    type="text"
                    value={option}
                    onChange={(e) => {
                      const newOptions = [...editedOptions];
                      newOptions[optIndex] = e.target.value;
                      setEditedOptions(newOptions);
                    }}
                    className="flex-1 p-2.5 border rounded-lg focus:ring-2 focus:ring-indigo-500
focus:border-indigo-500"
                  />
                  <button
                    onClick={() => handleRemoveOption(optIndex)}
                    className="ml-2 p-2 text-red-500 hover:text-red-700 rounded-full hover:bg-red-
50"
                  >
                    <TrashIcon className="h-5 w-5" />
                  </button>
                </div>
              ))}

              <div className="flex items-center mt-2">
                <input
                  type="text"
                  value={newOption}
                  onChange={(e) => setNewOption(e.target.value)}
                  className="flex-1 p-2.5 border rounded-lg focus:ring-2 focus:ring-indigo-500
focus:border-indigo-500"
                  placeholder="Add new option"
                />
                <button
```

```
                    onClick={handleAddOption}
                    className="ml-2 px-4 py-2.5 bg-indigo-100 text-indigo-700 rounded-lg hover:bg-indigo-200"
                  >
                    Add
                  </button>
                </div>
              </div>
            </div>

            <div>
              <label className="block text-sm font-medium text-gray-700 mb-1">Correct Answer</label>
              <select
                value={editedCorrectAnswer}
                onChange={(e) => setEditedCorrectAnswer(e.target.value)}
                className="w-full p-2.5 border rounded-lg focus:ring-2 focus:ring-indigo-500 focus:border-indigo-500"
              >
                <option value="">Select correct answer</option>
                {editedOptions.map((option, i) => (
                  <option key={i} value={option}>
                    {option || `Option ${i + 1}`}
                  </option>
                ))}
              </select>
            </div>

            <div className="flex justify-end space-x-3 pt-2">
              <button
```

```
          onClick={handleCancelEdit}
          className="px-4 py-2 bg-gray-200 text-gray-700 rounded-lg hover:bg-gray-300"
        >
          Cancel
        </button>
        <button
          onClick={() => handleSaveQuestion(quiz.id, index)}
          className="px-4 py-2 bg-indigo-600 text-white rounded-lg hover:bg-indigo-700"
        >
          Save Changes
        </button>
      </div>
    </div>
  ) : (
    <div className="flex justify-between items-start">
      <div>
        <p className="font-medium text-gray-800">
          {index + 1}. {question.text}
        </p>
        <div className="mt-2">
          <p className="text-sm text-gray-600">
            <span className="font-medium">Options:</span> {question.options.join(', ')}
          </p>
          <p className="text-sm text-green-600 mt-1">
            <span className="font-medium">Correct Answer:</span> {question.correctAnswer}
          </p>
        </div>
      </div>
      <button
```

```
                  onClick={() => handleEditQuestion(quiz.id, index, question)}
                  className="p-2 text-indigo-600 hover:text-indigo-800 rounded-full hover:bg-indigo-50"
                >
                  <PencilIcon className="h-5 w-5" />
                </button>
              </div>
            )}
          </div>
        ))}
      </div>

      <div className="mt-6 pt-4 border-t">
        <h4 className="font-medium text-gray-700 mb-3">Student Submissions</h4>

        {quiz.submissions && quiz.submissions.length > 0 ? (
          <div className="overflow-hidden rounded-lg border border-gray-200">
            <table className="min-w-full divide-y divide-gray-200">
              <thead className="bg-gray-50">
                <tr>
                  <th className="px-4 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Student</th>
                  <th className="px-4 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Score</th>
                  <th className="px-4 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Submitted</th>
                </tr>
              </thead>
              <tbody className="bg-white divide-y divide-gray-200">
                {quiz.submissions.map((submission) => (
                  <tr key={submission.studentId}>
```

```
                <td className="px-4 py-3 whitespace-nowrap text-sm font-medium text-gray-800">
                  {submission.studentName}
                </td>
                <td className="px-4 py-3 whitespace-nowrap text-sm">
                  <span className={`px-2 py-1 rounded-full text-xs font-medium ${
                    submission.score >= 80 ? 'bg-green-100 text-green-800' :
                    submission.score >= 50 ? 'bg-yellow-100 text-yellow-800' :
                    'bg-red-100 text-red-800'
                  }`}>
                    {submission.score}%
                  </span>
                </td>
                <td className="px-4 py-3 whitespace-nowrap text-sm text-gray-500">
                  {new Date(submission.submissionDate).toLocaleString()}
                </td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    ) : (
      <div className="text-center py-6 bg-gray-50 rounded-lg border border-dashed border-gray-
300">
        <DocumentTextIcon className="mx-auto h-12 w-12 text-gray-400" />
        <h4 className="mt-2 text-sm font-medium text-gray-700">No submissions yet</h4>
        <p className="mt-1 text-sm text-gray-500">Students haven't taken this quiz yet.</p>
      </div>
    )}
  </div>
```

```
            <div className="mt-6 pt-4 border-t flex justify-end space-x-3">
              <motion.button
                onClick={() => handleCloseQuiz(quiz.id)}
                disabled={quiz.isClosed}
                className={`px-5 py-2.5 rounded-lg text-sm font-medium ${
                  quiz.isClosed
                    ? 'bg-gray-200 text-gray-500 cursor-not-allowed'
                    : 'bg-amber-500 text-white hover:bg-amber-600'
                }`}
                whileHover={!quiz.isClosed ? { scale: 1.05 } : {}}
                whileTap={!quiz.isClosed ? { scale: 0.95 } : {}}
              >
                {quiz.isClosed ? 'Quiz Closed' : 'Close Quiz'}
              </motion.button>

              <motion.button
                onClick={() => handleDeleteQuiz(quiz.id)}
                className="px-5 py-2.5 rounded-lg text-sm font-medium bg-red-500 text-white hover:bg-red-600"
                whileHover={{ scale: 1.05 }}
                whileTap={{ scale: 0.95 }}
              >
                Delete Quiz
              </motion.button>
            </div>
          </motion.div>
        )}
      </div>
```

```
  </motion.div>
 ))}

 {showCreateQuizModal && (
  <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center p-4 z-50">
   <motion.div
    initial={{ opacity: 0, scale: 0.9 }}
    animate={{ opacity: 1, scale: 1 }}
    className="bg-white rounded-xl p-6 w-full max-w-2xl max-h-[90vh] overflow-y-auto shadow-xl"
   >
    <div className="flex justify-between items-center mb-4">
     <h2 className="text-2xl font-bold text-gray-800">Create New Quiz</h2>
     <button
      onClick={() => setShowCreateQuizModal(false)}
      className="text-gray-400 hover:text-gray-500"
     >
      <XMarkIcon  className="h-6 w-6" />
     </button>
    </div>

    <div className="space-y-4">
     <div>
      <label className="block text-sm font-medium text-gray-700 mb-1">Quiz Title</label>
      <input
       type="text"
       value={newQuiz.title}
       onChange={(e) => setNewQuiz({...newQuiz, title: e.target.value})}
       className="w-full p-2.5 border rounded-lg focus:ring-2 focus:ring-indigo-500 focus:border-
indigo-500"
```

```
            placeholder="Enter quiz title"
          />
        </div>

        <div>
          <label className="block text-sm font-medium text-gray-700 mb-1">Description</label>
          <textarea
            value={newQuiz.description}
            onChange={(e) => setNewQuiz({...newQuiz, description: e.target.value})}
            className="w-full p-2.5 border rounded-lg focus:ring-2 focus:ring-indigo-500 focus:border-indigo-500"
            placeholder="Enter quiz description"
            rows={3}
          />
        </div>

        <div className="grid grid-cols-2 gap-4">
          <div>
            <label className="block text-sm font-medium text-gray-700 mb-1">Start Time</label>
            <input
              type="datetime-local"
              value={newQuiz.startTime}
              onChange={(e) => setNewQuiz({...newQuiz, startTime: e.target.value})}
              className="w-full p-2.5 border rounded-lg focus:ring-2 focus:ring-indigo-500 focus:border-indigo-500"
            />
          </div>
          <div>
            <label className="block text-sm font-medium text-gray-700 mb-1">End Time</label>
```

```
            <input
              type="datetime-local"
              value={newQuiz.endTime}
              onChange={(e) => setNewQuiz({...newQuiz, endTime: e.target.value})}
              className="w-full p-2.5 border rounded-lg focus:ring-2 focus:ring-indigo-500 focus:border-indigo-500"
            />
          </div>
        </div>

        <div className="border-t pt-4">
          <h3 className="font-medium mb-2">Questions</h3>

          {newQuiz.questions.map((q, qIndex) => (
            <div key={qIndex} className="mb-4 p-3 bg-gray-50 rounded-lg border border-gray-200">
              <div className="flex justify-between items-start">
                <div>
                  <p className="font-medium">{qIndex + 1}. {q.text}</p>
                  <p className="text-sm text-green-600 mt-1">Correct Answer: {q.correctAnswer}</p>
                </div>
                <button
                  onClick={() => setNewQuiz({
                    ...newQuiz,
                    questions: newQuiz.questions.filter((_, i) => i !== qIndex)
                  })}
                  className="text-red-500 hover:text-red-700 p-1"
                >
                  <TrashIcon className="h-5 w-5" />
                </button>
```

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

      <div className="mt-4 p-4 border rounded-lg">
        <h4 className="font-medium mb-2">Add New Question</h4>

        <div className="mb-3">
          <label className="block text-sm font-medium text-gray-700 mb-1">Question Text</label>
          <input
            type="text"
            value={newQuestion.text}
            onChange={(e) => setNewQuestion({...newQuestion, text: e.target.value})}
            className="w-full p-2.5 border rounded-lg focus:ring-2 focus:ring-indigo-500 focus:border-indigo-500"
            placeholder="Enter question text"
          />
        </div>

        <div className="mb-3">
          <label className="block text-sm font-medium text-gray-700 mb-1">Options</label>
          {newQuestion.options.map((option, optIndex) => (
            <div key={optIndex} className="flex items-center mb-2">
              <input
                type="text"
                value={option}
                onChange={(e) => {
                  const newOptions = [...newQuestion.options];
                  newOptions[optIndex] = e.target.value;
```

```
                    setNewQuestion({...newQuestion, options: newOptions});
                  }}
                  className="flex-1 p-2.5 border rounded-lg focus:ring-2 focus:ring-indigo-500 focus:border-indigo-500"
                  placeholder={`Option ${optIndex + 1}`}
                />
                <button
                  onClick={() => removeOptionFromQuestion(optIndex)}
                  className="ml-2 p-2 text-red-500 hover:text-red-700 rounded-full hover:bg-red-50"
                >
                  <TrashIcon className="h-5 w-5" />
                </button>
              </div>
            ))}
            <button
              onClick={addOptionToQuestion}
              className="mt-1 text-indigo-600 hover:text-indigo-800 text-sm flex items-center"
            >
              <PlusIcon className="h-4 w-4 mr-1" />
              Add Option
            </button>
          </div>

          <div className="mb-3">
            <label className="block text-sm font-medium text-gray-700 mb-1">Correct Answer</label>
            <select
              value={newQuestion.correctAnswer}
              onChange={(e) => setNewQuestion({...newQuestion, correctAnswer: e.target.value})}
```

```
                  className="w-full p-2.5 border rounded-lg focus:ring-2 focus:ring-indigo-500 focus:border-indigo-500"
                >
                  <option value="">Select correct answer</option>
                  {newQuestion.options.map((option, i) => (
                    <option key={i} value={option} disabled={!option.trim()}>
                      {option || `Option ${i + 1}`}
                    </option>
                  ))}
                </select>
              </div>

              <button
                onClick={addQuestionToQuiz}
                className="px-4 py-2.5 bg-indigo-600 text-white rounded-lg hover:bg-indigo-700 flex items-center gap-2"
              >
                <PlusIcon className="h-5 w-5" />
                Add Question
              </button>
            </div>
          </div>
        </div>

        <div className="flex justify-end gap-3 mt-6">
          <button
            onClick={() => setShowCreateQuizModal(false)}
            className="px-4 py-2.5 bg-gray-200 text-gray-700 rounded-lg hover:bg-gray-300"
          >
```

```
              Cancel
            </button>
            <button
              onClick={handleCreateQuiz}
              disabled={!newQuiz.title.trim() || newQuiz.questions.length === 0}
              className="px-4 py-2.5 bg-indigo-600 text-white rounded-lg hover:bg-indigo-700 disabled:bg-gray-400"
            >
              Create Quiz
            </button>
          </div>
        </motion.div>
      </div>
    )}
  </div>
 );
};

export default CourseDetails;
```