**13. ReactJS-HOL**

**Implementation:**

**App.css:**

.App {

  text-align: center;

  padding: 20px;

  min-height: 100vh;

  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

}

.App-header {

  background-color: #282c34;

  padding: 20px;

  color: white;

  border-radius: 10px;

  margin-bottom: 20px;

  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

}

.auth-section {

  margin: 20px 0;

  padding: 15px;

  background-color: rgba(255, 255, 255, 0.1);

  border-radius: 8px;

}

.auth-section button {

  background-color: #61dafb;

  color: #282c34;

  border: none;

  padding: 10px 20px;

  border-radius: 5px;

  cursor: pointer;

  font-weight: bold;

  transition: all 0.3s ease;

}

.auth-section button:hover {

  background-color: #21a1c4;

  transform: translateY(-2px);

}

.navigation {

  margin: 20px 0;

  display: flex;

  justify-content: center;

  gap: 10px;

  flex-wrap: wrap;

}

.navigation button {

  background-color: #4CAF50;

  color: white;

  border: none;

  padding: 12px 24px;

  border-radius: 25px;

  cursor: pointer;

  transition: all 0.3s ease;

  font-weight: 500;

}

.navigation button:hover {

  background-color: #45a049;

  transform: translateY(-2px);

  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

}

.navigation button.active {

  background-color: #FF6B6B;

  transform: scale(1.05);

}

.content-area {

  margin-top: 30px;

  text-align: left;

}

.logged-in-content, .logged-out-content {

  background-color: white;

  padding: 30px;

  border-radius: 15px;

  box-shadow: 0 8px 16px rgba(0, 0, 0, 0.1);

  color: #333;

}

.additional-info {

  margin-top: 20px;

  padding: 15px;

  background-color: #e3f2fd;

  border-radius: 8px;

  border-left: 4px solid #2196F3;

}

.controls {

  display: flex;

  gap: 20px;

  margin: 20px 0;

  padding: 20px;

  background-color: #f5f5f5;

  border-radius: 10px;

  flex-wrap: wrap;

  align-items: center;

}

.filter-group {

  display: flex;

  flex-direction: column;

  gap: 5px;

}

.filter-group label {

  font-weight: 600;

  color: #333;

}

.filter-group select, .filter-group input[type="checkbox"] {

  padding: 8px;

  border: 1px solid #ddd;

  border-radius: 5px;

  font-size: 14px;

}

.filter-group input[type="checkbox"] {

  width: auto;

  margin-right: 8px;

}

.book-card, .blog-card, .course-card {

  background: white;

  border: 1px solid #e0e0e0;

  border-radius: 12px;

  padding: 20px;

  margin: 15px 0;

  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);

  transition: all 0.3s ease;

  cursor: pointer;

}

.book-card:hover, .blog-card:hover, .course-card:hover {

  transform: translateY(-5px);

  box-shadow: 0 4px 16px rgba(0, 0, 0, 0.15);

  border-color: #2196F3;

}

.books-grid, .blogs-container, .courses-grid {

  display: grid;

  grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));

  gap: 20px;

  margin: 20px 0;

}

.status {

  padding: 5px 12px;

  border-radius: 20px;

  font-size: 12px;

  font-weight: bold;

  display: inline-block;

  margin: 5px;

}

.status.available {

  background-color: #4CAF50;

  color: white;

}

.status.unavailable {

  background-color: #f44336;

  color: white;

}

.status.published {

  background-color: #2196F3;

  color: white;

}

.status.draft {

  background-color: #FF9800;

  color: white;

}

.bestseller, .popular-badge, .new-badge {

  background-color: #FFD700;

  color: #333;

  padding: 5px 10px;

  border-radius: 15px;

  font-size: 12px;

  font-weight: bold;

  display: inline-block;

  margin: 5px 0;

}

.new-badge {

  position: absolute;

  top: 10px;

  right: 10px;

  background-color: #FF6B6B;

  color: white;

}

.course-card {

  position: relative;

}

.level-badge {

  padding: 4px 8px;

  border-radius: 12px;

  font-size: 11px;

  font-weight: bold;

  margin-left: 10px;

}

.level-beginner {

  background-color: #4CAF50;

  color: white;

}

.level-intermediate {

  background-color: #FF9800;

  color: white;

}

.level-advanced {

  background-color: #f44336;

  color: white;

}

.price {

  padding: 8px 15px;

  border-radius: 20px;

  font-weight: bold;

  font-size: 14px;

}

.price.free {

  background-color: #4CAF50;

  color: white;

}

.price.paid {

  background-color: #2196F3;

  color: white;

}

.btn-primary, .btn-secondary, .btn-success, .btn-disabled, .close-btn {

  padding: 10px 20px;

  border: none;

  border-radius: 5px;

  cursor: pointer;

  font-weight: 500;

  margin: 5px;

  transition: all 0.3s ease;

}

.btn-primary {

  background-color: #2196F3;

  color: white;

}

.btn-primary:hover {

  background-color: #1976D2;

  transform: translateY(-1px);

}

.btn-secondary {

  background-color: #757575;

  color: white;

}

.btn-secondary:hover {

  background-color: #616161;

  transform: translateY(-1px);

}

.btn-success {

  background-color: #4CAF50;

  color: white;

}

.btn-success:hover {

  background-color: #45a049;

  transform: translateY(-1px);

}

.btn-disabled {

  background-color: #e0e0e0;

  color: #9e9e9e;

  cursor: not-allowed;

}

.close-btn {

  background-color: #f44336;

  color: white;

  position: absolute;

  top: 10px;

  right: 10px;

  padding: 5px 10px;

  font-size: 12px;

}

.progress-section {

  margin: 15px 0;

}

.progress-info {

  display: flex;

  justify-content: space-between;

  align-items: center;

  margin-bottom: 8px;

}

.progress-bar {

  width: 100%;

  height: 8px;

  background-color: #e0e0e0;

  border-radius: 4px;

  overflow: hidden;

}

.progress-fill {

  height: 100%;

  background: linear-gradient(90deg, #4CAF50, #8BC34A);

  transition: width 0.3s ease;

}

.completed-badge {

  background-color: #4CAF50;

  color: white;

  padding: 2px 8px;

  border-radius: 10px;

  font-size: 10px;

}

.book-details-modal {

  position: fixed;

  top: 0;

  left: 0;

  width: 100%;

  height: 100%;

  background-color: rgba(0, 0, 0, 0.5);

  display: flex;

  justify-content: center;

  align-items: center;

  z-index: 1000;

}

.modal-content {

  background: white;

  padding: 30px;

  border-radius: 15px;

  max-width: 500px;

  width: 90%;

  position: relative;

  box-shadow: 0 10px 30px rgba(0, 0, 0, 0.3);

}

.blog-stats, .summary-stats {

  display: flex;

  justify-content: space-around;

  margin: 20px 0;

  padding: 20px;

  background-color: #f8f9fa;

  border-radius: 10px;

}

.stat-item, .stat-card {

  text-align: center;

  padding: 15px;

  background: white;

  border-radius: 8px;

  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

  min-width: 100px;

}

.stat-number {

  display: block;

  font-size: 24px;

  font-weight: bold;

  color: #2196F3;

}

.stat-label {

  display: block;

  font-size: 12px;

  color: #666;

  margin-top: 5px;

}

.empty-state {

  text-align: center;

  padding: 40px;

  background-color: #f8f9fa;

  border-radius: 10px;

  margin: 20px 0;

}

.empty-state h3 {

  color: #666;

  margin-bottom: 10px;

}

.empty-state p {

  color: #999;

  margin-bottom: 20px;

}

.results-info, .book-count {

  padding: 10px 0;

  color: #666;

  font-style: italic;

  text-align: center;

}

.blog-header {

  display: flex;

  justify-content: space-between;

  align-items: flex-start;

  margin-bottom: 15px;

}

.blog-meta {

  margin: 15px 0;

  color: #666;

}

.blog-meta p {

  margin: 5px 0;

}

.blog-content, .blog-preview {

  margin: 15px 0;

  padding: 15px;

  background-color: #f8f9fa;

  border-radius: 8px;

}

.blog-actions, .course-actions {

  display: flex;

  gap: 10px;

  margin-top: 15px;

  flex-wrap: wrap;

}

.purchase-options, .unavailable-options {

  display: flex;

  gap: 10px;

  margin: 15px 0;

}

.course-header {

  display: flex;

  justify-content: space-between;

  align-items: flex-start;

  margin-bottom: 15px;

}

.course-meta {

  display: flex;

  gap: 10px;

  align-items: center;

}

.course-info {

  margin: 15px 0;

}

.course-info p {

  margin: 8px 0;

}

.rating-high {

  color: #FF9800;

  font-weight: bold;

}

.rating-normal {

  color: #666;

}

.certificate-info {

  background-color: #fff3cd;

  border: 1px solid #ffeaa7;

  padding: 10px;

  border-radius: 5px;

  margin: 10px 0;

  font-size: 14px;

}

@media (max-width: 768px) {

  .controls {

    flex-direction: column;

    gap: 15px;

  }

  .filter-group {

    width: 100%;

  }

  .books-grid, .blogs-container, .courses-grid {

    grid-template-columns: 1fr;

  }

  .navigation {

    flex-direction: column;

    gap: 10px;

  }

  .blog-stats, .summary-stats {

    flex-direction: column;

    gap: 15px;

  }

  .course-header, .blog-header {

    flex-direction: column;

    gap: 10px;

  }

  .course-meta {

    justify-content: flex-start;

  }

}

**App.js:**

import React, { useState } from 'react';

import BookDetails from './components/BookDetails';

import BlogDetails from './components/BlogDetails';

import CourseDetails from './components/CourseDetails';

import './App.css';

function App() {

  const [selectedComponent, setSelectedComponent] = useState('book');

  const [isLoggedIn, setIsLoggedIn] = useState(false);

  const [showContent, setShowContent] = useState(true);

  return (

    <div className="App">

      <header className="App-header">

        <h1>Blogger App - Conditional Rendering Demo</h1>

        <div className="auth-section">

          <button onClick={() => setIsLoggedIn(!isLoggedIn)}>

            {isLoggedIn ? 'Logout' : 'Login'}

          </button>

          <p>Status: {isLoggedIn ? 'Logged In' : 'Logged Out'}</p>

        </div>

        <div className="navigation">

          <button

            onClick={() => setSelectedComponent('book')}

            className={selectedComponent === 'book' ? 'active' : ''}

          >

          </button>

          <button

            onClick={() => setSelectedComponent('blog')}

            className={selectedComponent === 'blog' ? 'active' : ''}

          >

          </button>

          <button

            onClick={() => setSelectedComponent('course')}

            className={selectedComponent === 'course' ? 'active' : ''}

          >

          </button>

        </div>

        <button onClick={() => setShowContent(!showContent)}>

          {showContent ? 'Hide Content' : 'Show Content'}

        </button>

        <div className="content-area">

            {isLoggedIn ? (

            <div className="logged-in-content">

              <h2>Welcome! You are logged in.</h2>

                {showContent && (

                <div>

                  {selectedComponent === 'book' ? (

                    <BookDetails />

                  ) : selectedComponent === 'blog' ? (

                    <BlogDetails />

                  ) : selectedComponent === 'course' ? (

                    <CourseDetails />

                  ) : (

                    <div>Select a component to view</div>

                  )}

                </div>

              )}

              {renderAdditionalContent(selectedComponent)}

            </div>

          ) : (

            <div className="logged-out-content">

              <h2>Please log in to view content</h2>

              <p>Click the Login button to access the blogger app features.</p>

            </div>

          )}

        </div>

      </header>

    </div>

  );}

function renderAdditionalContent(component) {

  const additionalInfo = {

    book: "Explore our vast collection of books!",

    blog: "Read the latest blog posts from our community!",

    course: "Enroll in our comprehensive courses!"

  };

  return (

    <div className="additional-info">

      <p>{additionalInfo[component]}</p>

    </div>

  );

}

export default App;

**BlogDetails.js**

import React, { useState } from 'react';

const BlogDetails = () => {

  const [blogs, setBLogs] = useState([

    {

      id: 1,

      title: 'Getting Started with React Hooks',

      author: 'Tech Writer',

      published: true,

      category: 'React',

      likes: 45,

      publishDate: '2024-01-15'

    },

    {

      id: 2,

      title: 'Advanced State Management',

      author: 'Code Master',

      published: false,

      category: 'React',

      likes: 0,

      publishDate: null

    },

    {

      id: 3,

      title: 'JavaScript ES2024 Features',

      author: 'JS Expert',

      published: true,

      category: 'JavaScript',

      likes: 78,

      publishDate: '2024-02-01'

    },

    {

      id: 4,

      title: 'CSS Grid vs Flexbox',

      author: 'Design Guru',

      published: true,

      category: 'CSS',

      likes: 32,

      publishDate: '2024-01-28'

    }

  ]);

  const [filter, setFilter] = useState('all'); // all, published, draft

  const [selectedCategory, setSelectedCategory] = useState('all');

  const [expandedBlog, setExpandedBlog] = useState(null);

  const categories = ['all', ...new Set(blogs.map(blog => blog.category))];

  const getFilteredBlogs = () => {

    let filtered = blogs;

    if (filter === 'published') {

      filtered = filtered.filter(blog => blog.published);

    } else if (filter === 'draft') {

      filtered = filtered.filter(blog => !blog.published);

    }

    if (selectedCategory !== 'all') {

      filtered = filtered.filter(blog => blog.category === selectedCategory);

    }

    return filtered;

  };

  const filteredBlogs = getFilteredBlogs();

  const renderBlogStatus = (blog) => {

    return blog.published ? (

      <span className="status published">📝 Published</span>

    ) : (

      <span className="status draft">✏️ Draft</span>

    );

  };

  const renderBlogCard = (blog) => {

    const isExpanded = expandedBlog === blog.id;

    return (

      <div key={blog.id} className="blog-card">

        <div className="blog-header">

          <h3>{blog.title}</h3>

          {renderBlogStatus(blog)}

        </div>

        <div className="blog-meta">

          <p><strong>Author:</strong> {blog.author}</p>

          <p><strong>Category:</strong> {blog.category}</p>

            {blog.publishDate && (

            <p><strong>Published:</strong> {blog.publishDate}</p>

          )}

            {blog.published && (

            <p><strong>Likes:</strong> ❤️ {blog.likes}</p>

          )}

        </div>

        {blog.likes > 50 && (

          <div className="popular-badge">🔥 Popular Post!</div>

        )}

        {isExpanded ? (

          <div className="blog-content">

            <p>This is the full content of the blog post. In a real application, this would contain the actual blog content.</p>

            <button onClick={() => setExpandedBlog(null)}>Show Less</button>

          </div>

        ) : (

          <div className="blog-preview">

            <p>This is a preview of the blog post...</p>

            <button onClick={() => setExpandedBlog(blog.id)}>Read More</button>

          </div>

        )}

        <div className="blog-actions">

          {blog.published ? (

            <>

              <button className="btn-secondary">Share</button>

              <button className="btn-secondary">Comment</button>

            </>

          ) : (

            <>

              <button className="btn-primary">Edit Draft</button>

              <button className="btn-success">Publish</button>

            </>

          )}

        </div>

      </div>

    );

  };

  const renderStats = () => {

    const totalBlogs = blogs.length;

    const publishedBlogs = blogs.filter(blog => blog.published).length;

    const draftBlogs = totalBlogs - publishedBlogs;

    return (

      <div className="blog-stats">

        <div className="stat-item">

          <span className="stat-number">{totalBlogs}</span>

          <span className="stat-label">Total Blogs</span>

        </div>

        <div className="stat-item">

          <span className="stat-number">{publishedBlogs}</span>

          <span className="stat-label">Published</span>

        </div>

        <div className="stat-item">

          <span className="stat-number">{draftBlogs}</span>

          <span className="stat-label">Drafts</span>

        </div>

      </div>

    );

  };

  return (

    <div className="blog-details">

      <h2>📖 Blog Details Component</h2>

       {renderStats()}

        <div className="controls">

        <div className="filter-group">

          <label>Status Filter:</label>

          <select value={filter} onChange={(e) => setFilter(e.target.value)}>

            <option value="all">All Posts</option>

            <option value="published">Published Only</option>

            <option value="draft">Drafts Only</option>

          </select>

        </div>

          <div className="filter-group">

          <label>Category Filter:</label>

          <select value={selectedCategory} onChange={(e) => setSelectedCategory(e.target.value)}>

            {categories.map(category => (

              <option key={category} value={category}>

                {category.charAt(0).toUpperCase() + category.slice(1)}

              </option>

            ))}

          </select>

        </div>

      </div>

      {filteredBlogs.length === 0 ? (

        <div className="empty-state">

          <h3>No blogs found</h3>

          <p>Try adjusting your filters to see more results.</p>

        </div>

      ) : (

        <>

          <div className="results-info">

            <p>Showing {filteredBlogs.length} of {blogs.length} blogs</p>

          </div>

          <div className="blogs-container">

            {filteredBlogs.map(blog => renderBlogCard(blog))}

          </div>

        </>

      )}

    </div>

  );

};

export default BlogDetails;

**BookDetails:**

import React, { useState } from 'react';

const BookDetails = () => {

  const [books, setBooks] = useState([

    { id: 1, title: 'React Fundamentals', author: 'John Doe', available: true, rating: 4.5 },

    { id: 2, title: 'Advanced JavaScript', author: 'Jane Smith', available: false, rating: 4.8 },

    { id: 3, title: 'Node.js Guide', author: 'Bob Johnson', available: true, rating: 4.2 },

    { id: 4, title: 'CSS Mastery', author: 'Alice Brown', available: true, rating: 4.0 }

  ]);

  const [showOnlyAvailable, setShowOnlyAvailable] = useState(false);

  const [selectedBook, setSelectedBook] = useState(null);

  const filteredBooks = showOnlyAvailable

    ? books.filter(book => book.available)

    : books;

  const renderBookCard = (book) => {

    return (

      <div key={book.id} className="book-card" onClick={() => setSelectedBook(book)}>

        <h3>{book.title}</h3>

        <p><strong>Author:</strong> {book.author}</p>

        <p><strong>Rating:</strong> {book.rating}/5</p>

          {book.available ? (

          <span className="status available">✅ Available</span>

        ) : (

          <span className="status unavailable">❌ Out of Stock</span>

        )}

          {book.rating >= 4.5 && (

          <div className="bestseller">🏆 Bestseller!</div>

        )}

      </div>

    );

  };

  const renderBookDetails = () => {

    if (!selectedBook) return null;

    return (

      <div className="book-details-modal">

        <div className="modal-content">

          <h2>{selectedBook.title}</h2>

          <p><strong>Author:</strong> {selectedBook.author}</p>

          <p><strong>Rating:</strong> {selectedBook.rating}/5</p>

          <p><strong>Status:</strong> {selectedBook.available ? 'Available' : 'Out of Stock'}</p>

            {selectedBook.available ? (

            <div className="purchase-options">

              <button className="btn-primary">Buy Now</button>

              <button className="btn-secondary">Add to Cart</button>

            </div>

          ) : (

            <div className="unavailable-options">

              <button className="btn-disabled" disabled>Out of Stock</button>

              <button className="btn-secondary">Notify When Available</button>

            </div>

          )}

            <button onClick={() => setSelectedBook(null)} className="close-btn">

            Close

          </button>

        </div>

      </div>

    );

  };

  return (

    <div className="book-details">

      <h2>📚 Book Details Component</h2>

        <div className="controls">

        <label>

          <input

            type="checkbox"

            checked={showOnlyAvailable}

            onChange={(e) => setShowOnlyAvailable(e.target.checked)}

          />

          Show only available books

        </label>

      </div>

      {filteredBooks.length === 0 ? (

        <div className="empty-state">

          <p>No books found matching your criteria.</p>

        </div>

      ) : (

        <div className="books-grid">

          {filteredBooks.map(book => renderBookCard(book))}

        </div>

      )}

      {filteredBooks.length > 0 && (

        <div className="book-count">

          <p>Showing {filteredBooks.length} out of {books.length} books</p>

        </div>

      )}

      {selectedBook && renderBookDetails()}

    </div>

  );

};

export default BookDetails;

**CourseDetails.js:**

import React, { useState } from 'react';

const CourseDetails = () => {

  const [courses, setCourses] = useState([

    {

      id: 1,

      title: 'Complete React Development',

      instructor: 'Sarah Johnson',

      level: 'Beginner',

      duration: '40 hours',

      price: 99.99,

      enrolled: true,

      completed: false,

      progress: 65,

      rating: 4.8,

      students: 15420,

      isNew: false,

      hasCertificate: true

    },

    {

      id: 2,

      title: 'Advanced JavaScript Patterns',

      instructor: 'Mike Chen',

      level: 'Advanced',

      duration: '25 hours',

      price: 149.99,

      enrolled: false,

      completed: false,

      progress: 0,

      rating: 4.9,

      students: 8750,

      isNew: true,

      hasCertificate: true

    },

    {

      id: 3,

      title: 'CSS Grid and Flexbox Mastery',

      instructor: 'Emma Davis',

      level: 'Intermediate',

      duration: '15 hours',

      price: 0,

      enrolled: true,

      completed: true,

      progress: 100,

      rating: 4.6,

      students: 22340,

      isNew: false,

      hasCertificate: false

    },

    {

      id: 4,

      title: 'Node.js Backend Development',

      instructor: 'Alex Rodriguez',

      level: 'Intermediate',

      duration: '35 hours',

      price: 129.99,

      enrolled: false,

      completed: false,

      progress: 0,

      rating: 4.7,

      students: 12680,

      isNew: false,

      hasCertificate: true

    }

  ]);

  const [viewMode, setViewMode] = useState('all'); // all, enrolled, available

  const [selectedLevel, setSelectedLevel] = useState('all');

  const [showFreeOnly, setShowFreeOnly] = useState(false);

  const levels = ['all', ...new Set(courses.map(course => course.level))];

  const getFilteredCourses = () => {

    let filtered = courses;

    switch (viewMode) {

      case 'enrolled':

        filtered = filtered.filter(course => course.enrolled);

        break;

      case 'available':

        filtered = filtered.filter(course => !course.enrolled);

        break;

      default:

        break;

    }

  if (selectedLevel !== 'all') {

      filtered = filtered.filter(course => course.level === selectedLevel);

    }

    if (showFreeOnly) {

      filtered = filtered.filter(course => course.price === 0);

    }

    return filtered;

  };

  const renderPriceTag = (course) => {

    return course.price === 0 ? (

      <span className="price free">FREE</span>

    ) : (

      <span className="price paid">${course.price}</span>

    );

  };

  const renderLevelBadge = (level) => {

    const levelStyles = {

      'Beginner': 'level-beginner',

      'Intermediate': 'level-intermediate',

      'Advanced': 'level-advanced'

    };

    return (

      <span className={`level-badge ${levelStyles[level]}`}>

        {level}

      </span>

    );

  };

  const renderProgressBar = (course) => {

    if (!course.enrolled) return null;

    return (

      <div className="progress-section">

        <div className="progress-info">

          <span>Progress: {course.progress}%</span>

          {course.completed && <span className="completed-badge">✅ Completed</span>}

        </div>

        <div className="progress-bar">

          <div

            className="progress-fill"

            style={{ width: `${course.progress}%` }}

          ></div>

        </div>

      </div>

    );

  };

  const renderEnrollmentActions = (course) => {

    if (course.enrolled) {

      if (course.completed) {

        return (

          <div className="course-actions">

            <button className="btn-success" disabled>Course Completed</button>

            {course.hasCertificate && (

              <button className="btn-secondary">Download Certificate</button>

            )}

          </div>

        );

      } else {

        return (

          <div className="course-actions">

            <button className="btn-primary">Continue Learning</button>

            <button className="btn-secondary">View Progress</button>

          </div>

        );

      }

    } else {

      return (

        <div className="course-actions">

          <button className="btn-primary">

            {course.price === 0 ? 'Enroll for Free' : 'Enroll Now'}

          </button>

          <button className="btn-secondary">Preview</button>

        </div>

      );

    }

  };

  const renderCourseCard = (course) => {

    return (

      <div key={course.id} className="course-card">

        {/\* New course indicator \*/}

        {course.isNew && <div className="new-badge">🆕 New!</div>}

          <div className="course-header">

          <h3>{course.title}</h3>

          <div className="course-meta">

            {renderPriceTag(course)}

            {renderLevelBadge(course.level)}

          </div>

        </div>

        <div className="course-info">

          <p><strong>Instructor:</strong> {course.instructor}</p>

          <p><strong>Duration:</strong> {course.duration}</p>

          <p><strong>Students:</strong> {course.students.toLocaleString()}</p>

            {course.rating >= 4.5 ? (

            <p className="rating-high">⭐ {course.rating} (Highly Rated!)</p>

          ) : (

            <p className="rating-normal">⭐ {course.rating}</p>

          )}

        </div>

        {renderProgressBar(course)}

        {course.hasCertificate && (

          <div className="certificate-info">

            🏆 Certificate of completion included

          </div>

        )}

        {renderEnrollmentActions(course)}

      </div>

    );

  };

  const renderSummaryStats = () => {

    const enrolledCourses = courses.filter(course => course.enrolled);

    const completedCourses = courses.filter(course => course.completed);

    const totalProgress = enrolledCourses.length > 0

      ? Math.round(enrolledCourses.reduce((sum, course) => sum + course.progress, 0) / enrolledCourses.length)

      : 0;

    return (

      <div className="summary-stats">

        <div className="stat-card">

          <h4>Enrolled Courses</h4>

          <span className="stat-number">{enrolledCourses.length}</span>

        </div>

        <div className="stat-card">

          <h4>Completed</h4>

          <span className="stat-number">{completedCourses.length}</span>

        </div>

        <div className="stat-card">

          <h4>Average Progress</h4>

          <span className="stat-number">{totalProgress}%</span>

        </div>

      </div>

    );

  };

  const filteredCourses = getFilteredCourses();

  return (

    <div className="course-details">

      <h2>🎓 Course Details Component</h2>

      {courses.some(course => course.enrolled) && renderSummaryStats()}

        <div className="controls">

        <div className="filter-group">

          <label>View:</label>

          <select value={viewMode} onChange={(e) => setViewMode(e.target.value)}>

            <option value="all">All Courses</option>

            <option value="enrolled">My Courses</option>

            <option value="available">Available Courses</option>

          </select>

        </div>

            <div className="filter-group">

          <label>Level:</label>

          <select value={selectedLevel} onChange={(e) => setSelectedLevel(e.target.value)}>

            {levels.map(level => (

              <option key={level} value={level}>

                {level.charAt(0).toUpperCase() + level.slice(1)}

              </option>

            ))}

          </select>

        </div>

          <div className="filter-group">

          <label>

            <input

              type="checkbox"

              checked={showFreeOnly}

              onChange={(e) => setShowFreeOnly(e.target.checked)}

            />

            Free courses only

          </label>

        </div>

      </div>

      {filteredCourses.length === 0 ? (

        <div className="empty-state">

          <h3>No courses found</h3>

          <p>

            {viewMode === 'enrolled'

              ? "You haven't enrolled in any courses yet."

              : "Try adjusting your filters to see more courses."

            }

          </p>

          {viewMode === 'enrolled' && (

            <button onClick={() => setViewMode('available')} className="btn-primary">

              Browse Available Courses

            </button>

          )}

        </div>

      ) : (

        <>

          <div className="results-info">

            <p>

              Showing {filteredCourses.length} of {courses.length} courses

              {showFreeOnly && " (Free courses only)"}

            </p>

          </div>

          <div className="courses-grid">

            {filteredCourses.map(course => renderCourseCard(course))}

          </div>

        </>

      )}

    </div>

  );

};

export default CourseDetails;











