code

```jsx
import React, { useState, useEffect, useContext, createContext, useRef } from 'react';


// Mock Data
const mockCourses = [
  {
    id: 1,
    title: "Introduction to Web Development",
    instructor: "Alice Johnson",
    duration: "8 weeks",
    price: 99.99,
    image: "https://placehold.co/400x200/3b82f6/ffffff?text=Web+Dev",
    description: "Learn HTML, CSS, and JavaScript from scratch.",
    media: [
      { type: 'video', src: 'https://example.com/video1.mp4', duration: '15:30', title: 'HTML Basics' },
      { type: 'audio', src: 'https://example.com/audio1.mp3', duration: '12:15', title: 'CSS Fundamentals' },
      { type: 'video', src: 'https://example.com/video2.mp4', duration: '20:45', title: 'JavaScript Intro' }
    ],
    progress: 60
  },
  {
    id: 2,
    title: "Data Science with Python",
    instructor: "Bob Smith",
    duration: "10 weeks",
    price: 149.99,
    image: "https://placehold.co/400x200/10b981/ffffff?text=Data+Science",
    description: "Master data analysis and machine learning.",
    media: [
      { type: 'video', src: 'https://example.com/video3.mp4', duration: '18:20', title: 'Pandas
Overview' },
      { type: 'audio', src: 'https://example.com/audio2.mp3', duration: '10:05', title: 'NumPy
Basics' }
    ],
    progress: 30
  },
  {
    id: 3,
    title: "UI/UX Design Principles",
    instructor: "Carol Lee",
    duration: "6 weeks",
    price: 89.99,
    image: "https://placehold.co/400x200/8b5cf6/ffffff?text=Design",
    description: "Create beautiful and intuitive user interfaces.",
    media: [
      { type: 'video', src: 'https://example.com/video4.mp4', duration: '22:10', title: 'Design
Thinking' }
    ],
    progress: 0
  }
];

const mockUser = {
  id: 1,
  name: "John Doe",
  email: "john@example.com",
  enrolledCourses: [1, 2],
  completedCourses: 1,
  totalHours: 45
};

// Contexts
const AuthContext = createContext();
const ThemeContext = createContext();
const CartContext = createContext();

// Auth Provider
const AuthProvider = ({ children }) => {
  const [user, setUser] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
   // Simulate checking for stored token
   const token = localStorage.getItem('authToken');
   if (token) {
      setUser(mockUser);
   }
   setLoading(false);
}, []);

const login = (email, password) => {
   // Simulate API call
   return new Promise((resolve) => {
      setTimeout(() => {
        if (email && password) {
          const token = 'mock-jwt-token-' + Date.now();
          localStorage.setItem('authToken', token);
          setUser(mockUser);
          resolve({ success: true });
        } else {
          resolve({ success: false, error: 'Invalid credentials' });
        }
      }, 500);
   });
};

const logout = () => {
   localStorage.removeItem('authToken');
   setUser(null);
};
```

```
  return (
    <AuthContext.Provider value={{ user, loading, login, logout }}>
      {children}
    </AuthContext.Provider>
  );
};

// Theme Provider
const ThemeProvider = ({ children }) => {
  const [darkMode, setDarkMode] = useState(() => {
    const saved = localStorage.getItem('darkMode');
    return saved ? JSON.parse(saved) : false;
  });

  useEffect(() => {
    localStorage.setItem('darkMode', JSON.stringify(darkMode));
    if (darkMode) {
      document.documentElement.classList.add('dark');
    } else {
      document.documentElement.classList.remove('dark');
    }
  }, [darkMode]);

  return (
    <ThemeContext.Provider value={{ darkMode, setDarkMode }}>
      {children}
    </ThemeContext.Provider>
  );
};

// Cart Provider
const CartProvider = ({ children }) => {
  const [cart, setCart] = useState(() => {
    const saved = localStorage.getItem('cart');
    return saved ? JSON.parse(saved) : [];
  });

  useEffect(() => {
    localStorage.setItem('cart', JSON.stringify(cart));
  }, [cart]);

  const addToCart = (course) => {
    setCart(prev => {
      if (!prev.find(item => item.id === course.id)) {
        return [...prev, course];
      }
      return prev;
    });
  };

  const removeFromCart = (courseId) => {
    setCart(prev => prev.filter(item => item.id !== courseId));
```

```
 };


 const clearCart = () => {
   setCart([]);
 };


 const isInCart = (courseId) => {
   return cart.some(item => item.id === courseId);
 };


 const total = cart.reduce((sum, course) => sum + course.price, 0);


 return (
   <CartContext.Provider value={{ cart, addToCart, removeFromCart, clearCart, isInCart, total
}}>
     {children}
   </CartContext.Provider>
 );
};


// Custom Hook for Canvas Visualization
const useCanvasVisualization = (ref, data) => {
 useEffect(() => {
   if (!ref.current || !data) return;


   const canvas = ref.current;
   const ctx = canvas.getContext('2d');

   const { width, height } = canvas;



   // Clear canvas

   ctx.clearRect(0, 0, width, height);



   // Draw gradient background

   const gradient = ctx.createLinearGradient(0, 0, width, height);

   gradient.addColorStop(0, 'rgba(59, 130, 246, 0.3)');

   gradient.addColorStop(1, 'rgba(16, 185, 129, 0.3)');

   ctx.fillStyle = gradient;

   ctx.fillRect(0, 0, width, height);



   // Draw data visualization

   const max = Math.max(...data);

   const barWidth = width / data.length - 10;



   data.forEach((value, index) => {

     const x = index * (barWidth + 10) + 5;

     const barHeight = (value / max) * (height - 40);

     const y = height - barHeight - 20;



     // Draw bar

     ctx.fillStyle = `hsl(${index * 45}, 70%, 50%)`;

     ctx.fillRect(x, y, barWidth, barHeight);
```

```
      // Add glow effect
      ctx.shadowColor = ctx.fillStyle;
      ctx.shadowBlur = 10;
      ctx.fillRect(x, y, barWidth, barHeight);
      ctx.shadowBlur = 0;

      // Draw label
      ctx.fillStyle = 'rgba(255, 255, 255, 0.8)';
      ctx.font = '12px system-ui';
      ctx.textAlign = 'center';
      ctx.fillText(value, x + barWidth / 2, height - 5);
    });

    // Draw axes
    ctx.strokeStyle = 'rgba(255, 255, 255, 0.3)';
    ctx.lineWidth = 1;
    ctx.beginPath();
    ctx.moveTo(0, height - 20);
    ctx.lineTo(width, height - 20);
    ctx.stroke();

  }, [ref, data]);
};

// Components
const Header = () => {
  const { user, logout } = useContext(AuthContext);

  const { darkMode, setDarkMode } = useContext(ThemeContext);

  const { cart } = useContext(CartContext);



  return (

    <header className="bg-white dark:bg-gray-800 shadow-md sticky top-0 z-50">

      <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">

        <div className="flex justify-between items-center h-16">

          <div className="flex items-center">

            <h1 className="text-2xl font-bold text-blue-600 dark:text-blue-400">EduStream</h1>

          </div>



          <div className="flex items-center space-x-6">

            <button

              onClick={() => setDarkMode(!darkMode)}

              className="p-2 rounded-full hover:bg-gray-200 dark:hover:bg-gray-700 transition-
colors"

              aria-label="Toggle dark mode"

            >

              {darkMode ? (

                <svg className="w-5 h-5 text-yellow-300" fill="currentColor" viewBox="0 0 20 20">

                  <path fillRule="evenodd" d="M10 2a1 1 0 011 1v1a1 1 0 11-2 0V3a1 1 0 011-1zm4 8a4
4 0 11-8 0 4 4 0 018 0zm-.464 4.95l.707.707a1 1 0 001.414-1.414l-.707-.707a1 1 0 00-1.414
1.414zm2.12-10.607a1 1 0 010 1.414l-.706.707a1 1 0 11-1.414-1.414l.707-.707a1 1 0 011.414
0zM17 11a1 1 0 100-2h-1a1 1 0 100 2h1zm-7 4a1 1 0 011 1v1a1 1 0 11-2 0v-1a1 1 0 011-1zM5.05
6.464A1 1 0 106.465 5.05l-.708-.707a1 1 0 00-1.414 1.414l.707.707zm1.414 8.486l-.707.707a1 1
0 01-1.414-1.414l.707-.707a1 1 0 011.414 1.414zM4 11a1 1 0 100-2H3a1 1 0 000 2h1z"
clipRule="evenodd" />

                </svg>
              ) : (
                <svg className="w-5 h-5 text-gray-700" fill="currentColor" viewBox="0 0 20 20">
                  <path d="M17.293 13.293A8 8 0 016.707 2.707a8.001 8.001 0 1010.586 10.586z" />
                </svg>
              )}
            </button>

            {user ? (
              <div className="flex items-center space-x-4">
                <div className="relative">
                  <button className="flex items-center space-x-1 text-gray-700 dark:text-gray-200
hover:text-blue-600 dark:hover:text-blue-400">
                <svg className="w-5 h-5" fill="none" stroke="currentColor" viewBox="0 0 24 24">
                  <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M3
3h18v18H3V3z" />
                  <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M9
7h6m-6 4h6m-6 4h6" />
                </svg>
                <span className="font-medium">{cart.length}</span>
                  </button>
                  {cart.length > 0 && (
                <div className="absolute -top-2 -right-2 bg-red-500 text-white text-xs rounded-
full w-5 h-5 flex items-center justify-center">
                  {cart.length}
                </div>
                  )}
                </div>
                <div className="flex items-center space-x-2">

                  <span className="text-gray-700 dark:text-gray-200">{user.name}</span>

                  <button

                onClick={logout}

                className="text-sm bg-red-500 hover:bg-red-600 text-white px-3 py-1 rounded-
md transition-colors"

                  >

                Logout

                  </button>

                </div>

              </div>

            ) : (

              <button className="bg-blue-600 hover:bg-blue-700 text-white px-4 py-2 rounded-
md transition-colors">

                Login

              </button>

            )}

          </div>

        </div>

      </div>

     </header>

     );

};


const LoginForm = () => {

     const { login } = useContext(AuthContext);
const [formData, setFormData] = useState({ email: '', password: '' });

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

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


const validate = () => {

      const newErrors = {};


      if (!formData.email) {

        newErrors.email = 'Email is required';

      } else if (!/\S+@\S+\.\S+/.test(formData.email)) {

        newErrors.email = 'Email is invalid';

      }


      if (!formData.password) {

        newErrors.password = 'Password is required';

      } else if (formData.password.length < 6) {

        newErrors.password = 'Password must be at least 6 characters';

      }


      setErrors(newErrors);

      return Object.keys(newErrors).length === 0;

};


const handleChange = (e) => {

      const { name, value } = e.target;

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

```
   // Real-time validation
   if (errors[name]) {
     validate();
   }
 };

 const handleSubmit = async (e) => {
   e.preventDefault();
   if (validate()) {
     setLoading(true);
     const result = await login(formData.email, formData.password);
     if (!result.success) {
       setErrors({ form: result.error });
     }
     setLoading(false);
   }
 };

 return (
   <div className="max-w-md mx-auto bg-white dark:bg-gray-800 p-8 rounded-lg shadow-
lg">
     <h2 className="text-2xl font-bold mb-6 text-center text-gray-800 dark:text-white">Login
to EduStream</h2>

     {errors.form && (
       <div className="mb-4 p-3 bg-red-100 dark:bg-red-900 text-red-700 dark:text-red-200
rounded-md">
          {errors.form}
       </div>
     )}

     <form onSubmit={handleSubmit} noValidate>
       <div className="mb-4">
          <label className="block text-gray-700 dark:text-gray-300 mb-2">Email</label>
          <input
            type="email"
            name="email"
            value={formData.email}
            onChange={handleChange}
            className={`w-full px-3 py-2 border rounded-md focus:outline-none focus:ring-2
focus:ring-blue-500 dark:bg-gray-700 dark:border-gray-600 ${
              errors.email ? 'border-red-500' : 'border-gray-300'
            }`}
            placeholder="Enter your email"
          />
          {errors.email && <p className="mt-1 text-red-500 text-sm">{errors.email}</p>}
       </div>

       <div className="mb-6">
          <label className="block text-gray-700 dark:text-gray-300 mb-2">Password</label>
          <input
            type="password"
            name="password"

            value={formData.password}

            onChange={handleChange}

            className={`w-full px-3 py-2 border rounded-md focus:outline-none focus:ring-2
focus:ring-blue-500 dark:bg-gray-700 dark:border-gray-600 ${

              errors.password ? 'border-red-500' : 'border-gray-300'

            }`}

            placeholder="Enter your password"

          />

          {errors.password && <p className="mt-1 text-red-500 text-sm">{errors.password}</p>}

       </div>



       <button

          type="submit"

          disabled={loading}

          className="w-full bg-blue-600 hover:bg-blue-700 disabled:bg-blue-400 text-white py-
2 px-4 rounded-md transition-colors flex items-center justify-center"

       >

          {loading ? (

            <>

              <svg className="animate-spin -ml-1 mr-3 h-5 w-5 text-white" fill="none" viewBox="0 0
24 24">

              <circle className="opacity-25" cx="12" cy="12" r="10" stroke="currentColor"
strokeWidth="4"></circle>

              <path className="opacity-75" fill="currentColor" d="M4 12a8 8 0 018-8V0C5.373 0 0
5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"></path>

              </svg>

              Logging in...
            </>
          ) : (
            'Login'
          )}
       </button>
     </form>
   </div>
  );
};

const MediaGallery = ({ course }) => {
  const [currentMedia, setCurrentMedia] = useState(course.media[0]);
  const [isPlaying, setIsPlaying] = useState(false);
  const [currentTime, setCurrentTime] = useState(0);
  const [duration, setDuration] = useState(0);
  const videoRef = useRef(null);
  const audioRef = useRef(null);
  const canvasRef = useRef(null);

  const handlePlayPause = () => {
    if (currentMedia.type === 'video' && videoRef.current) {
      if (isPlaying) {
        videoRef.current.pause();
      } else {
        videoRef.current.play();
      }
     setIsPlaying(!isPlaying);
   } else if (currentMedia.type === 'audio' && audioRef.current) {
     if (isPlaying) {
       audioRef.current.pause();
     } else {
       audioRef.current.play();
     }
     setIsPlaying(!isPlaying);
   }
};

const handleTimeUpdate = (time) => {
   setCurrentTime(time);
   // Update canvas visualization with time data
   if (canvasRef.current) {
     const ctx = canvasRef.current.getContext('2d');
     const width = canvasRef.current.width;
     const height = canvasRef.current.height;

     // Draw progress indicator
     ctx.fillStyle = 'rgba(59, 130, 246, 0.5)';
     ctx.fillRect(0, height - 10, (time / duration) * width, 10);
   }
};

const handleLoadedMetadata = (mediaDuration) => {
   setDuration(mediaDuration);
  };

  const handleMediaChange = (media) => {
   setCurrentMedia(media);
   setCurrentTime(0);
   setIsPlaying(false);
  };

  // Simulate analytics data for canvas
  const analyticsData = [25, 45, 30, 60, 40, 75, 55];

  useCanvasVisualization(canvasRef, analyticsData);

  return (
   <div className="bg-white dark:bg-gray-800 rounded-lg shadow-lg overflow-hidden">
    <div className="p-6">
     <h3 className="text-xl font-bold mb-4 text-gray-800 dark:text-white">{course.title}</h3>

     {/* Media List */}
     <div className="mb-6">
      <h4 className="font-semibold mb-3 text-gray-700 dark:text-gray-300">Course
Materials</h4>
      <div className="space-y-2">
       {course.media.map((media, index) => (
        <button
```

```
            key={index}

            onClick={() => handleMediaChange(media)}

            className={`w-full text-left p-3 rounded-lg transition-colors flex items-center space-
x-3 ${

              currentMedia === media

                ? 'bg-blue-100 dark:bg-blue-900 text-blue-800 dark:text-blue-200'

                : 'hover:bg-gray-100 dark:hover:bg-gray-700'

            }`}

          >

            {media.type === 'video' ? (

              <svg className="w-5 h-5 text-red-500" fill="currentColor" viewBox="0 0 20 20">

                <path fillRule="evenodd" d="M4 3a2 2 0 00-2 2v10a2 2 0 002 2h12a2 2 0 002-2V5a2
2 0 00-2-2H4zm12 12H4l4-8 3 6 2-4 3 6z" clipRule="evenodd" />

              </svg>

            ) : (

              <svg className="w-5 h-5 text-green-500" fill="currentColor" viewBox="0 0 20 20">

                <path fillRule="evenodd" d="M9.383 3.076A1 1 0 0110 4v12a1 1 0 01-
1.617.793L4.828 13H2a1 1 0 01-1-1V8a1 1 0 011-1h2.828l3.555-3.793A1 1 0 019.383 3.076zM15
8a1 1 0 011-1h1a1 1 0 011 1v4a1 1 0 01-1 1h-1a1 1 0 01-1-1V8z" clipRule="evenodd" />

              </svg>

            )}

            <span className="flex-1">{media.title}</span>

            <span className="text-sm text-gray-500 dark:text-gray-400">{media.duration}
</span>

          </button>

        ))}

      </div>

    </div>
```

```
{/* Media Player */}
<div className="bg-gray-900 rounded-lg overflow-hidden">
 {currentMedia.type === 'video' ? (
  <video
   ref={videoRef}
   src={currentMedia.src}
   className="w-full"
   onPlay={() => setIsPlaying(true)}
   onPause={() => setIsPlaying(false)}
   onEnded={() => setIsPlaying(false)}
   onTimeUpdate={(e) => handleTimeUpdate(e.target.currentTime)}
   onLoadedMetadata={(e) => handleLoadedMetadata(e.target.duration)}
  />
 ) : (
  <audio
   ref={audioRef}
   src={currentMedia.src}
   onPlay={() => setIsPlaying(true)}
   onPause={() => setIsPlaying(false)}
   onEnded={() => setIsPlaying(false)}
   onTimeUpdate={(e) => handleTimeUpdate(e.target.currentTime)}
   onLoadedMetadata={(e) => handleLoadedMetadata(e.target.duration)}
  />
 )}
```

```
        {/* Custom Controls */}
        <div className="p-4 bg-gray-800">
          <div className="flex items-center space-x-4">
            <button
              onClick={handlePlayPause}
              className="text-white hover:text-blue-400 transition-colors"
            >
              {isPlaying ? (
                <svg className="w-6 h-6" fill="currentColor" viewBox="0 0 20 20">
                  <path fillRule="evenodd" d="M18 10a8 8 0 11-16 0 8 8 0 0116 0zM7 8a1 1 0 012
0v4a1 1 0 11-2 0V8zm5-1a1 1 0 00-1 1v4a1 1 0 102 0V8a1 1 0 00-1-1z" clipRule="evenodd" />
                </svg>
              ) : (
                <svg className="w-6 h-6" fill="currentColor" viewBox="0 0 20 20">
                  <path fillRule="evenodd" d="M10 18a8 8 0 100-16 8 8 0 000 16zM9.555 7.168A1 1 0
008 8v4a1 1 0 001.555.832l3-2a1 1 0 000-1.664l-3-2z" clipRule="evenodd" />
                </svg>
              )}
            </button>

            <div className="flex-1">
              <div className="w-full bg-gray-600 rounded-full h-1">
                <div
                  className="bg-blue-500 h-1 rounded-full transition-all"
                  style={{ width: `${duration ? (currentTime / duration) * 100 : 0}%` }}
                ></div>
              </div>
              <div className="text-xs text-gray-400 mt-1">
                {Math.floor(currentTime / 60)}:{Math.floor(currentTime % 60).toString().padStart(2,
'0')} /
                {Math.floor(duration / 60)}:{Math.floor(duration % 60).toString().padStart(2, '0')}
              </div>
            </div>
          </div>
        </div>
      </div>


      {/* Canvas Visualization */}
      <div className="mt-6">
        <h4 className="font-semibold mb-3 text-gray-700 dark:text-gray-300">Learning
Analytics</h4>
        <canvas
          ref={canvasRef}
          width={400}
          height={150}
          className="w-full bg-gray-100 dark:bg-gray-700 rounded-lg"
        />
      </div>
     </div>
    </div>
   );
};


const CourseCard = ({ course }) => {
  const { addToCart, isInCart } = useContext(CartContext);

  const [showMedia, setShowMedia] = useState(false);


  return (

    <div className="bg-white dark:bg-gray-800 rounded-lg shadow-md overflow-hidden
transition-all hover:shadow-lg">

      <img src={course.image} alt={course.title} className="w-full h-48 object-cover" />


      <div className="p-6">

        <h3 className="text-xl font-bold mb-2 text-gray-800 dark:text-white">{course.title}</h3>

        <p className="text-gray-600 dark:text-gray-400 mb-2">by {course.instructor}</p>

        <p className="text-gray-600 dark:text-gray-400 mb-4">{course.description}</p>


        <div className="flex items-center justify-between mb-4">

          <span className="text-blue-600 dark:text-blue-400 font-bold text-lg">${course.price}
</span>

          <span className="text-sm text-gray-500 dark:text-gray-400">{course.duration}</span>

        </div>


        {course.progress > 0 && (

          <div className="mb-4">

            <div className="flex justify-between text-sm text-gray-600 dark:text-gray-400 mb-1">

              <span>Progress</span>

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

            </div>

            <div className="w-full bg-gray-200 dark:bg-gray-700 rounded-full h-2">

              <div
            className="bg-green-500 h-2 rounded-full transition-all"
            style={{ width: `${course.progress}%` }}
          ></div>
        </div>
      </div>
    )}

    <div className="flex space-x-2">
      <button
        onClick={() => setShowMedia(!showMedia)}
        className="flex-1 bg-gray-200 dark:bg-gray-700 hover:bg-gray-300 dark:hover:bg-
gray-600 text-gray-800 dark:text-white py-2 px-4 rounded-md transition-colors text-sm"
      >
        {showMedia ? 'Hide Materials' : 'View Materials'}
      </button>

      {!isInCart(course.id) && (
        <button
          onClick={() => addToCart(course)}
          className="bg-blue-600 hover:bg-blue-700 text-white py-2 px-4 rounded-md
transition-colors text-sm"
        >
          Add to Cart
        </button>
      )}
    </div>
```

```
        {showMedia && (
          <div className="mt-4 pt-4 border-t border-gray-200 dark:border-gray-700">
            <h4 className="font-semibold mb-2 text-gray-700 dark:text-gray-300">Media
Files</h4>
            <div className="space-y-1">
              {course.media.map((media, index) => (
                <div key={index} className="text-sm text-gray-600 dark:text-gray-400 flex items-
center space-x-2">
                  <span>{media.type === 'video' ? '🎥' : '🎵'}</span>
                  <span>{media.title}</span>
                  <span className="text-xs">({media.duration})</span>
                </div>
              ))}
            </div>
          </div>
        )}
      </div>
    </div>
  );
};


const Dashboard = () => {
  const { user } = useContext(AuthContext);
  const { cart } = useContext(CartContext);


  const stats = [
    { label: 'Completed Courses', value: user?.completedCourses || 0, icon: '📚' },
  { label: 'Total Hours', value: user?.totalHours || 0, icon: '⏰' },

  { label: 'Courses in Progress', value: user?.enrolledCourses?.length || 0, icon: '🎯' },

  { label: 'Achievements', value: 5, icon: '🏆' }

 ];


 return (

  <div className="bg-white dark:bg-gray-800 rounded-lg shadow-lg p-6">

   <h2 className="text-2xl font-bold mb-6 text-gray-800 dark:text-white">Dashboard</h2>


   {/* Stats Grid */}

   <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6 mb-8">

    {stats.map((stat, index) => (

     <div key={index} className="bg-gradient-to-br from-blue-50 to-indigo-50 dark:from-
gray-700 dark:to-gray-800 p-6 rounded-lg">

      <div className="flex items-center justify-between">

       <div>

        <p className="text-sm font-medium text-gray-600 dark:text-gray-400">{stat.label}
</p>

        <p className="text-3xl font-bold text-gray-800 dark:text-white">{stat.value}</p>

       </div>

       <span className="text-4xl">{stat.icon}</span>

      </div>

     </div>

    ))}

   </div>


   {/* Recent Activity */}
   <div className="mb-8">
    <h3 className="text-xl font-semibold mb-4 text-gray-800 dark:text-white">Recent Activity</h3>
    <div className="space-y-3">
     {mockCourses.slice(0, 3).map((course, index) => (
      <div key={index} className="flex items-center p-4 bg-gray-50 dark:bg-gray-700 rounded-lg">
       <img src={course.image} alt={course.title} className="w-16 h-16 object-cover rounded-lg mr-4" />
       <div className="flex-1">
        <h4 className="font-medium text-gray-800 dark:text-white">{course.title}</h4>
        <p className="text-sm text-gray-600 dark:text-gray-400">Progress: {course.progress}%</p>
       </div>
       <div className="text-right">
        <p className="text-sm text-gray-600 dark:text-gray-400">2 hours ago</p>
       </div>
      </div>
     ))}
    </div>
   </div>


   {/* Cart Summary */}
   {cart.length > 0 && (
    <div className="bg-blue-50 dark:bg-blue-900/30 p-6 rounded-lg">
     <h3 className="text-xl font-semibold mb-4 text-gray-800 dark:text-white">Shopping Cart</h3>
     <div className="space-y-3 mb-4">
```

```
          {cart.map(course => (
            <div key={course.id} className="flex justify-between items-center">
              <span className="text-gray-700 dark:text-gray-300">{course.title}</span>
              <span className="font-medium text-gray-800 dark:text-white">${course.price}
</span>
            </div>
          ))}
        </div>
        <div className="flex justify-between items-center pt-4 border-t border-blue-200
dark:border-blue-800">
          <span className="font-bold text-gray-800 dark:text-white">Total:</span>
          <span className="font-bold text-lg text-blue-600 dark:text-blue-
400">${cart.reduce((sum, course) => sum + course.price, 0).toFixed(2)}</span>
        </div>
      </div>
     )}
    </div>
  );
};


const CourseCatalog = () => {
  return (
    <div>
      <div className="flex justify-between items-center mb-8">
        <h2 className="text-3xl font-bold text-gray-800 dark:text-white">Courses</h2>
        <div className="flex items-center space-x-4">
```

```
          <select className="bg-white dark:bg-gray-800 border border-gray-300 dark:border-
gray-600 rounded-md px-3 py-2">
            <option>All Categories</option>
            <option>Web Development</option>
            <option>Data Science</option>
            <option>Design</option>
          </select>
        </div>
      </div>

      <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-8">
        {mockCourses.map(course => (
          <CourseCard key={course.id} course={course} />
        ))}
      </div>
    </div>
  );
};

const App = () => {
  const [activeTab, setActiveTab] = useState('dashboard');

  return (
    <AuthProvider>
      <ThemeProvider>
        <CartProvider>
```

```
     <div className="min-h-screen bg-gray-100 dark:bg-gray-900 transition-colors
duration-200">
      <Header />

      <main className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-8">
       <div className="mb-8">
        <nav className="flex space-x-8 border-b border-gray-200 dark:border-gray-700">
         {['dashboard', 'courses', 'media'].map((tab) => (
          <button
           key={tab}
           onClick={() => setActiveTab(tab)}
           className={`py-4 px-1 border-b-2 font-medium text-sm capitalize transition-
colors ${
            activeTab === tab
             ? 'border-blue-500 text-blue-600 dark:text-blue-400'
             : 'border-transparent text-gray-500 hover:text-gray-700 hover:border-gray-300
dark:text-gray-400 dark:hover:text-gray-300'
           }`}
          >
           {tab}
          </button>
         ))}
        </nav>
       </div>

       {activeTab === 'dashboard' && <Dashboard />}
       {activeTab === 'courses' && <CourseCatalog />}
```

```
              {activeTab === 'media' && <MediaGallery course={mockCourses[0]} />}
            </main>

            <footer className="bg-white dark:bg-gray-800 shadow-md mt-12">
              <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-8">
                <div className="text-center text-gray-600 dark:text-gray-400">
                  <h3 className="text-xl font-bold text-blue-600 dark:text-blue-400 mb-2">EduStream</h3>
                  <p>© 2024 EduStream. All rights reserved.</p>
                </div>
              </div>
            </footer>
          </div>
        </CartProvider>
      </ThemeProvider>
    </AuthProvider>
  );
};

export default App;
```