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

import { Link, useParams, useLocation } from "react-router-dom";

import {

  FileImage,

  FileVideo,

  MapPin,

  Mail,

  Tag,

  CheckCircle,

  DollarSign,

  ChevronLeft,

  AlertCircle,

  Clock,

  Star,

  Share2,

  Heart,

  X,

  Phone,

  Calendar,

  MessageCircle,

  ArrowLeft,

  ArrowRight,

  MapPinCheckInsideIcon,

} from "lucide-react";

*// Custom hook for service details fetching with query parameters*

const useServiceDetails = (id) => {

  const [service, setService] = useState(null);

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

  const [error, setError] = useState(null);

  const webUrl = "https://backend.onetouchmoments.com/";

  const location = useLocation();

  useEffect(() => {

    const fetchServiceDetails = *async* () => {

      try {

*// Get query parameters from URL*

        const queryParams = new URLSearchParams(location.search);

        const firmName = queryParams.get("firm\_name");

        const vendorId = queryParams.get("vendor\_id") || id;

        if (!firmName) {

          throw new Error("Firm name not provided in URL");

        }

*// Construct API URL similar to HTML implementation*

        const ApiUrl = `${webUrl}user\_controller/Service/index\_get?data=${encodeURIComponent(

          firmName

        )}&vendor\_id=${encodeURIComponent(vendorId)}`;

        const response = await fetch(ApiUrl);

        if (!response.ok) {

          throw new Error("Failed to fetch service details");

        }

        const data = await response.json();

        console.log("Service Details API Response:", data);

        if (!data.data || data.data.length === 0) {

          throw new Error("No service found");

        }

        setService(data.data[0]); *// Using first record as shown in HTML implementation*

        setLoading(false);

      } catch (error) {

        console.error("Error fetching service details:", error);

        setError(error.message);

        setLoading(false);

      }

    };

    if (id) {

      fetchServiceDetails();

    }

  }, [id, webUrl, location.search]);

  return { service, loading, error, webUrl };

};

*// Service Details Component*

function ServiceDetails() {

  const { id } = useParams();

  const { service, loading, error, webUrl } = useServiceDetails(id);

  const [selectedMedia, setSelectedMedia] = useState(null);

  const [mainImage, setMainImage] = useState("");

  const [liked, setLiked] = useState(false);

  const [showImageCarousel, setShowImageCarousel] = useState(false);

  const [showVideoCarousel, setShowVideoCarousel] = useState(false);

  const [currentMediaIndex, setCurrentMediaIndex] = useState(0);

  const [showToast, setShowToast] = useState(false);

  const [toastMessage, setToastMessage] = useState("");

*// Function to handle main image and thumbnail gallery setup*

  useEffect(() => {

    if (service?.images) {

      const imageArray = service.images.split(",").filter((img) => img.trim());

      if (imageArray.length > 0) {

        setMainImage(`${webUrl}${imageArray[0].trim()}`);

      }

    }

  }, [service, webUrl]);

*// Share handler function*

  const ShareHandle = () => {

*// Copy the current page URL to the clipboard*

    navigator.clipboard

      .writeText(window.location.href)

      .then(() => {

        showToastNotification("Link copied to clipboard!");

      })

      .catch(() => {

        showToastNotification("Failed to copy link.");

      });

*// Share the current page URL to WhatsApp*

    const whatsappUrl = `https://api.whatsapp.com/send?text=${encodeURIComponent(

      window.location.href

    )}`;

    window.open(whatsappUrl, "\_blank");

  };

*// Show toast notification*

  const showToastNotification = (message) => {

    setToastMessage(message);

    setShowToast(true);

    setTimeout(() => setShowToast(false), 3000);

  };

*// Function to render thumbnail images*

  const getImages = () => {

    if (!service?.images) return [];

    return service.images.split(",").filter((img) => img.trim());

  };

  const getVideos = () => {

    if (!service?.videos) return [];

    return service.videos.split(",").filter((vid) => vid.trim());

  };

*// Function to display image as main preview*

  const setImageAsMain = (imagePath) => {

    setMainImage(`${webUrl}${imagePath}`);

  };

*// Handle like/favorite*

  const handleLikeToggle = () => {

    setLiked(!liked);

    showToastNotification(

      liked ? "Removed from favorites" : "Added to favorites"

    );

  };

  const MediaCarousel = ({

    items,

    type,

    onClose,

    currentIndex,

    setCurrentIndex,

  }) => {

    const goToNext = () => {

      setCurrentIndex((prev) => (prev === items.length - 1 ? 0 : prev + 1));

    };

    const goToPrevious = () => {

      setCurrentIndex((prev) => (prev === 0 ? items.length - 1 : prev - 1));

    };

    return (

      <div *className*="fixed inset-0 bg-black bg-opacity-95 z-50 flex flex-col items-center justify-center">

        <button

*onClick*={onClose}

*className*="absolute top-4 right-4 text-white hover:text-red-500 bg-black bg-opacity-50 p-2 rounded-full transition-all"

        >

          <X *size*={24} />

        </button>

        <div *className*="relative w-full max-w-6xl mx-auto px-4">

          <button

*onClick*={goToPrevious}

*className*="absolute left-4 top-1/2 -translate-y-1/2 text-white hover:text-red-500 bg-black bg-opacity-50 p-2 rounded-full transition-all"

          >

            <ArrowLeft *size*={24} />

          </button>

          <button

*onClick*={goToNext}

*className*="absolute right-4 top-1/2 -translate-y-1/2 text-white hover:text-red-500 bg-black bg-opacity-50 p-2 rounded-full transition-all"

          >

            <ArrowRight *size*={24} />

          </button>

          <div *className*="flex justify-center items-center h-[80vh]">

            {type === "image" ? (

              <img

*src*={`${webUrl}${items[currentIndex]}`}

*alt*={`Media ${currentIndex + 1}`}

*className*="max-h-full max-w-full object-contain"

              />

            ) : (

              <video

*controls*

*className*="max-h-full max-w-full"

*src*={`${webUrl}${items[currentIndex]}`}

              >

                Your browser does not support video playback.

              </video>

            )}

          </div>

          <div *className*="absolute left-4 right-4 bottom-4">

            <div *className*="flex justify-center gap-2 overflow-x-auto py-2">

              {items.map((item, index) => (

                <button

*key*={index}

*onClick*={() => setCurrentIndex(index)}

*className*={`w-16 h-16 flex-shrink-0 rounded-lg overflow-hidden border-2 transition-all

                    ${

                      currentIndex === index

                        ? "border-red-500 scale-110"

                        : "border-transparent opacity-70"

                    }`}

                >

                  {type === "image" ? (

                    <img

*src*={`${webUrl}${item}`}

*alt*={`Thumbnail ${index + 1}`}

*className*="w-full h-full object-cover"

                    />

                  ) : (

                    <div *className*="w-full h-full bg-gray-800 flex items-center justify-center">

                      <FileVideo *className*="text-white" *size*={24} />

                    </div>

                  )}

                </button>

              ))}

            </div>

          </div>

        </div>

      </div>

    );

  };

  const renderMediaPreviews = () => {

    const images = getImages();

    const videos = getVideos();

    return (

      <div *className*="space-y-4 mt-4">

        {*/\* Images preview \*/*}

        <div *className*="space-y-2">

          <div *className*="flex items-center gap-2">

            <FileImage *size*={18} *className*="text-gray-500" />

            <span *className*="font-medium">Images ({images.length})</span>

          </div>

          <div *className*="flex gap-2 overflow-x-auto pb-2">

            {images.slice(0, 4).map((image, index) => (

              <button

*key*={`img-${index}`}

*onClick*={() => {

                  setCurrentMediaIndex(index);

                  setShowImageCarousel(true);

                }}

*onMouseEnter*={() => setImageAsMain(image)}

*className*="relative flex-shrink-0 w-24 h-24 rounded-lg overflow-hidden transition-all hover:shadow-lg hover:scale-105 border-2 border-transparent hover:border-red-400"

              >

                <img

*src*={`${webUrl}${image}`}

*alt*={`Preview ${index + 1}`}

*className*="w-full h-full object-cover"

                />

                {index === 3 && images.length > 4 && (

                  <div *className*="absolute inset-0 bg-black bg-opacity-75 flex items-center justify-center text-white">

                    +{images.length - 4}

                  </div>

                )}

              </button>

            ))}

          </div>

        </div>

        {*/\* Videos preview \*/*}

        {videos.length > 0 && (

          <div *className*="space-y-2">

            <div *className*="flex items-center gap-2">

              <FileVideo *size*={18} *className*="text-gray-500" />

              <span *className*="font-medium">Videos ({videos.length})</span>

            </div>

            <div *className*="flex gap-2 overflow-x-auto pb-2">

              {videos.slice(0, 4).map((video, index) => (

                <button

*key*={`vid-${index}`}

*onClick*={() => {

                    setCurrentMediaIndex(index);

                    setShowVideoCarousel(true);

                  }}

*className*="relative flex-shrink-0 w-24 h-24 rounded-lg overflow-hidden bg-gray-100 transition-all hover:shadow-lg hover:scale-105 border-2 border-transparent hover:border-red-400"

                >

                  <div *className*="absolute inset-0 flex items-center justify-center">

                    <FileVideo *className*="text-red-500" *size*={24} />

                  </div>

                  <div *className*="absolute bottom-0 left-0 right-0 bg-black bg-opacity-60 py-1 text-white text-xs text-center">

                    Play

                  </div>

                  {index === 3 && videos.length > 4 && (

                    <div *className*="absolute inset-0 bg-black bg-opacity-75 flex items-center justify-center text-white">

                      +{videos.length - 4}

                    </div>

                  )}

                </button>

              ))}

            </div>

          </div>

        )}

      </div>

    );

  };

*// Skeleton loader*

  const SkeletonLoader = () => (

    <div *className*="max-w-6xl mx-auto p-6 animate-pulse">

      <div *className*="h-10 bg-gray-200 w-40 mb-6 rounded-lg"></div>

      <div *className*="flex flex-wrap">

        <div *className*="w-full md:w-1/3 pr-0 md:pr-6">

          <div *className*="h-64 bg-gray-200 rounded-lg mb-4"></div>

          <div *className*="flex justify-between space-x-2">

            {[...Array(3)].map((\_, i) => (

              <div *key*={i} *className*="h-20 w-20 bg-gray-200 rounded-lg"></div>

            ))}

          </div>

        </div>

        <div *className*="w-full md:w-2/3 mt-6 md:mt-0">

          <div *className*="h-10 bg-gray-200 w-3/4 mb-4 rounded-lg"></div>

          <div *className*="h-5 bg-gray-200 w-1/4 mb-6 rounded-lg"></div>

          <div *className*="space-y-2">

            <div *className*="h-4 bg-gray-200 rounded w-full"></div>

            <div *className*="h-4 bg-gray-200 rounded w-full"></div>

            <div *className*="h-4 bg-gray-200 rounded w-3/4"></div>

          </div>

          <div *className*="h-8 bg-gray-200 w-1/3 my-6 rounded-lg"></div>

          <div *className*="flex space-x-2">

            <div *className*="h-8 bg-gray-200 w-24 rounded-full"></div>

            <div *className*="h-8 bg-gray-200 w-24 rounded-full"></div>

          </div>

          <div *className*="mt-8 space-y-4">

            <div *className*="h-12 bg-gray-200 rounded-lg"></div>

            <div *className*="h-12 bg-gray-200 rounded-lg"></div>

          </div>

        </div>

      </div>

    </div>

  );

*// Error Component*

  const ErrorDisplay = ({ message }) => (

    <div *className*="flex flex-col items-center justify-center p-10 text-center min-h-screen bg-gray-50">

      <div *className*="bg-white p-8 rounded-2xl shadow-xl max-w-lg">

        <AlertCircle *className*="w-20 h-20 text-red-500 mb-6 mx-auto" />

        <h2 *className*="text-2xl font-bold text-gray-800 mb-4">

          Oops! Something went wrong

        </h2>

        <p *className*="text-gray-600 mb-8 bg-red-50 p-4 rounded-lg border-l-4 border-red-500">

          {message}

        </p>

        <Link

*to*="/services"

*className*="bg-red-600 text-white px-6 py-3 rounded-lg

          hover:bg-red-700 transition-colors flex items-center justify-center shadow-lg hover:shadow-xl"

        >

          <ChevronLeft *className*="mr-2" /> Back to Services

        </Link>

      </div>

    </div>

  );

*// Toast Notification Component*

  const ToastNotification = ({ message }) => (

    <div

*className*={`fixed bottom-4 right-4 bg-gray-800 text-white px-4 py-3 rounded-lg shadow-lg z-50 transform transition-all duration-300 ${

        showToast ? "translate-y-0 opacity-100" : "translate-y-12 opacity-0"

      }`}

    >

      {message}

    </div>

  );

*// Main render*

  if (loading) return <SkeletonLoader />;

  if (error) return <ErrorDisplay *message*={error} />;

  if (!service) return <ErrorDisplay *message*="Service not found" />;

*// Extract data from service object*

  const firmName = service.firm\_name;

  const description = service.description || "No description available";

  const price = service.off\_price || service.price || "0";

  const originalPrice = service.mrp\_price || "";

  const city = service.city || "";

  const state = service.state || "";

*// Calculate discount if applicable*

  const discountPercentage =

    originalPrice && price

      ? Math.round((1 - parseFloat(price) / parseFloat(originalPrice)) \* 100)

      : 0;

*// Use the first image from service or fallback to placeholder*

  const displayMainImage = mainImage;

*// Get specifications*

  const specifications = service.specification

    ? service.specification.split(",").map((spec) => spec.trim())

    : [

        "24/7 Support",

        "Free Consultation",

        "Satisfaction Guarantee",

        "Fast Response",

      ];

  return (

    <>

      <div *className*="max-w-6xl mx-auto bg-white shadow-2xl rounded-2xl border border-gray-200 my-10 overflow-hidden transform transition-all">

        {*/\* Header with back button and actions \*/*}

        <div *className*="p-4 md:p-6 bg-gradient-to-r from-gray-50 to-white border-b border-gray-200 flex justify-between items-center">

          <Link

*to*="/services"

*className*="inline-flex items-center text-red-600 hover:text-red-800

            transition-colors text-sm font-medium bg-white px-4 py-2 rounded-full shadow-sm hover:shadow group"

          >

            <ChevronLeft *className*="mr-1 group-hover:-translate-x-1 transition-transform" />

            Back to Services

          </Link>

          <div *className*="flex items-center space-x-3">

            <button

*className*="p-2 rounded-full hover:bg-gray-100 transition-colors"

*onClick*={ShareHandle}

*aria-label*="Share"

            >

              <Share2

*size*={20}

*className*="text-gray-600 hover:text-red-600 transition-colors"

              />

            </button>

            <button

*className*={`p-2 rounded-full hover:bg-red-50 transition-colors ${

                liked ? "text-red-500 bg-red-50" : "text-gray-600"

              }`}

*onClick*={handleLikeToggle}

*aria-label*={liked ? "Remove from favorites" : "Add to favorites"}

            >

              <Heart

*size*={20}

*fill*={liked ? "currentColor" : "none"}

*className*={liked ? "scale-110 transition-transform" : ""}

              />

            </button>

          </div>

        </div>

        <div *className*="flex flex-wrap">

          {*/\* Left column - Images \*/*}

          <div *className*="md:w-2/5 w-full p-4 md:p-6">

            <div *className*="relative rounded-xl overflow-hidden shadow-lg group">

              <img

*className*="w-full h-80 object-cover transition-transform duration-500 group-hover:scale-105"

*src*={displayMainImage}

*alt*="Service main"

*onClick*={() => setSelectedMedia(displayMainImage)}

              />

              {discountPercentage > 0 && (

                <div *className*="absolute top-4 right-4 bg-red-600 text-white px-3 py-1 rounded-full text-sm font-bold shadow-lg">

                  {discountPercentage}% OFF

                </div>

              )}

              <div *className*="absolute inset-0 bg-black opacity-0 group-hover:opacity-10 transition-opacity"></div>

              <div *className*="absolute bottom-4 right-4">

                <button

*className*="bg-white bg-opacity-90 hover:bg-opacity-100 p-2 rounded-full shadow-lg transform transition-transform hover:scale-110"

*onClick*={() => {

                    setCurrentMediaIndex(0);

                    setShowImageCarousel(true);

                  }}

                >

                  <FileImage *size*={20} *className*="text-red-600" />

                </button>

              </div>

            </div>

            {*/\* Thumbnails \*/*}

            {renderMediaPreviews()}

          </div>

          {*/\* Right column - Details \*/*}

          <div *className*="md:w-3/5 w-full p-6 md:p-8">

            {*/\* Service name and rating \*/*}

            <div *className*="flex flex-col md:flex-row justify-between items-start md:items-center">

              <h1 *className*="text-3xl font-bold text-gray-800 mb-2 md:mb-0 leading-tight">

                {firmName}

              </h1>

              <div *className*="flex items-center text-yellow-500 bg-yellow-50 px-3 py-1 rounded-lg">

                <Star *size*={18} *fill*="currentColor" />

                <Star *size*={18} *fill*="currentColor" />

                <Star *size*={18} *fill*="currentColor" />

                <Star *size*={18} *fill*="currentColor" />

                <Star *size*={18} *fill*="none" *stroke*="currentColor" />

                <span *className*="ml-2 text-gray-700 font-medium">(4.0)</span>

              </div>

            </div>

            {*/\* Service description \*/*}

            <div *className*="mt-6 text-gray-600 bg-gray-50 p-4 rounded-lg border-l-4 border-red-400">

              <p *className*="leading-relaxed">{description}</p>

            </div>

            {*/\* Price section \*/*}

            <div *className*="mt-6 flex items-baseline gap-3">

              <div *className*="text-3xl font-bold text-red-600">

                ₹ {parseFloat(price).toLocaleString()}

              </div>

              {originalPrice &&

                parseFloat(originalPrice) > parseFloat(price) && (

                  <div *className*="text-xl line-through text-gray-400">

                    ₹ {parseFloat(originalPrice).toLocaleString()}

                  </div>

                )}

              {discountPercentage > 0 && (

                <div *className*="bg-green-100 text-green-800 px-3 py-1 rounded-full text-sm font-medium shadow-sm">

                  Save {discountPercentage}%

                </div>

              )}

            </div>

            {*/\* Location and attributes \*/*}

            <div *className*="mt-6 flex flex-wrap items-center gap-3">

              {(city || state) && (

                <div *className*="flex items-center bg-blue-50 py-2 px-4 rounded-full shadow-sm">

                  <MapPin *className*="text-blue-600" *size*={16} />

                  <p *className*="text-gray-700 text-sm ml-1 font-medium">

                    {city}

                    {city && state && ", "}

                    {state}

                  </p>

                </div>

              )}

              <div *className*="flex items-center bg-purple-50 py-2 px-4 rounded-full shadow-sm">

                <Clock *className*="text-purple-600" *size*={16} />

                <p *className*="text-gray-700 text-sm ml-1 font-medium">

                  24/7 Service

                </p>

              </div>

              {service.is\_verified && (

                <div *className*="flex items-center bg-green-50 py-2 px-4 rounded-full shadow-sm">

                  <CheckCircle *className*="text-green-600" *size*={16} />

                  <p *className*="text-gray-700 text-sm ml-1 font-medium">

                    Verified Provider

                  </p>

                </div>

              )}

            </div>

            {*/\* Action buttons \*/*}

            <div *className*="mt-8">

              <div *className*="flex flex-col md:flex-row gap-4">

                <button

*className*="bg-gradient-to-r from-red-600 to-red-700 hover:from-red-700 hover:to-red-800

                  text-white font-medium py-3 px-6 rounded-xl flex-1 transition-all duration-300

                  transform hover:scale-105 shadow-lg hover:shadow-xl flex justify-center items-center"

                >

                  <Calendar *className*="mr-2" *size*={18} />

                  Book Now

                </button>

              </div>

            </div>

            {*/\* Service highlights \*/*}

            <div *className*="mt-8 bg-gray-50 p-5 rounded-xl border border-gray-200">

              <h3 *className*="font-semibold text-gray-800 mb-4 flex items-center">

                <Tag *className*="mr-2 text-red-600" *size*={18} />

                Contact Details

              </h3>

              <div *className*="grid grid-cols-1 sm:grid-cols-2 gap-3">

                <div *className*="flex flex-col bg-white py-3 px-4 rounded-lg shadow-sm">

                  <div *className*="flex items-center mb-2">

                    <MapPinCheckInsideIcon

*className*="text-green-500 mr-2 flex-shrink-0"

*size*={16}

                    />

                    <p *className*="text-gray-700 text-sm">

                      Pincode: {service.pincode}

                    </p>

                  </div>

                  <p *className*="text-gray-700 text-sm ml-6 mb-1">

                    State: {service.state}

                  </p>

                  <p *className*="text-gray-700 text-sm ml-6 mb-1">

                    Status: {service.status}

                  </p>

                  <p *className*="text-gray-700 text-sm ml-6 mb-1">

                    District: {service.district}

                  </p>

                  <p *className*="text-gray-700 text-sm ml-6 mb-1">

                    Firm Name: {service.firm\_name}

                  </p>

                  <p *className*="text-gray-700 text-sm ml-6">

                    Service Category: {service.service\_cat}

                  </p>

                </div>

              </div>

            </div>

            <div *className*="mt-8 bg-gray-50 p-5 rounded-xl border border-gray-200">

              <h3 *className*="font-semibold text-gray-800 mb-4 flex items-center">

                <Tag *className*="mr-2 text-red-600" *size*={18} />

                Service Include

              </h3>

              <div *className*="grid grid-cols-1 sm:grid-cols-2 gap-3">

                {specifications.map((spec, index) => (

                  <div

*key*={index}

*className*="flex items-center bg-white py-2 px-3 rounded-lg shadow-sm"

                  >

                    <CheckCircle

*className*="text-green-500 mr-2 flex-shrink-0"

*size*={16}

                    />

                    <p *className*="text-gray-700 text-sm">{spec}</p>

                  </div>

                ))}

              </div>

            </div>

          </div>

        </div>

      </div>

      {*/\* Media Modal \*/*}

      {showImageCarousel && (

        <MediaCarousel

*items*={getImages()}

*type*="image"

*onClose*={() => setShowImageCarousel(false)}

*currentIndex*={currentMediaIndex}

*setCurrentIndex*={setCurrentMediaIndex}

        />

      )}

      {showVideoCarousel && (

        <MediaCarousel

*items*={getVideos()}

*type*="video"

*onClose*={() => setShowVideoCarousel(false)}

*currentIndex*={currentMediaIndex}

*setCurrentIndex*={setCurrentMediaIndex}

        />

      )}

      {*/\* Toast notification \*/*}

      <ToastNotification *message*={toastMessage} />

    </>

  );

}

*export* default ServiceDetails;

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

import { Link, useParams, useLocation } from "react-router-dom";

import {

  FileImage,

  FileVideo,

  MapPin,

  Mail,

  Tag,

  CheckCircle,

  DollarSign,

  ChevronLeft,

  AlertCircle,

  Clock,

  Star,

  Share2,

  Heart,

  X,

  Phone,

  Calendar,

  MessageCircle,

  ArrowLeft,

  ArrowRight,

  MapPinCheckInsideIcon,

} from "lucide-react";

*// Custom hook for service details fetching with query parameters*

const useServiceDetails = (id) => {

  const [service, setService] = useState(null);

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

  const [error, setError] = useState(null);

  const webUrl = "https://backend.onetouchmoments.com/";

  const location = useLocation();

  useEffect(() => {

    const fetchServiceDetails = *async* () => {

      try {

*// Get query parameters from URL*

        const queryParams = new URLSearchParams(location.search);

        const firmName = queryParams.get("firm\_name");

        const vendorId = queryParams.get("vendor\_id") || id;

        if (!firmName) {

          throw new Error("Firm name not provided in URL");

        }

*// Construct API URL similar to HTML implementation*

        const ApiUrl = `${webUrl}user\_controller/Service/index\_get?data=${encodeURIComponent(

          firmName

        )}&vendor\_id=${encodeURIComponent(vendorId)}`;

        const response = await fetch(ApiUrl);

        if (!response.ok) {

          throw new Error("Failed to fetch service details");

        }

        const data = await response.json();

        console.log("Service Details API Response:", data);

        if (!data.data || data.data.length === 0) {

          throw new Error("No service found");

        }

        setService(data.data[0]); *// Using first record as shown in HTML implementation*

        setLoading(false);

      } catch (error) {

        console.error("Error fetching service details:", error);

        setError(error.message);

        setLoading(false);

      }

    };

    if (id) {

      fetchServiceDetails();

    }

  }, [id, webUrl, location.search]);

  return { service, loading, error, webUrl };

};

*// Service Details Component*

function ServiceDetails() {

  const { id } = useParams();

  const { service, loading, error, webUrl } = useServiceDetails(id);

  const [selectedMedia, setSelectedMedia] = useState(null);

  const [mainImage, setMainImage] = useState("");

  const [liked, setLiked] = useState(false);

  const [showImageCarousel, setShowImageCarousel] = useState(false);

  const [showVideoCarousel, setShowVideoCarousel] = useState(false);

  const [currentMediaIndex, setCurrentMediaIndex] = useState(0);

  const [showToast, setShowToast] = useState(false);

  const [toastMessage, setToastMessage] = useState("");

*// Function to handle main image and thumbnail gallery setup*

  useEffect(() => {

    if (service?.images) {

      const imageArray = service.images.split(",").filter((img) => img.trim());

      if (imageArray.length > 0) {

        setMainImage(`${webUrl}${imageArray[0].trim()}`);

      }

    }

  }, [service, webUrl]);

*// Share handler function*

  const ShareHandle = () => {

*// Copy the current page URL to the clipboard*

    navigator.clipboard

      .writeText(window.location.href)

      .then(() => {

        showToastNotification("Link copied to clipboard!");

      })

      .catch(() => {

        showToastNotification("Failed to copy link.");

      });

*// Share the current page URL to WhatsApp*

    const whatsappUrl = `https://api.whatsapp.com/send?text=${encodeURIComponent(

      window.location.href

    )}`;

    window.open(whatsappUrl, "\_blank");

  };

*// Show toast notification*

  const showToastNotification = (message) => {

    setToastMessage(message);

    setShowToast(true);

    setTimeout(() => setShowToast(false), 3000);

  };

*// Function to render thumbnail images*

  const getImages = () => {

    if (!service?.images) return [];

    return service.images.split(",").filter((img) => img.trim());

  };

  const getVideos = () => {

    if (!service?.videos) return [];

    return service.videos.split(",").filter((vid) => vid.trim());

  };

*// Function to display image as main preview*

  const setImageAsMain = (imagePath) => {

    setMainImage(`${webUrl}${imagePath}`);

  };

*// Handle like/favorite*

  const handleLikeToggle = () => {

    setLiked(!liked);

    showToastNotification(

      liked ? "Removed from favorites" : "Added to favorites"

    );

  };

  const MediaCarousel = ({

    items,

    type,

    onClose,

    currentIndex,

    setCurrentIndex,

  }) => {

    const goToNext = () => {

      setCurrentIndex((prev) => (prev === items.length - 1 ? 0 : prev + 1));

    };

    const goToPrevious = () => {

      setCurrentIndex((prev) => (prev === 0 ? items.length - 1 : prev - 1));

    };

    return (

      <div *className*="fixed inset-0 bg-black bg-opacity-95 z-50 flex flex-col items-center justify-center">

        <button

*onClick*={onClose}

*className*="absolute top-4 right-4 text-white hover:text-red-500 bg-black bg-opacity-50 p-2 rounded-full transition-all"

        >

          <X *size*={24} />

        </button>

        <div *className*="relative w-full max-w-6xl mx-auto px-4">

          <button

*onClick*={goToPrevious}

*className*="absolute left-4 top-1/2 -translate-y-1/2 text-white hover:text-red-500 bg-black bg-opacity-50 p-2 rounded-full transition-all"

          >

            <ArrowLeft *size*={24} />

          </button>

          <button

*onClick*={goToNext}

*className*="absolute right-4 top-1/2 -translate-y-1/2 text-white hover:text-red-500 bg-black bg-opacity-50 p-2 rounded-full transition-all"

          >

            <ArrowRight *size*={24} />

          </button>

          <div *className*="flex justify-center items-center h-[80vh]">

            {type === "image" ? (

              <img

*src*={`${webUrl}${items[currentIndex]}`}

*alt*={`Media ${currentIndex + 1}`}

*className*="max-h-full max-w-full object-contain"

              />

            ) : (

              <video

*controls*

*className*="max-h-full max-w-full"

*src*={`${webUrl}${items[currentIndex]}`}

              >

                Your browser does not support video playback.

              </video>

            )}

          </div>

          <div *className*="absolute left-4 right-4 bottom-4">

            <div *className*="flex justify-center gap-2 overflow-x-auto py-2">

              {items.map((item, index) => (

                <button

*key*={index}

*onClick*={() => setCurrentIndex(index)}

*className*={`w-16 h-16 flex-shrink-0 rounded-lg overflow-hidden border-2 transition-all

                    ${

                      currentIndex === index

                        ? "border-red-500 scale-110"

                        : "border-transparent opacity-70"

                    }`}

                >

                  {type === "image" ? (

                    <img

*src*={`${webUrl}${item}`}

*alt*={`Thumbnail ${index + 1}`}

*className*="w-full h-full object-cover"

                    />

                  ) : (

                    <div *className*="w-full h-full bg-gray-800 flex items-center justify-center">

                      <FileVideo *className*="text-white" *size*={24} />

                    </div>

                  )}

                </button>

              ))}

            </div>

          </div>

        </div>

      </div>

    );

  };

  const renderMediaPreviews = () => {

    const images = getImages();

    const videos = getVideos();

    return (

      <div *className*="space-y-4 mt-4">

        {*/\* Images preview \*/*}

        <div *className*="space-y-2">

          <div *className*="flex items-center gap-2">

            <FileImage *size*={18} *className*="text-gray-500" />

            <span *className*="font-medium">Images ({images.length})</span>

          </div>

          <div *className*="flex gap-2 overflow-x-auto pb-2">

            {images.slice(0, 4).map((image, index) => (

              <button

*key*={`img-${index}`}

*onClick*={() => {

                  setCurrentMediaIndex(index);

                  setShowImageCarousel(true);

                }}

*onMouseEnter*={() => setImageAsMain(image)}

*className*="relative flex-shrink-0 w-24 h-24 rounded-lg overflow-hidden transition-all hover:shadow-lg hover:scale-105 border-2 border-transparent hover:border-red-400"

              >

                <img

*src*={`${webUrl}${image}`}

*alt*={`Preview ${index + 1}`}

*className*="w-full h-full object-cover"

                />

                {index === 3 && images.length > 4 && (

                  <div *className*="absolute inset-0 bg-black bg-opacity-75 flex items-center justify-center text-white">

                    +{images.length - 4}

                  </div>

                )}

              </button>

            ))}

          </div>

        </div>

        {*/\* Videos preview \*/*}

        {videos.length > 0 && (

          <div *className*="space-y-2">

            <div *className*="flex items-center gap-2">

              <FileVideo *size*={18} *className*="text-gray-500" />

              <span *className*="font-medium">Videos ({videos.length})</span>

            </div>

            <div *className*="flex gap-2 overflow-x-auto pb-2">

              {videos.slice(0, 4).map((video, index) => (

                <button

*key*={`vid-${index}`}

*onClick*={() => {

                    setCurrentMediaIndex(index);

                    setShowVideoCarousel(true);

                  }}

*className*="relative flex-shrink-0 w-24 h-24 rounded-lg overflow-hidden bg-gray-100 transition-all hover:shadow-lg hover:scale-105 border-2 border-transparent hover:border-red-400"

                >

                  <div *className*="absolute inset-0 flex items-center justify-center">

                    <FileVideo *className*="text-red-500" *size*={24} />

                  </div>

                  <div *className*="absolute bottom-0 left-0 right-0 bg-black bg-opacity-60 py-1 text-white text-xs text-center">

                    Play

                  </div>

                  {index === 3 && videos.length > 4 && (

                    <div *className*="absolute inset-0 bg-black bg-opacity-75 flex items-center justify-center text-white">

                      +{videos.length - 4}

                    </div>

                  )}

                </button>

              ))}

            </div>

          </div>

        )}

      </div>

    );

  };

*// Skeleton loader*

  const SkeletonLoader = () => (

    <div *className*="max-w-6xl mx-auto p-6 animate-pulse">

      <div *className*="h-10 bg-gray-200 w-40 mb-6 rounded-lg"></div>

      <div *className*="flex flex-wrap">

        <div *className*="w-full md:w-1/3 pr-0 md:pr-6">

          <div *className*="h-64 bg-gray-200 rounded-lg mb-4"></div>

          <div *className*="flex justify-between space-x-2">

            {[...Array(3)].map((\_, i) => (

              <div *key*={i} *className*="h-20 w-20 bg-gray-200 rounded-lg"></div>

            ))}

          </div>

        </div>

        <div *className*="w-full md:w-2/3 mt-6 md:mt-0">

          <div *className*="h-10 bg-gray-200 w-3/4 mb-4 rounded-lg"></div>

          <div *className*="h-5 bg-gray-200 w-1/4 mb-6 rounded-lg"></div>

          <div *className*="space-y-2">

            <div *className*="h-4 bg-gray-200 rounded w-full"></div>

            <div *className*="h-4 bg-gray-200 rounded w-full"></div>

            <div *className*="h-4 bg-gray-200 rounded w-3/4"></div>

          </div>

          <div *className*="h-8 bg-gray-200 w-1/3 my-6 rounded-lg"></div>

          <div *className*="flex space-x-2">

            <div *className*="h-8 bg-gray-200 w-24 rounded-full"></div>

            <div *className*="h-8 bg-gray-200 w-24 rounded-full"></div>

          </div>

          <div *className*="mt-8 space-y-4">

            <div *className*="h-12 bg-gray-200 rounded-lg"></div>

            <div *className*="h-12 bg-gray-200 rounded-lg"></div>

          </div>

        </div>

      </div>

    </div>

  );

*// Error Component*

  const ErrorDisplay = ({ message }) => (

    <div *className*="flex flex-col items-center justify-center p-10 text-center min-h-screen bg-gray-50">

      <div *className*="bg-white p-8 rounded-2xl shadow-xl max-w-lg">

        <AlertCircle *className*="w-20 h-20 text-red-500 mb-6 mx-auto" />

        <h2 *className*="text-2xl font-bold text-gray-800 mb-4">

          Oops! Something went wrong

        </h2>

        <p *className*="text-gray-600 mb-8 bg-red-50 p-4 rounded-lg border-l-4 border-red-500">

          {message}

        </p>

        <Link

*to*="/services"

*className*="bg-red-600 text-white px-6 py-3 rounded-lg

          hover:bg-red-700 transition-colors flex items-center justify-center shadow-lg hover:shadow-xl"

        >

          <ChevronLeft *className*="mr-2" /> Back to Services

        </Link>

      </div>

    </div>

  );

*// Toast Notification Component*

  const ToastNotification = ({ message }) => (

    <div

*className*={`fixed bottom-4 right-4 bg-gray-800 text-white px-4 py-3 rounded-lg shadow-lg z-50 transform transition-all duration-300 ${

        showToast ? "translate-y-0 opacity-100" : "translate-y-12 opacity-0"

      }`}

    >

      {message}

    </div>

  );

*// Main render*

  if (loading) return <SkeletonLoader />;

  if (error) return <ErrorDisplay *message*={error} />;

  if (!service) return <ErrorDisplay *message*="Service not found" />;

*// Extract data from service object*

  const firmName = service.firm\_name;

  const description = service.description || "No description available";

  const price = service.off\_price || service.price || "0";

  const originalPrice = service.mrp\_price || "";

  const city = service.city || "";

  const state = service.state || "";

*// Calculate discount if applicable*

  const discountPercentage =

    originalPrice && price

      ? Math.round((1 - parseFloat(price) / parseFloat(originalPrice)) \* 100)

      : 0;

*// Use the first image from service or fallback to placeholder*

  const displayMainImage = mainImage;

*// Get specifications*

  const specifications = service.specification

    ? service.specification.split(",").map((spec) => spec.trim())

    : [

        "24/7 Support",

        "Free Consultation",

        "Satisfaction Guarantee",

        "Fast Response",

      ];

  return (

    <>

      <div *className*="max-w-6xl mx-auto bg-white shadow-2xl rounded-2xl border border-gray-200 my-10 overflow-hidden transform transition-all">

        {*/\* Header with back button and actions \*/*}

        <div *className*="p-4 md:p-6 bg-gradient-to-r from-gray-50 to-white border-b border-gray-200 flex justify-between items-center">

          <Link

*to*="/services"

*className*="inline-flex items-center text-red-600 hover:text-red-800

            transition-colors text-sm font-medium bg-white px-4 py-2 rounded-full shadow-sm hover:shadow group"

          >

            <ChevronLeft *className*="mr-1 group-hover:-translate-x-1 transition-transform" />

            Back to Services

          </Link>

          <div *className*="flex items-center space-x-3">

            <button

*className*="p-2 rounded-full hover:bg-gray-100 transition-colors"

*onClick*={ShareHandle}

*aria-label*="Share"

            >

              <Share2

*size*={20}

*className*="text-gray-600 hover:text-red-600 transition-colors"

              />

            </button>

            <button

*className*={`p-2 rounded-full hover:bg-red-50 transition-colors ${

                liked ? "text-red-500 bg-red-50" : "text-gray-600"

              }`}

*onClick*={handleLikeToggle}

*aria-label*={liked ? "Remove from favorites" : "Add to favorites"}

            >

              <Heart

*size*={20}

*fill*={liked ? "currentColor" : "none"}

*className*={liked ? "scale-110 transition-transform" : ""}

              />

            </button>

          </div>

        </div>

        <div *className*="flex flex-wrap">

          {*/\* Left column - Images \*/*}

          <div *className*="md:w-2/5 w-full p-4 md:p-6">

            <div *className*="relative rounded-xl overflow-hidden shadow-lg group">

              <img

*className*="w-full h-80 object-cover transition-transform duration-500 group-hover:scale-105"

*src*={displayMainImage}

*alt*="Service main"

*onClick*={() => setSelectedMedia(displayMainImage)}

              />

              {discountPercentage > 0 && (

                <div *className*="absolute top-4 right-4 bg-red-600 text-white px-3 py-1 rounded-full text-sm font-bold shadow-lg">

                  {discountPercentage}% OFF

                </div>

              )}

              <div *className*="absolute inset-0 bg-black opacity-0 group-hover:opacity-10 transition-opacity"></div>

              <div *className*="absolute bottom-4 right-4">

                <button

*className*="bg-white bg-opacity-90 hover:bg-opacity-100 p-2 rounded-full shadow-lg transform transition-transform hover:scale-110"

*onClick*={() => {

                    setCurrentMediaIndex(0);

                    setShowImageCarousel(true);

                  }}

                >

                  <FileImage *size*={20} *className*="text-red-600" />

                </button>

              </div>

            </div>

            {*/\* Thumbnails \*/*}

            {renderMediaPreviews()}

          </div>

          {*/\* Right column - Details \*/*}

          <div *className*="md:w-3/5 w-full p-6 md:p-8">

            {*/\* Service name and rating \*/*}

            <div *className*="flex flex-col md:flex-row justify-between items-start md:items-center">

              <h1 *className*="text-3xl font-bold text-gray-800 mb-2 md:mb-0 leading-tight">

                {firmName}

              </h1>

              <div *className*="flex items-center text-yellow-500 bg-yellow-50 px-3 py-1 rounded-lg">

                <Star *size*={18} *fill*="currentColor" />

                <Star *size*={18} *fill*="currentColor" />

                <Star *size*={18} *fill*="currentColor" />

                <Star *size*={18} *fill*="currentColor" />

                <Star *size*={18} *fill*="none" *stroke*="currentColor" />

                <span *className*="ml-2 text-gray-700 font-medium">(4.0)</span>

              </div>

            </div>

            {*/\* Service description \*/*}

            <div *className*="mt-6 text-gray-600 bg-gray-50 p-4 rounded-lg border-l-4 border-red-400">

              <p *className*="leading-relaxed">{description}</p>

            </div>

            {*/\* Price section \*/*}

            <div *className*="mt-6 flex items-baseline gap-3">

              <div *className*="text-3xl font-bold text-red-600">

                ₹ {parseFloat(price).toLocaleString()}

              </div>

              {originalPrice &&

                parseFloat(originalPrice) > parseFloat(price) && (

                  <div *className*="text-xl line-through text-gray-400">

                    ₹ {parseFloat(originalPrice).toLocaleString()}

                  </div>

                )}

              {discountPercentage > 0 && (

                <div *className*="bg-green-100 text-green-800 px-3 py-1 rounded-full text-sm font-medium shadow-sm">

                  Save {discountPercentage}%

                </div>

              )}

            </div>

            {*/\* Location and attributes \*/*}

            <div *className*="mt-6 flex flex-wrap items-center gap-3">

              {(city || state) && (

                <div *className*="flex items-center bg-blue-50 py-2 px-4 rounded-full shadow-sm">

                  <MapPin *className*="text-blue-600" *size*={16} />

                  <p *className*="text-gray-700 text-sm ml-1 font-medium">

                    {city}

                    {city && state && ", "}

                    {state}

                  </p>

                </div>

              )}

              <div *className*="flex items-center bg-purple-50 py-2 px-4 rounded-full shadow-sm">

                <Clock *className*="text-purple-600" *size*={16} />

                <p *className*="text-gray-700 text-sm ml-1 font-medium">

                  24/7 Service

                </p>

              </div>

              {service.is\_verified && (

                <div *className*="flex items-center bg-green-50 py-2 px-4 rounded-full shadow-sm">

                  <CheckCircle *className*="text-green-600" *size*={16} />

                  <p *className*="text-gray-700 text-sm ml-1 font-medium">

                    Verified Provider

                  </p>

                </div>

              )}

            </div>

            {*/\* Action buttons \*/*}

            <div *className*="mt-8">

              <div *className*="flex flex-col md:flex-row gap-4">

                <button

*className*="bg-gradient-to-r from-red-600 to-red-700 hover:from-red-700 hover:to-red-800

                  text-white font-medium py-3 px-6 rounded-xl flex-1 transition-all duration-300

                  transform hover:scale-105 shadow-lg hover:shadow-xl flex justify-center items-center"

                >

                  <Calendar *className*="mr-2" *size*={18} />

                  Book Now

                </button>

              </div>

            </div>

            {*/\* Service highlights \*/*}

            <div *className*="mt-8 bg-gray-50 p-5 rounded-xl border border-gray-200">

              <h3 *className*="font-semibold text-gray-800 mb-4 flex items-center">

                <Tag *className*="mr-2 text-red-600" *size*={18} />

                Contact Details

              </h3>

              <div *className*="grid grid-cols-1 sm:grid-cols-2 gap-3">

                <div *className*="flex flex-col bg-white py-3 px-4 rounded-lg shadow-sm">

                  <div *className*="flex items-center mb-2">

                    <MapPinCheckInsideIcon

*className*="text-green-500 mr-2 flex-shrink-0"

*size*={16}

                    />

                    <p *className*="text-gray-700 text-sm">

                      Pincode: {service.pincode}

                    </p>

                  </div>

                  <p *className*="text-gray-700 text-sm ml-6 mb-1">

                    State: {service.state}

                  </p>

                  <p *className*="text-gray-700 text-sm ml-6 mb-1">

                    Status: {service.status}

                  </p>

                  <p *className*="text-gray-700 text-sm ml-6 mb-1">

                    District: {service.district}

                  </p>

                  <p *className*="text-gray-700 text-sm ml-6 mb-1">

                    Firm Name: {service.firm\_name}

                  </p>

                  <p *className*="text-gray-700 text-sm ml-6">

                    Service Category: {service.service\_cat}

                  </p>

                </div>

              </div>

            </div>

            <div *className*="mt-8 bg-gray-50 p-5 rounded-xl border border-gray-200">

              <h3 *className*="font-semibold text-gray-800 mb-4 flex items-center">

                <Tag *className*="mr-2 text-red-600" *size*={18} />

                Service Include

              </h3>

              <div *className*="grid grid-cols-1 sm:grid-cols-2 gap-3">

                {specifications.map((spec, index) => (

                  <div

*key*={index}

*className*="flex items-center bg-white py-2 px-3 rounded-lg shadow-sm"

                  >

                    <CheckCircle

*className*="text-green-500 mr-2 flex-shrink-0"

*size*={16}

                    />

                    <p *className*="text-gray-700 text-sm">{spec}</p>

                  </div>

                ))}

              </div>

            </div>

          </div>

        </div>

      </div>

      {*/\* Media Modal \*/*}

      {showImageCarousel && (

        <MediaCarousel

*items*={getImages()}

*type*="image"

*onClose*={() => setShowImageCarousel(false)}

*currentIndex*={currentMediaIndex}

*setCurrentIndex*={setCurrentMediaIndex}

        />

      )}

      {showVideoCarousel && (

        <MediaCarousel

*items*={getVideos()}

*type*="video"

*onClose*={() => setShowVideoCarousel(false)}

*currentIndex*={currentMediaIndex}

*setCurrentIndex*={setCurrentMediaIndex}

        />

      )}

      {*/\* Toast notification \*/*}

      <ToastNotification *message*={toastMessage} />

    </>

  );

}

*export* default ServiceDetails;