Product Components - Complete Source Code & Documentation

Table of Contents

Products Grid

components/products/ProductsGrid.tsx

Purpose: Main grid layout for displaying wine products **Location**: `/components/products/ProductsGrid.tsx`

```
import { Card, CardContent } from "@/components/ui/card";
import Link from "next/link";
import Image from "next/image";
import FavoriteToggleButton from "./FavoriteToggleButton";
import { Badge } from "@/components/ui/badge";
import { formatCurrency } from "@/lib/mock-data";
import { cn, wineVariants } from "@/lib/design-system";
import { Star, MapPin } from "lucide-react";

interface Product {
  id: string;
  name: string;
  price: number;
  type: string;
  featured: boolean;
  images: { url: string }[];
  region: { name: string; country: string };
  averageRating?: number;
}

interface ProductsGridProps {
  products: Product[];
}

function ProductsGrid({ products }: ProductsGridProps) {
  return (
    <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 xl:grid-
cols-4 gap-6">
      {products.map((product) => (
        <ProductCard key={product.id} product={product} />
      ))}
    </div>
  );
}

function ProductCard({ product }: { product: Product }) {
  const { name, images, featured, price, type, region, averageRating } =
    product;

  if (!images?.length || !images[0]?.url) return null;
  const imageUrl = images[0].url;
  const formattedPrice = formatCurrency(price);

  return (
    <article className="group relative">
      <Link href={`#`} className="block h-full">
        <Card
          className={cn(
            "h-full overflow-hidden transition-all duration-300",
            "hover:shadow-xl hover:-translate-y-1",
            "border-wine-100 hover:border-wine-200"
          )}
        >
          {/* Image Container */}
          <div className="relative aspect-[3/4] overflow-hidden">
            <Image
              src={imageUrl || "/placeholder.svg"}
              alt={name}
              fill
              className="object-cover transition-transform duration-300 group-
hover:scale-105"
              sizes="(max-width: 640px) 100vw, (max-width: 1024px) 50vw, (max-
width: 1280px) 33vw, 25vw"
            />

            {/* Featured Badge */}
            {featured && (
              <Badge
                className={cn(
                  "absolute top-3 left-3 z-10",
                  wineVariants.button.primary,
                  "text-xs font-medium"
                )}
              >
                Featured
              </Badge>
            )}

            {/* Favorite Button */}
            <div className="absolute top-3 right-3 z-10 opacity-0 group-
hover:opacity-100 transition-opacity duration-200">
              <FavoriteToggleButton productId={product.id} />
            </div>

            {/* Price Overlay */}
            <div className="absolute bottom-0 left-0 right-0 bg-gradient-to-t
from-black/60 to-transparent p-4">
              <div className="text-white font-bold text-lg">
                {formattedPrice}
              </div>
            </div>
          </div>

          {/* Content */}
          <CardContent className="p-4 space-y-3">
            <div className="space-y-2">
              <h3 className="font-semibold text-lg leading-tight line-clamp-2
group-hover:text-wine-600 transition-colors">
                {name}
              </h3>

              <div className="flex items-center justify-between text-sm text-
muted-foreground">
                <span className="capitalize font-medium">{type}</span>
                {averageRating && (
                  <div className="flex items-center gap-1">
```

```
                    <Star className="h-4 w-4 fill-yellow-400 text-yellow-400"
/>
                    <span className="font-medium">
                      {averageRating.toFixed(1)}
                    </span>
                  </div>
                )}
              </div>

              {region && (
                <div className="flex items-center gap-1 text-xs text-muted-
foreground">
                  <MapPin className="h-3 w-3" />
                  <span>
                    {region.name}, {region.country}
                  </span>
                </div>
              )}
            </div>
          </CardContent>
        </Card>
      </Link>
    </article>
  );
}

export default ProductsGrid;
```

Products Grid Features Explained:

- **Responsive Grid**: Adapts from 1 to 4 columns based on screen size
- **Product Cards**: Individual product display with hover effects
- **Image Optimization**: Next.js Image component with proper sizing
- **Featured Badges**: Highlights premium products
- **Favorite Toggle**: Quick add/remove from favorites
- **Price Overlay**: Prominent price display on images
- **Rating Display**: Shows user ratings when available

Filter Sidebar

components/products/FilterSidebar.tsx

Purpose: Sidebar for filtering and sorting wine products **Location**: `/components/products/FilterSidebar.tsx`

```
"use client";

import { useState } from "react";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Label } from "@/components/ui/label";
import { Checkbox } from "@/components/ui/checkbox";
import {
  Select,
  SelectContent,
  SelectItem,
  SelectTrigger,
  SelectValue,
} from "@/components/ui/select";
import { Slider } from "@/components/ui/slider";
import { Separator } from "@/components/ui/separator";
import { X, Filter, SortAsc, SortDesc } from "lucide-react";
import { cn } from "@/lib/utils";
```

```
interface FilterSidebarProps {
  isOpen: boolean;
  onClose: () => void;
  onFiltersChange: (filters: any) => void;
}

const wineTypes = [
  "Red Wine",
  "White Wine",
  "Rosé Wine",
  "Sparkling Wine",
  "Dessert Wine",
  "Fortified Wine",
];

const regions = [
  "France",
  "Italy",
  "Spain",
  "United States",
  "Australia",
  "Argentina",
  "Chile",
  "Germany",
  "New Zealand",
  "South Africa",
];

const priceRanges = [
  { label: "Under $20", min: 0, max: 20 },
  { label: "$20 - $50", min: 20, max: 50 },
  { label: "$50 - $100", min: 50, max: 100 },
  { label: "$100 - $200", min: 100, max: 200 },
  { label: "Over $200", min: 200, max: 1000 },
];

export default function FilterSidebar({
  isOpen,
  onClose,
  onFiltersChange,
}: FilterSidebarProps) {
  const [filters, setFilters] = useState({
    search: "",
    types: [] as string[],
    regions: [] as string[],
    priceRange: [0, 500] as [number, number],
    rating: 0,
    featured: false,
    sortBy: "name",
    sortOrder: "asc" as "asc" | "desc",
  });

  const handleFilterChange = (key: string, value: any) => {
    const newFilters = { ...filters, [key]: value };
    setFilters(newFilters);
    onFiltersChange(newFilters);
  };

  const handleTypeToggle = (type: string) => {
    const newTypes = filters.types.includes(type)
      ? filters.types.filter((t) => t !== type)
      : [...filters.types, type];
    handleFilterChange("types", newTypes);
  };
  const handleRegionToggle = (region: string) => {
    const newRegions = filters.regions.includes(region)
      ? filters.regions.filter((r) => r !== region)
      : [...filters.regions, region];
    handleFilterChange("regions", newRegions);
  };

  const clearFilters = () => {
    const clearedFilters = {
      search: "",
      types: [],
      regions: [],
      priceRange: [0, 500],
      rating: 0,
      featured: false,
      sortBy: "name",
      sortOrder: "asc" as "asc" | "desc",
    };
    setFilters(clearedFilters);
    onFiltersChange(clearedFilters);
  };

  const hasActiveFilters = Object.values(filters).some((value) => {
    if (Array.isArray(value)) return value.length > 0;
    if (typeof value === "number") return value > 0;
    if (typeof value === "string") return value !== "";
    return value === true;
  });

  return (
    <div
      className={cn(
        "fixed inset-y-0 left-0 z-50 w-80 bg-background border-r border-border
transform transition-transform duration-300 ease-in-out",
        isOpen ? "translate-x-0" : "-translate-x-full"
      )}
    >
      <div className="flex flex-col h-full">
        {/* Header */}
        <div className="flex items-center justify-between p-4 border-b border-
border">
          <div className="flex items-center gap-2">
            <Filter className="h-5 w-5" />
            <h2 className="text-lg font-semibold">Filters</h2>
          </div>
          <Button variant="ghost" size="sm" onClick={onClose}>
            <X className="h-5 w-5" />
          </Button>
        </div>

        {/* Content */}
        <div className="flex-1 overflow-y-auto p-4 space-y-6">
          {/* Search */}
          <div className="space-y-2">
            <Label htmlFor="search">Search Wines</Label>
            <Input
              id="search"
              placeholder="Search by name, region, or type..."
              value={filters.search}
              onChange={(e) => handleFilterChange("search", e.target.value)}
            />
          </div>

          {/* Sort */}
          <div className="space-y-2">
            <Label>Sort By</Label>
```

```
              <div className="flex gap-2">
                <Select
                  value={filters.sortBy}
                  onValueChange={(value) => handleFilterChange("sortBy", value)}
                >
                  <SelectTrigger className="flex-1">
                    <SelectValue />
                  </SelectTrigger>
                  <SelectContent>
                    <SelectItem value="name">Name</SelectItem>
                    <SelectItem value="price">Price</SelectItem>
                    <SelectItem value="rating">Rating</SelectItem>
                    <SelectItem value="date">Date Added</SelectItem>
                  </SelectContent>
                </Select>
                <Button
                  variant="outline"
                  size="icon"
                  onClick={() =>
                    handleFilterChange(
                      "sortOrder",
                      filters.sortOrder === "asc" ? "desc" : "asc"
                    )
                  }
                >
                  {filters.sortOrder === "asc" ? (
                    <SortAsc className="h-4 w-4" />
                  ) : (
                    <SortDesc className="h-4 w-4" />
                  )}
                </Button>
              </div>
            </div>

            <Separator />

            {/* Wine Types */}
            <div className="space-y-3">
              <Label>Wine Types</Label>
              <div className="space-y-2">
                {wineTypes.map((type) => (
                  <div key={type} className="flex items-center space-x-2">
                    <Checkbox
                      id={type}
                      checked={filters.types.includes(type)}
                      onCheckedChange={() => handleTypeToggle(type)}
                    />
                    <Label
                      htmlFor={type}
                      className="text-sm font-normal cursor-pointer"
                    >
                      {type}
                    </Label>
                  </div>
                ))}
              </div>
            </div>

            <Separator />

            {/* Regions */}
            <div className="space-y-3">
              <Label>Regions</Label>
              <div className="space-y-2 max-h-40 overflow-y-auto">
                {regions.map((region) => (
                  <div key={region} className="flex items-center space-x-2">
```

```
                    <Checkbox
                      id={region}
                      checked={filters.regions.includes(region)}
                      onCheckedChange={() => handleRegionToggle(region)}
                    />
                    <Label
                      htmlFor={region}
                      className="text-sm font-normal cursor-pointer"
                    >
                      {region}
                    </Label>
                  </div>
                ))}
              </div>
            </div>

            <Separator />

            {/* Price Range */}
            <div className="space-y-3">
              <Label>Price Range</Label>
              <div className="px-2">
                <Slider
                  value={filters.priceRange}
                  onValueChange={(value) =>
                    handleFilterChange("priceRange", value)
                  }
                  max={500}
                  min={0}
                  step={10}
                  className="w-full"
                />
                <div className="flex justify-between text-sm text-muted-
foreground mt-2">
                  <span>${filters.priceRange[0]}</span>
                  <span>${filters.priceRange[1]}</span>
                </div>
              </div>
            </div>

            <Separator />

            {/* Rating */}
            <div className="space-y-3">
              <Label>Minimum Rating</Label>
              <div className="px-2">
                <Slider
                  value={[filters.rating]}
                  onValueChange={(value) =>
                    handleFilterChange("rating", value[0])
                  }
                  max={5}
                  min={0}
                  step={0.5}
                  className="w-full"
                />
                <div className="text-center text-sm text-muted-foreground mt-2">
                  {filters.rating} stars
                </div>
              </div>
            </div>

            <Separator />

            {/* Featured Only */}
            <div className="space-y-3">
              <div className="flex items-center space-x-2">
                <Checkbox
                  id="featured"
                  checked={filters.featured}
                  onCheckedChange={(checked) =>
                    handleFilterChange("featured", checked)
                  }
                />
                <Label
                  htmlFor="featured"
                  className="text-sm font-normal cursor-pointer"
                >
                  Featured Wines Only
                </Label>
              </div>
            </div>
          </div>

          {/* Footer */}
          <div className="p-4 border-t border-border">
            <div className="flex gap-2">
              <Button
                variant="outline"
                className="flex-1"
                onClick={clearFilters}
                disabled={!hasActiveFilters}
              >
                Clear All
              </Button>
              <Button className="flex-1" onClick={onClose}>
                Apply Filters
              </Button>
            </div>
          </div>
        </div>
      </div>
    );
  }
```

Filter Sidebar Features Explained:

- **Search Functionality**: Text search across product attributes
- **Type Filtering**: Filter by wine types (red, white, rosé, etc.)
- **Region Filtering**: Filter by wine regions and countries
- **Price Range**: Slider for price range selection
- **Rating Filter**: Minimum rating threshold
- **Sorting Options**: Multiple sort criteria with direction
- **Featured Filter**: Show only featured wines
- **Responsive Design**: Mobile-friendly sidebar layout

Products Container

components/products/ProductsContainer.tsx

Purpose: Container component that manages product filtering and display **Location**: `/components/products/ProductsContainer.tsx`

```
"use client";

import { useState, useEffect } from "react";
import { Button } from "@/components/ui/button";
import { Filter, Grid, List } from "lucide-react";
import ProductsGrid from "./ProductsGrid";
import ProductsList from "./ProductsList";
import FilterSidebar from "./FilterSidebar";
import { cn } from "@/lib/utils";

interface Product {
  id: string;
  name: string;
  price: number;
  type: string;
  featured: boolean;
  images: { url: string }[];
  region: { name: string; country: string };
  averageRating?: number;
}

interface ProductsContainerProps {
  initialProducts: Product[];
}

export default function ProductsContainer({
  initialProducts,
}: ProductsContainerProps) {
  const [products, setProducts] = useState<Product[]>(initialProducts);
  const [filteredProducts, setFilteredProducts] =
    useState<Product[]>(initialProducts);
  const [isFilterOpen, setIsFilterOpen] = useState(false);
  const [viewMode, setViewMode] = useState<"grid" | "list">("grid");
  const [filters, setFilters] = useState({});

  useEffect(() => {
    // Apply filters to products
    let filtered = [...products];

    // Apply search filter
    if (filters.search) {
      const searchTerm = filters.search.toLowerCase();
      filtered = filtered.filter(
        (product) =>
          product.name.toLowerCase().includes(searchTerm) ||
          product.type.toLowerCase().includes(searchTerm) ||
          product.region.name.toLowerCase().includes(searchTerm) ||
          product.region.country.toLowerCase().includes(searchTerm)
      );
    }

    // Apply type filter
    if (filters.types && filters.types.length > 0) {
      filtered = filtered.filter((product) =>
        filters.types.includes(product.type)
      );
    }

    // Apply region filter
    if (filters.regions && filters.regions.length > 0) {
      filtered = filtered.filter(
        (product) =>
          filters.regions.includes(product.region.name) ||
          filters.regions.includes(product.region.country)
      );
    }

    // Apply price filter
    if (filters.priceRange) {
      filtered = filtered.filter(
        (product) =>
          product.price >= filters.priceRange[0] &&
          product.price <= filters.priceRange[1]
      );
    }

    // Apply rating filter
    if (filters.rating > 0) {
      filtered = filtered.filter(
        (product) =>
          product.averageRating && product.averageRating >= filters.rating
      );
    }

    // Apply featured filter
    if (filters.featured) {
      filtered = filtered.filter((product) => product.featured);
    }

    // Apply sorting
    if (filters.sortBy) {
      filtered.sort((a, b) => {
        let aValue: any, bValue: any;

        switch (filters.sortBy) {
          case "name":
            aValue = a.name.toLowerCase();
            bValue = b.name.toLowerCase();
            break;
          case "price":
            aValue = a.price;
            bValue = b.price;
            break;
          case "rating":
            aValue = a.averageRating || 0;
            bValue = b.averageRating || 0;
            break;
          case "date":
            // Assuming products have a date field
            aValue = new Date(a.createdAt || 0);
            bValue = new Date(b.createdAt || 0);
            break;
          default:
            return 0;
        }

        if (filters.sortOrder === "desc") {
          return aValue < bValue ? 1 : -1;
        }
        return aValue > bValue ? 1 : -1;
      });
    }

    setFilteredProducts(filtered);
  }, [products, filters]);

  const handleFiltersChange = (newFilters: any) => {
    setFilters(newFilters);
  };

  const toggleFilter = () => {
    setIsFilterOpen(!isFilterOpen);
  };

  return (
    <div className="relative">
      {/* Header */}
      <div className="flex items-center justify-between mb-6">
```

```
          <div className="flex items-center gap-4">
            <Button
              variant="outline"
              onClick={toggleFilter}
              className="flex items-center gap-2"
            >
              <Filter className="h-4 w-4" />
              Filters
              {Object.keys(filters).length > 0 && (
                <span className="ml-2 px-2 py-1 text-xs bg-primary text-primary-
foreground rounded-full">
                  {
                    Object.keys(filters).filter((key) => {
                      const value = filters[key];
                      if (Array.isArray(value)) return value.length > 0;
                      if (typeof value === "number") return value > 0;
                      if (typeof value === "string") return value !== "";
                      return value === true;
                    }).length
                  }
                </span>
              )}
            </Button>
          </div>

          <div className="flex items-center gap-2">
            <span className="text-sm text-muted-foreground">
              {filteredProducts.length} of {products.length} wines
            </span>

            <div className="flex items-center border border-border rounded-md">
              <Button
                variant={viewMode === "grid" ? "default" : "ghost"}
                size="sm"
                onClick={() => setViewMode("grid")}
                className={cn(
                  "rounded-r-none border-r",
                  viewMode === "grid" && "bg-primary text-primary-foreground"
                )}
              >
                <Grid className="h-4 w-4" />
              </Button>
              <Button
                variant={viewMode === "list" ? "default" : "ghost"}
                size="sm"
                onClick={() => setViewMode("list")}
                className={cn(
                  "rounded-l-none",
                  viewMode === "list" && "bg-primary text-primary-foreground"
                )}
              >
                <List className="h-4 w-4" />
              </Button>
            </div>
          </div>
        </div>

        {/* Products Display */}
        <div className="min-h-[400px]">
          {filteredProducts.length === 0 ? (
            <div className="text-center py-12">
              <h3 className="text-lg font-semibold text-muted-foreground mb-2">
                No wines found
              </h3>
              <p className="text-muted-foreground">
                Try adjusting your filters or search terms
              </p>
            </div>
          ) : (
            <>
              {viewMode === "grid" ? (
                <ProductsGrid products={filteredProducts} />
              ) : (
                <ProductsList products={filteredProducts} />
              )}
            </>
          )}
        </div>

        {/* Filter Sidebar */}
        <FilterSidebar
          isOpen={isFilterOpen}
          onClose={() => setIsFilterOpen(false)}
          onFiltersChange={handleFiltersChange}
        />

        {/* Overlay */}
        {isFilterOpen && (
          <div
            className="fixed inset-0 bg-black/50 z-40"
            onClick={() => setIsFilterOpen(false)}
          />
        )}
      </div>
    );
  }
```

Products Container Features Explained:

- **State Management**: Manages products, filters, and view modes
- **Filter Application**: Applies multiple filter criteria to products
- **View Modes**: Toggle between grid and list views
- **Filter Sidebar**: Integrates with filter sidebar component
- **Search Integration**: Real-time search across product attributes
- **Sorting**: Multiple sort options with direction control
- **Responsive Design**: Mobile-friendly filter controls

Products List

components/products/ProductsList.tsx

Purpose: List view layout for displaying wine products **Location**: `/components/products/ProductsList.tsx`

```
import Link from "next/link";
import Image from "next/image";
import { Card, CardContent } from "@/components/ui/card";
import { Badge } from "@/components/ui/badge";
import { Button } from "@/components/ui/button";
import { formatCurrency } from "@/lib/mock-data";
import { cn, wineVariants } from "@/lib/design-system";
import { Star, MapPin, Heart, ShoppingCart } from "lucide-react";
import FavoriteToggleButton from "./FavoriteToggleButton";

interface Product {
  id: string;
  name: string;
  price: number;
  type: string;
  featured: boolean;
  images: { url: string }[];
  region: { name: string; country: string };
  averageRating?: number;
  description?: string;
}

interface ProductsListProps {
  products: Product[];
}

export default function ProductsList({ products }: ProductsListProps) {
  return (
    <div className="space-y-4">
      {products.map((product) => (
        <ProductListItem key={product.id} product={product} />
      ))}
    </div>
  );
}

function ProductListItem({ product }: { product: Product }) {
  const {
    name,
    images,
    featured,
    price,
    type,
    region,
    averageRating,
    description,
  } = product;

  if (!images?.length || !images[0]?.url) return null;

  const imageUrl = images[0].url;
  const formattedPrice = formatCurrency(price);

  return (
    <Card className="group hover:shadow-lg transition-all duration-300">
      <CardContent className="p-6">
        <div className="flex gap-6">
          {/* Image */}
          <div className="relative w-32 h-32 flex-shrink-0">
            <Image
              src={imageUrl || "/placeholder.svg"}
              alt={name}
              fill
              className="object-cover rounded-lg transition-transform duration-
300 group-hover:scale-105"
              sizes="128px"
            />

            {featured && (
              <Badge
                className={cn(
                  "absolute top-2 left-2",
                  wineVariants.button.primary,
                  "text-xs font-medium"
                )}
              >
                Featured
              </Badge>
            )}
          </div>
          {/* Content */}
          <div className="flex-1 min-w-0">
            <div className="flex items-start justify-between gap-4">
              <div className="flex-1 min-w-0">
                <h3 className="text-xl font-semibold text-foreground group-
hover:text-primary transition-colors line-clamp-2">
                  {name}
                </h3>

                {description && (
                  <p className="text-muted-foreground mt-2 line-clamp-2">
                    {description}
                  </p>
                )}

                <div className="flex items-center gap-4 mt-3 text-sm text-
muted-foreground">
                  <span className="capitalize font-medium">{type}</span>
                  {averageRating && (
                    <div className="flex items-center gap-1">
                      <Star className="h-4 w-4 fill-yellow-400 text-yellow-400"
/>
                      <span className="font-medium">
                        {averageRating.toFixed(1)}
                      </span>
                    </div>
                  )}
                  {region && (
                    <div className="flex items-center gap-1">
                      <MapPin className="h-3 w-3" />
                      <span>
                        {region.name}, {region.country}
                      </span>
                    </div>
                  )}
                </div>
              </div>

              <div className="flex flex-col items-end gap-3">
                <div className="text-right">
                  <div className="text-2xl font-bold text-foreground">
                    {formattedPrice}
                  </div>
                </div>

                <div className="flex items-center gap-2">
                  <FavoriteToggleButton productId={product.id} />
                  <Button size="sm" className="flex items-center gap-2">
                    <ShoppingCart className="h-4 w-4" />
                    Add to Cart
                  </Button>
                </div>
              </div>
            </div>
          </div>
        </div>
      </CardContent>
    </Card>
  );
}
```

Products List Features Explained:

- **List Layout**: Horizontal layout with image and content
- **Product Details**: Comprehensive product information display

- **Action Buttons**: Quick add to favorites and cart
- **Hover Effects**: Smooth transitions and visual feedback
- **Responsive Design**: Adapts to different screen sizes
- **Image Optimization**: Proper image sizing and loading

Favorite Toggle Button

components/products/FavoriteToggleButton.tsx

Purpose: Button for adding/removing products from favorites **Location**: `/components/products/FavoriteToggleButton.tsx`

```
"use client";

import { useState } from "react";
import { Button } from "@/components/ui/button";
import { Heart } from "lucide-react";
import { cn } from "@/lib/utils";
import { useFavorites } from "@/utils/actions";

interface FavoriteToggleButtonProps {
  productId: string;
  className?: string;
}

export default function FavoriteToggleButton({
  productId,
  className,
}: FavoriteToggleButtonProps) {
  const [isAnimating, setIsAnimating] = useState(false);
  const { isFavorite, toggleFavorite } = useFavorites(productId);

  const handleToggle = async () => {
    if (isAnimating) return;

    setIsAnimating(true);
    await toggleFavorite();

    setTimeout(() => {
      setIsAnimating(false);
    }, 300);
  };

  return (
    <Button
      variant="ghost"
      size="icon"
      onClick={handleToggle}
      className={cn(
        "h-8 w-8 rounded-full transition-all duration-300",
        isFavorite
          ? "bg-red-500 text-white hover:bg-red-600"
          : "bg-white/90 text-gray-600 hover:bg-white hover:text-red-500",
        isAnimating && "scale-110",
        className
      )}
      aria-label={isFavorite ? "Remove from favorites" : "Add to favorites"}
    >
      <Heart
        className={cn(
          "h-4 w-4 transition-all duration-300",
          isFavorite && "fill-current"
        )}
      />
    </Button>
  );
}
```

Favorite Toggle Button Features Explained:

- **State Management**: Tracks favorite status and animation state
- **Visual Feedback**: Different styles for favorited/unfavorited states
- **Animation**: Smooth scale animation on click
- **Accessibility**: Proper ARIA labels for screen readers
- **Integration**: Uses favorites context for state management
- **Responsive Design**: Adapts to different button sizes

Products Grid With Auth

components/products/ProductsGridWithAuth.tsx

Purpose: Products grid with authentication-aware features **Location**: `/components/products/ProductsGridWithAuth.tsx`

```
"use client";

import { useAuth } from "@clerk/nextjs";
import { Card, CardContent } from "@/components/ui/card";
import { Badge } from "@/components/ui/badge";
import { Button } from "@/components/ui/button";
import { formatCurrency } from "@/lib/mock-data";
import { cn, wineVariants } from "@/lib/design-system";
import { Star, MapPin, Heart, ShoppingCart, Lock } from "lucide-react";
import Link from "next/link";
import Image from "next/image";
import FavoriteToggleButton from "./FavoriteToggleButton";

interface Product {
  id: string;
  name: string;
  price: number;
  type: string;
  featured: boolean;
  images: { url: string }[];
  region: { name: string; country: string };
  averageRating?: number;
  requiresAuth?: boolean;
}

interface ProductsGridWithAuthProps {
  products: Product[];
}

export default function ProductsGridWithAuth({
  products,
}: ProductsGridWithAuthProps) {
  const { isSignedIn } = useAuth();

  return (
    <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 xl:grid-
cols-4 gap-6">
      {products.map((product) => (
        <ProductCardWithAuth
          key={product.id}
          product={product}
          isSignedIn={isSignedIn}
        />
      ))}
    </div>
  );
}

function ProductCardWithAuth({
  product,
  isSignedIn,
}: {
  product: Product;
  isSignedIn: boolean;
}) {
  const {
    name,
    images,
    featured,
    price,
    type,
    region,
    averageRating,
    requiresAuth,
  } = product;

  if (!images?.length || !images[0]?.url) return null;

  const imageUrl = images[0].url;
  const formattedPrice = formatCurrency(price);
  const canInteract = isSignedIn || !requiresAuth;

  return (
    <article className="group relative">
      <Link href={`#`} className="block h-full">
        <Card
          className={cn(
            "h-full overflow-hidden transition-all duration-300",
            "hover:shadow-xl hover:-translate-y-1",
            "border-wine-100 hover:border-wine-200",
            !canInteract && "opacity-75"
          )}
        >
          {/* Image Container */}
          <div className="relative aspect-[3/4] overflow-hidden">
            <Image
              src={imageUrl || "/placeholder.svg"}
              alt={name}
              fill
              className="object-cover transition-transform duration-300 group-
hover:scale-105"
              sizes="(max-width: 640px) 100vw, (max-width: 1024px) 50vw, (max-
width: 1280px) 33vw, 25vw"
            />

            {/* Featured Badge */}
            {featured && (
              <Badge
                className={cn(
                  "absolute top-3 left-3 z-10",
                  wineVariants.button.primary,
                  "text-xs font-medium"
                )}
              >
                Featured
              </Badge>
            )}

            {/* Auth Required Badge */}
            {requiresAuth && !isSignedIn && (
              <Badge
                variant="secondary"
                className="absolute top-3 right-3 z-10 bg-yellow-500 text-
white"
              >
                <Lock className="h-3 w-3 mr-1" />
                Sign In
              </Badge>
            )}

            {/* Favorite Button */}
            {canInteract && (
              <div className="absolute top-3 right-3 z-10 opacity-0 group-
hover:opacity-100 transition-opacity duration-200">
                <FavoriteToggleButton productId={product.id} />
              </div>
            )}

            {/* Price Overlay */}
            <div className="absolute bottom-0 left-0 right-0 bg-gradient-to-t
from-black/60 to-transparent p-4">
              <div className="text-white font-bold text-lg">
                {formattedPrice}
              </div>
            </div>
          </div>

          {/* Content */}
          <CardContent className="p-4 space-y-3">
            <div className="space-y-2">
              <h3 className="font-semibold text-lg leading-tight line-clamp-2
group-hover:text-wine-600 transition-colors">
                {name}
              </h3>

              <div className="flex items-center justify-between text-sm text-
muted-foreground">
                <span className="capitalize font-medium">{type}</span>
                {averageRating && (
                  <div className="flex items-center gap-1">
                    <Star className="h-4 w-4 fill-yellow-400 text-yellow-400"
/>
                    <span className="font-medium">
                      {averageRating.toFixed(1)}
                    </span>
                  </div>
                )}
              </div>

              {region && (
                <div className="flex items-center gap-1 text-xs text-muted-
foreground">
                  <MapPin className="h-3 w-3" />
                  <span>
                    {region.name}, {region.country}
                  </span>
                </div>
              )}
            </div>

            {/* Action Buttons */}
            <div className="flex gap-2">
              {canInteract ? (
                <>
                  <Button size="sm" className="flex-1 flex items-center gap-2">
                    <ShoppingCart className="h-4 w-4" />
                    Add to Cart
                  </Button>
                  <FavoriteToggleButton productId={product.id} />
                </>
              ) : (
                <Link href="/sign-in" className="w-full">
                  <Button
                    size="sm"
                    variant="outline"
                    className="w-full flex items-center gap-2"
                  >
                    <Lock className="h-4 w-4" />
                    Sign In to Purchase
                  </Button>
                </Link>
              )}
            </div>
          </CardContent>
        </Card>
      </Link>
    </article>
  );
}
```

Products Grid With Auth Features Explained:

- **Authentication Awareness**: Different behavior for signed-in/out users
- **Protected Features**: Some products require authentication
- **Visual Indicators**: Clear signs for authentication requirements
- **Conditional Actions**: Different buttons based on auth status
- **Accessibility**: Proper labeling for authentication states
- **User Experience**: Smooth transitions between auth states

---

These product components provide a comprehensive product management system for the Wine Store application. They handle:

- **Product Display**: Multiple view modes (grid, list)
- **Filtering & Search**: Advanced filtering capabilities
- **User Interaction**: Favorites, cart, and authentication
- **Performance**: Optimized rendering and state management
- **Accessibility**: Screen reader support and keyboard navigation
- **Responsiveness**: Mobile-first design approach