src/components/dashboard/Dashboard.tsx

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { StatsCards } from "./StatsCards";

import { InventoryChart } from "./InventoryChart";

import { RecentOrders } from "./RecentOrders";

import { LowStockAlerts } from "./LowStockAlerts";

export const Dashboard = () => {

return (

<div className="space-y-6">

<div>

<h1 className="text-3xl font-bold text-gray-900">Dashboard</h1>

<p className="text-gray-500 mt-2">Welcome back! Here's what's happening with your inventory.</p>

</div>

<StatsCards />

<div className="grid grid-cols-1 lg:grid-cols-2 gap-6">

<InventoryChart />

<LowStockAlerts />

</div>

<RecentOrders />

</div>

);

};

src/components/dashboard/InventoryChart.tsx

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip, ResponsiveContainer } from "recharts";

const data = [

{ month: "Jan", stock: 4000, orders: 2400 },

{ month: "Feb", stock: 3000, orders: 1398 },

{ month: "Mar", stock: 2000, orders: 9800 },

{ month: "Apr", stock: 2780, orders: 3908 },

{ month: "May", stock: 1890, orders: 4800 },

{ month: "Jun", stock: 2390, orders: 3800 },

];

export const InventoryChart = () => {

return (

<Card>

<CardHeader>

<CardTitle>Inventory Trends</CardTitle>

</CardHeader>

<CardContent>

<ResponsiveContainer width="100%" height={300}>

<LineChart data={data}>

<CartesianGrid strokeDasharray="3 3" />

<XAxis dataKey="month" />

<YAxis />

<Tooltip />

<Line

type="monotone"

dataKey="stock"

stroke="#3B82F6"

strokeWidth={2}

name="Stock Levels"

/>

<Line

type="monotone"

dataKey="orders"

stroke="#10B981"

strokeWidth={2}

name="Orders"

/>

</LineChart>

</ResponsiveContainer>

</CardContent>

</Card>

);

};

src/components/dashboard/LowStockAlerts.tsx

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { AlertTriangle } from "lucide-react";

import { Badge } from "@/components/ui/badge";

const lowStockItems = [

{ name: "Wireless Headphones", sku: "WH-001", current: 5, minimum: 10, category: "Electronics" },

{ name: "Office Chair", sku: "OC-025", current: 2, minimum: 5, category: "Furniture" },

{ name: "Coffee Beans", sku: "CB-100", current: 8, minimum: 15, category: "Food" },

{ name: "Laptop Stand", sku: "LS-030", current: 3, minimum: 8, category: "Accessories" },

];

export const LowStockAlerts = () => {

return (

<Card>

<CardHeader>

<CardTitle className="flex items-center gap-2">

<AlertTriangle className="w-5 h-5 text-orange-600" />

Low Stock Alerts

</CardTitle>

</CardHeader>

<CardContent>

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

{lowStockItems.map((item) => (

<div key={item.sku} className="flex items-center justify-between p-3 bg-orange-50 rounded-lg border border-orange-200">

<div>

<p className="font-medium text-gray-900">{item.name}</p>

<p className="text-sm text-gray-500">SKU: {item.sku}</p>

</div>

<div className="text-right">

<Badge variant="destructive" className="mb-1">

{item.current} left

</Badge>

<p className="text-xs text-gray-500">Min: {item.minimum}</p>

</div>

</div>

))}

</div>

</CardContent>

</Card>

);

};

src/components/dashboard/RecentOrders.tsx

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Badge } from "@/components/ui/badge";

import { Button } from "@/components/ui/button";

const recentOrders = [

{ id: "ORD-001", customer: "Acme Corp", total: "$1,250", status: "completed", date: "2024-01-15" },

{ id: "ORD-002", customer: "Tech Solutions", total: "$890", status: "pending", date: "2024-01-14" },

{ id: "ORD-003", customer: "Design Studio", total: "$2,100", status: "shipped", date: "2024-01-13" },

{ id: "ORD-004", customer: "Marketing Inc", total: "$750", status: "processing", date: "2024-01-12" },

];

const getStatusColor = (status: string) => {

switch (status) {

case "completed": return "bg-green-100 text-green-800";

case "pending": return "bg-yellow-100 text-yellow-800";

case "shipped": return "bg-blue-100 text-blue-800";

case "processing": return "bg-purple-100 text-purple-800";

default: return "bg-gray-100 text-gray-800";

}

};

export const RecentOrders = () => {

return (

<Card>

<CardHeader className="flex flex-row items-center justify-between">

<CardTitle>Recent Orders</CardTitle>

<Button variant="outline" size="sm">View All</Button>

</CardHeader>

<CardContent>

<div className="overflow-x-auto">

<table className="w-full">

<thead>

<tr className="border-b border-gray-200">

<th className="text-left py-3 px-4 font-medium text-gray-600">Order ID</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Customer</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Total</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Status</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Date</th>

</tr>

</thead>

<tbody>

{recentOrders.map((order) => (

<tr key={order.id} className="border-b border-gray-100 hover:bg-gray-50">

<td className="py-3 px-4 font-medium text-blue-600">{order.id}</td>

<td className="py-3 px-4">{order.customer}</td>

<td className="py-3 px-4 font-medium">{order.total}</td>

<td className="py-3 px-4">

<Badge className={getStatusColor(order.status)}>

{order.status}

</Badge>

</td>

<td className="py-3 px-4 text-gray-500">{order.date}</td>

</tr>

))}

</tbody>

</table>

</div>

</CardContent>

</Card>

);

};

src/components/dashboard/StatsCards.tsx

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Package, ShoppingCart, TrendingUp, AlertTriangle } from "lucide-react";

const stats = [

{

title: "Total Products",

value: "1,247",

change: "+12%",

icon: Package,

color: "text-blue-600",

bgColor: "bg-blue-50",

},

{

title: "Total Orders",

value: "89",

change: "+23%",

icon: ShoppingCart,

color: "text-green-600",

bgColor: "bg-green-50",

},

{

title: "Revenue",

value: "$45,678",

change: "+18%",

icon: TrendingUp,

color: "text-purple-600",

bgColor: "bg-purple-50",

},

{

title: "Low Stock Items",

value: "23",

change: "-5%",

icon: AlertTriangle,

color: "text-orange-600",

bgColor: "bg-orange-50",

},

];

export const StatsCards = () => {

return (

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

{stats.map((stat) => {

const Icon = stat.icon;

return (

<Card key={stat.title} className="hover:shadow-lg transition-shadow duration-200">

<CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">

<CardTitle className="text-sm font-medium text-gray-600">

{stat.title}

</CardTitle>

<div className={`p-2 rounded-lg ${stat.bgColor}`}>

<Icon className={`w-4 h-4 ${stat.color}`} />

</div>

</CardHeader>

<CardContent>

<div className="text-2xl font-bold text-gray-900">{stat.value}</div>

<p className="text-xs text-green-600 mt-1">

{stat.change} from last month

</p>

</CardContent>

</Card>

);

})}

</div>

);

};

src/components/layout/Header.tsx

import { Bell, Search, User } from "lucide-react";

import { Button } from "@/components/ui/button";

import { Input } from "@/components/ui/input";

export const Header = () => {

return (

<header className="bg-white border-b border-gray-200 px-6 py-4">

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

<div className="flex items-center gap-4">

<div className="relative">

<Search className="absolute left-3 top-1/2 transform -translate-y-1/2 text-gray-400 w-4 h-4" />

<Input

placeholder="Search products, orders..."

className="pl-10 w-80"

/>

</div>

</div>

<div className="flex items-center gap-4">

<Button variant="ghost" size="icon" className="relative">

<Bell className="w-5 h-5" />

<span className="absolute -top-1 -right-1 bg-red-500 text-white text-xs rounded-full w-5 h-5 flex items-center justify-center">

3

</span>

</Button>

<div className="flex items-center gap-3">

<div className="text-right">

<p className="text-sm font-medium text-gray-900">Admin User</p>

<p className="text-xs text-gray-500">Administrator</p>

</div>

<Button variant="ghost" size="icon" className="rounded-full">

<User className="w-5 h-5" />

</Button>

</div>

</div>

</div>

</header>

);

};

src/components/layout/InventoryLayout.tsx

import { ReactNode } from "react";

import { Sidebar } from "./Sidebar";

import { Header } from "./Header";

interface InventoryLayoutProps {

children: ReactNode;

currentView: string;

onViewChange: (view: string) => void;

}

export const InventoryLayout = ({ children, currentView, onViewChange }: InventoryLayoutProps) => {

return (

<div className="min-h-screen bg-gray-50 flex">

<Sidebar currentView={currentView} onViewChange={onViewChange} />

<div className="flex-1 flex flex-col">

<Header />

<main className="flex-1 p-6">

{children}

</main>

</div>

</div>

);

};

src/components/layout/Sidebar.tsx

import { cn } from "@/lib/utils";

import {

Home,

Package,

ShoppingCart,

BarChart3,

Users,

Settings,

Box

} from "lucide-react";

interface SidebarProps {

currentView: string;

onViewChange: (view: string) => void;

}

const menuItems = [

{ id: "dashboard", label: "Dashboard", icon: Home },

{ id: "products", label: "Products", icon: Package },

{ id: "orders", label: "Orders", icon: ShoppingCart },

{ id: "reports", label: "Reports", icon: BarChart3 },

{ id: "users", label: "Users", icon: Users },

{ id: "settings", label: "Settings", icon: Settings },

];

export const Sidebar = ({ currentView, onViewChange }: SidebarProps) => {

return (

<div className="w-64 bg-white shadow-lg border-r border-gray-200">

<div className="p-6 border-b border-gray-200">

<div className="flex items-center gap-3">

<div className="w-8 h-8 bg-blue-600 rounded-lg flex items-center justify-center">

<Box className="w-5 h-5 text-white" />

</div>

<div>

<h1 className="text-xl font-bold text-gray-900">InvenTrack</h1>

<p className="text-sm text-gray-500">Inventory System</p>

</div>

</div>

</div>

<nav className="p-4">

<ul className="space-y-2">

{menuItems.map((item) => {

const Icon = item.icon;

const isActive = currentView === item.id;

return (

<li key={item.id}>

<button

onClick={() => onViewChange(item.id)}

className={cn(

"w-full flex items-center gap-3 px-4 py-3 rounded-lg text-left transition-all duration-200",

isActive

? "bg-blue-50 text-blue-700 border border-blue-200"

: "text-gray-600 hover:bg-gray-50 hover:text-gray-900"

)}

>

<Icon className="w-5 h-5" />

<span className="font-medium">{item.label}</span>

</button>

</li>

);

})}

</ul>

</nav>

</div>

);

};

src/components/orders/OrderManagement.tsx

import { useState } from "react";

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Button } from "@/components/ui/button";

import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";

import { Plus } from "lucide-react";

import { PurchaseOrders } from "./PurchaseOrders";

import { SalesOrders } from "./SalesOrders";

export const OrderManagement = () => {

return (

<div className="space-y-6">

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

<div>

<h1 className="text-3xl font-bold text-gray-900">Order Management</h1>

<p className="text-gray-500 mt-2">Manage purchase and sales orders.</p>

</div>

<Button className="gap-2">

<Plus className="w-4 h-4" />

New Order

</Button>

</div>

<Tabs defaultValue="sales" className="space-y-6">

<TabsList className="grid w-full max-w-md grid-cols-2">

<TabsTrigger value="sales">Sales Orders</TabsTrigger>

<TabsTrigger value="purchase">Purchase Orders</TabsTrigger>

</TabsList>

<TabsContent value="sales">

<SalesOrders />

</TabsContent>

<TabsContent value="purchase">

<PurchaseOrders />

</TabsContent>

</Tabs>

</div>

);

};

src/components/orders/PurchaseOrders.tsx

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Badge } from "@/components/ui/badge";

import { Button } from "@/components/ui/button";

import { Eye, Edit, Check } from "lucide-react";

const purchaseOrders = [

{ id: "PO-001", supplier: "TechCorp", total: 5500.00, status: "approved", date: "2024-01-15", items: 12 },

{ id: "PO-002", supplier: "CompuTech", total: 3200.75, status: "pending", date: "2024-01-14", items: 8 },

{ id: "PO-003", supplier: "FurnishPro", total: 1890.50, status: "received", date: "2024-01-13", items: 6 },

{ id: "PO-004", supplier: "Bean Master", total: 450.25, status: "ordered", date: "2024-01-12", items: 4 },

];

const getStatusColor = (status: string) => {

switch (status) {

case "approved": return "bg-green-100 text-green-800";

case "pending": return "bg-yellow-100 text-yellow-800";

case "received": return "bg-blue-100 text-blue-800";

case "ordered": return "bg-purple-100 text-purple-800";

default: return "bg-gray-100 text-gray-800";

}

};

export const PurchaseOrders = () => {

return (

<Card>

<CardHeader>

<CardTitle>Purchase Orders</CardTitle>

</CardHeader>

<CardContent>

<div className="overflow-x-auto">

<table className="w-full">

<thead>

<tr className="border-b border-gray-200">

<th className="text-left py-3 px-4 font-medium text-gray-600">Order ID</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Supplier</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Items</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Total</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Status</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Date</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Actions</th>

</tr>

</thead>

<tbody>

{purchaseOrders.map((order) => (

<tr key={order.id} className="border-b border-gray-100 hover:bg-gray-50">

<td className="py-3 px-4 font-medium text-blue-600">{order.id}</td>

<td className="py-3 px-4">{order.supplier}</td>

<td className="py-3 px-4">{order.items}</td>

<td className="py-3 px-4 font-medium">${order.total.toFixed(2)}</td>

<td className="py-3 px-4">

<Badge className={getStatusColor(order.status)}>

{order.status}

</Badge>

</td>

<td className="py-3 px-4 text-gray-500">{order.date}</td>

<td className="py-3 px-4">

<div className="flex gap-2">

<Button variant="ghost" size="sm">

<Eye className="w-4 h-4" />

</Button>

<Button variant="ghost" size="sm">

<Edit className="w-4 h-4" />

</Button>

<Button variant="ghost" size="sm">

<Check className="w-4 h-4" />

</Button>

</div>

</td>

</tr>

))}

</tbody>

</table>

</div>

</CardContent>

</Card>

);

};

src/components/orders/SalesOrders.tsx

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Badge } from "@/components/ui/badge";

import { Button } from "@/components/ui/button";

import { Eye, Edit, Printer } from "lucide-react";

const salesOrders = [

{ id: "SO-001", customer: "Acme Corp", total: 1250.00, status: "completed", date: "2024-01-15", items: 5 },

{ id: "SO-002", customer: "Tech Solutions", total: 890.50, status: "pending", date: "2024-01-14", items: 3 },

{ id: "SO-003", customer: "Design Studio", total: 2100.00, status: "shipped", date: "2024-01-13", items: 8 },

{ id: "SO-004", customer: "Marketing Inc", total: 750.25, status: "processing", date: "2024-01-12", items: 2 },

];

const getStatusColor = (status: string) => {

switch (status) {

case "completed": return "bg-green-100 text-green-800";

case "pending": return "bg-yellow-100 text-yellow-800";

case "shipped": return "bg-blue-100 text-blue-800";

case "processing": return "bg-purple-100 text-purple-800";

default: return "bg-gray-100 text-gray-800";

}

};

export const SalesOrders = () => {

return (

<Card>

<CardHeader>

<CardTitle>Sales Orders</CardTitle>

</CardHeader>

<CardContent>

<div className="overflow-x-auto">

<table className="w-full">

<thead>

<tr className="border-b border-gray-200">

<th className="text-left py-3 px-4 font-medium text-gray-600">Order ID</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Customer</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Items</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Total</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Status</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Date</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Actions</th>

</tr>

</thead>

<tbody>

{salesOrders.map((order) => (

<tr key={order.id} className="border-b border-gray-100 hover:bg-gray-50">

<td className="py-3 px-4 font-medium text-blue-600">{order.id}</td>

<td className="py-3 px-4">{order.customer}</td>

<td className="py-3 px-4">{order.items}</td>

<td className="py-3 px-4 font-medium">${order.total.toFixed(2)}</td>

<td className="py-3 px-4">

<Badge className={getStatusColor(order.status)}>

{order.status}

</Badge>

</td>

<td className="py-3 px-4 text-gray-500">{order.date}</td>

<td className="py-3 px-4">

<div className="flex gap-2">

<Button variant="ghost" size="sm">

<Eye className="w-4 h-4" />

</Button>

<Button variant="ghost" size="sm">

<Edit className="w-4 h-4" />

</Button>

<Button variant="ghost" size="sm">

<Printer className="w-4 h-4" />

</Button>

</div>

</td>

</tr>

))}

</tbody>

</table>

</div>

</CardContent>

</Card>

);

};

src/components/products/AddProductDialog.tsx

import { Dialog, DialogContent, DialogHeader, DialogTitle } from "@/components/ui/dialog";

import { Button } from "@/components/ui/button";

import { Input } from "@/components/ui/input";

import { Label } from "@/components/ui/label";

import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";

interface AddProductDialogProps {

open: boolean;

onOpenChange: (open: boolean) => void;

}

export const AddProductDialog = ({ open, onOpenChange }: AddProductDialogProps) => {

return (

<Dialog open={open} onOpenChange={onOpenChange}>

<DialogContent className="sm:max-w-[600px]">

<DialogHeader>

<DialogTitle>Add New Product</DialogTitle>

</DialogHeader>

<div className="grid gap-4 py-4">

<div className="grid grid-cols-2 gap-4">

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

<Label htmlFor="sku">SKU</Label>

<Input id="sku" placeholder="Enter SKU" />

</div>

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

<Label htmlFor="name">Product Name</Label>

<Input id="name" placeholder="Enter product name" />

</div>

</div>

<div className="grid grid-cols-2 gap-4">

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

<Label htmlFor="category">Category</Label>

<Select>

<SelectTrigger>

<SelectValue placeholder="Select category" />

</SelectTrigger>

<SelectContent>

<SelectItem value="electronics">Electronics</SelectItem>

<SelectItem value="furniture">Furniture</SelectItem>

<SelectItem value="food">Food</SelectItem>

<SelectItem value="accessories">Accessories</SelectItem>

</SelectContent>

</Select>

</div>

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

<Label htmlFor="price">Price</Label>

<Input id="price" type="number" placeholder="0.00" />

</div>

</div>

<div className="grid grid-cols-2 gap-4">

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

<Label htmlFor="quantity">Initial Quantity</Label>

<Input id="quantity" type="number" placeholder="0" />

</div>

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

<Label htmlFor="supplier">Supplier</Label>

<Input id="supplier" placeholder="Enter supplier name" />

</div>

</div>

</div>

<div className="flex justify-end gap-2">

<Button variant="outline" onClick={() => onOpenChange(false)}>

Cancel

</Button>

<Button onClick={() => onOpenChange(false)}>

Add Product

</Button>

</div>

</DialogContent>

</Dialog>

);

};

src/components/products/ProductManagement.tsx

import { useState } from "react";

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Button } from "@/components/ui/button";

import { Input } from "@/components/ui/input";

import { Plus, Search } from "lucide-react";

import { ProductTable } from "./ProductTable";

import { AddProductDialog } from "./AddProductDialog";

export const ProductManagement = () => {

const [isAddDialogOpen, setIsAddDialogOpen] = useState(false);

const [searchTerm, setSearchTerm] = useState("");

return (

<div className="space-y-6">

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

<div>

<h1 className="text-3xl font-bold text-gray-900">Product Management</h1>

<p className="text-gray-500 mt-2">Manage your inventory products and stock levels.</p>

</div>

<Button onClick={() => setIsAddDialogOpen(true)} className="gap-2">

<Plus className="w-4 h-4" />

Add Product

</Button>

</div>

<Card>

<CardHeader>

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

<CardTitle>Products</CardTitle>

<div className="relative w-80">

<Search className="absolute left-3 top-1/2 transform -translate-y-1/2 text-gray-400 w-4 h-4" />

<Input

placeholder="Search products..."

value={searchTerm}

onChange={(e) => setSearchTerm(e.target.value)}

className="pl-10"

/>

</div>

</div>

</CardHeader>

<CardContent>

<ProductTable searchTerm={searchTerm} />

</CardContent>

</Card>

<AddProductDialog

open={isAddDialogOpen}

onOpenChange={setIsAddDialogOpen}

/>

</div>

);

};

src/components/products/ProductTable.tsx

import { Badge } from "@/components/ui/badge";

import { Button } from "@/components/ui/button";

import { Edit, Trash2 } from "lucide-react";

interface Product {

id: string;

sku: string;

name: string;

category: string;

price: number;

quantity: number;

supplier: string;

status: string;

}

const mockProducts: Product[] = [

{ id: "1", sku: "WH-001", name: "Wireless Headphones", category: "Electronics", price: 99.99, quantity: 5, supplier: "TechCorp", status: "low" },

{ id: "2", sku: "LT-100", name: "Gaming Laptop", category: "Electronics", price: 1299.99, quantity: 15, supplier: "CompuTech", status: "in-stock" },

{ id: "3", sku: "OC-025", name: "Office Chair", category: "Furniture", price: 249.99, quantity: 2, supplier: "FurnishPro", status: "low" },

{ id: "4", sku: "CB-100", name: "Coffee Beans", category: "Food", price: 19.99, quantity: 8, supplier: "Bean Master", status: "low" },

{ id: "5", sku: "TB-200", name: "Standing Desk", category: "Furniture", price: 399.99, quantity: 25, supplier: "WorkSpace", status: "in-stock" },

];

interface ProductTableProps {

searchTerm: string;

}

export const ProductTable = ({ searchTerm }: ProductTableProps) => {

const filteredProducts = mockProducts.filter(product =>

product.name.toLowerCase().includes(searchTerm.toLowerCase()) ||

product.sku.toLowerCase().includes(searchTerm.toLowerCase()) ||

product.category.toLowerCase().includes(searchTerm.toLowerCase())

);

const getStatusColor = (status: string) => {

switch (status) {

case "in-stock": return "bg-green-100 text-green-800";

case "low": return "bg-yellow-100 text-yellow-800";

case "out-of-stock": return "bg-red-100 text-red-800";

default: return "bg-gray-100 text-gray-800";

}

};

return (

<div className="overflow-x-auto">

<table className="w-full">

<thead>

<tr className="border-b border-gray-200">

<th className="text-left py-3 px-4 font-medium text-gray-600">SKU</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Product Name</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Category</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Price</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Quantity</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Supplier</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Status</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Actions</th>

</tr>

</thead>

<tbody>

{filteredProducts.map((product) => (

<tr key={product.id} className="border-b border-gray-100 hover:bg-gray-50">

<td className="py-3 px-4 font-medium text-blue-600">{product.sku}</td>

<td className="py-3 px-4 font-medium">{product.name}</td>

<td className="py-3 px-4">{product.category}</td>

<td className="py-3 px-4">${product.price}</td>

<td className="py-3 px-4">{product.quantity}</td>

<td className="py-3 px-4">{product.supplier}</td>

<td className="py-3 px-4">

<Badge className={getStatusColor(product.status)}>

{product.status.replace("-", " ")}

</Badge>

</td>

<td className="py-3 px-4">

<div className="flex gap-2">

<Button variant="ghost" size="sm">

<Edit className="w-4 h-4" />

</Button>

<Button variant="ghost" size="sm" className="text-red-600 hover:text-red-700">

<Trash2 className="w-4 h-4" />

</Button>

</div>

</td>

</tr>

))}

</tbody>

</table>

</div>

);

};

src/components/reports/ExpiryReport.tsx

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Badge } from "@/components/ui/badge";

import { AlertTriangle, Calendar } from "lucide-react";

const expiryData = [

{ name: "Coffee Beans Premium", sku: "CB-100", expiryDate: "2024-02-15", daysLeft: 12, category: "Food", quantity: 8, risk: "high" },

{ name: "Organic Tea Leaves", sku: "TL-050", expiryDate: "2024-02-28", daysLeft: 25, category: "Food", quantity: 15, risk: "medium" },

{ name: "Protein Powder", sku: "PP-200", expiryDate: "2024-03-10", daysLeft: 35, category: "Supplements", quantity: 22, risk: "medium" },

{ name: "Energy Bars", sku: "EB-075", expiryDate: "2024-04-01", daysLeft: 57, category: "Food", quantity: 45, risk: "low" },

{ name: "Vitamin C Tablets", sku: "VC-300", expiryDate: "2024-05-15", daysLeft: 101, category: "Supplements", quantity: 30, risk: "low" },

];

const getRiskColor = (risk: string) => {

switch (risk) {

case "high": return "bg-red-100 text-red-800";

case "medium": return "bg-yellow-100 text-yellow-800";

case "low": return "bg-green-100 text-green-800";

default: return "bg-gray-100 text-gray-800";

}

};

export const ExpiryReport = () => {

const criticalItems = expiryData.filter(item => item.risk === "high").length;

const warningItems = expiryData.filter(item => item.risk === "medium").length;

const totalValue = expiryData.reduce((acc, item) => acc + (item.quantity \* 15), 0); // Assuming average value

return (

<div className="space-y-6">

<div className="grid grid-cols-1 md:grid-cols-3 gap-6">

<Card>

<CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">

<CardTitle className="text-sm font-medium text-gray-600">Critical Items</CardTitle>

<AlertTriangle className="w-4 h-4 text-red-600" />

</CardHeader>

<CardContent>

<p className="text-2xl font-bold text-red-600">{criticalItems}</p>

<p className="text-sm text-gray-500">Expire within 15 days</p>

</CardContent>

</Card>

<Card>

<CardHeader className="flex flex-row items-center justify-between space-y-0 pb-2">

<CardTitle className="text-sm font-medium text-gray-600">Warning Items</CardTitle>

<Calendar className="w-4 h-4 text-yellow-600" />

</CardHeader>

<CardContent>

<p className="text-2xl font-bold text-yellow-600">{warningItems}</p>

<p className="text-sm text-gray-500">Expire within 30 days</p>

</CardContent>

</Card>

<Card>

<CardHeader>

<CardTitle className="text-sm font-medium text-gray-600">Total Value at Risk</CardTitle>

</CardHeader>

<CardContent>

<p className="text-2xl font-bold text-gray-900">${totalValue.toLocaleString()}</p>

<p className="text-sm text-gray-500">Potential loss if expired</p>

</CardContent>

</Card>

</div>

<Card>

<CardHeader>

<CardTitle>Expiry Timeline</CardTitle>

</CardHeader>

<CardContent>

<div className="overflow-x-auto">

<table className="w-full">

<thead>

<tr className="border-b border-gray-200">

<th className="text-left py-3 px-4 font-medium text-gray-600">Product</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">SKU</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Expiry Date</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Days Left</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Quantity</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Risk Level</th>

</tr>

</thead>

<tbody>

{expiryData.map((item) => (

<tr key={item.sku} className="border-b border-gray-100 hover:bg-gray-50">

<td className="py-3 px-4 font-medium">{item.name}</td>

<td className="py-3 px-4 text-blue-600">{item.sku}</td>

<td className="py-3 px-4">{item.expiryDate}</td>

<td className="py-3 px-4">{item.daysLeft} days</td>

<td className="py-3 px-4">{item.quantity}</td>

<td className="py-3 px-4">

<Badge className={getRiskColor(item.risk)}>

{item.risk} risk

</Badge>

</td>

</tr>

))}

</tbody>

</table>

</div>

</CardContent>

</Card>

</div>

);

};

src/components/reports/ProfitLossReport.tsx

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip, ResponsiveContainer, PieChart, Pie, Cell } from "recharts";

const profitData = [

{ month: "Jan", revenue: 45000, cost: 28000, profit: 17000 },

{ month: "Feb", revenue: 52000, cost: 32000, profit: 20000 },

{ month: "Mar", revenue: 48000, cost: 30000, profit: 18000 },

{ month: "Apr", revenue: 61000, cost: 38000, profit: 23000 },

{ month: "May", revenue: 55000, cost: 35000, profit: 20000 },

{ month: "Jun", revenue: 67000, cost: 42000, profit: 25000 },

];

const categoryData = [

{ name: "Electronics", value: 35, color: "#3B82F6" },

{ name: "Furniture", value: 28, color: "#10B981" },

{ name: "Food", value: 22, color: "#F59E0B" },

{ name: "Accessories", value: 15, color: "#EF4444" },

];

export const ProfitLossReport = () => {

return (

<div className="space-y-6">

<div className="grid grid-cols-1 md:grid-cols-3 gap-6">

<Card>

<CardHeader>

<CardTitle className="text-sm font-medium text-gray-600">Total Revenue</CardTitle>

</CardHeader>

<CardContent>

<p className="text-2xl font-bold text-green-600">$328,000</p>

<p className="text-sm text-gray-500">+12% from last period</p>

</CardContent>

</Card>

<Card>

<CardHeader>

<CardTitle className="text-sm font-medium text-gray-600">Total Costs</CardTitle>

</CardHeader>

<CardContent>

<p className="text-2xl font-bold text-red-600">$205,000</p>

<p className="text-sm text-gray-500">+8% from last period</p>

</CardContent>

</Card>

<Card>

<CardHeader>

<CardTitle className="text-sm font-medium text-gray-600">Net Profit</CardTitle>

</CardHeader>

<CardContent>

<p className="text-2xl font-bold text-blue-600">$123,000</p>

<p className="text-sm text-gray-500">+18% from last period</p>

</CardContent>

</Card>

</div>

<div className="grid grid-cols-1 lg:grid-cols-2 gap-6">

<Card>

<CardHeader>

<CardTitle>Profit Trend</CardTitle>

</CardHeader>

<CardContent>

<ResponsiveContainer width="100%" height={300}>

<LineChart data={profitData}>

<CartesianGrid strokeDasharray="3 3" />

<XAxis dataKey="month" />

<YAxis />

<Tooltip />

<Line type="monotone" dataKey="revenue" stroke="#10B981" strokeWidth={2} name="Revenue" />

<Line type="monotone" dataKey="cost" stroke="#EF4444" strokeWidth={2} name="Cost" />

<Line type="monotone" dataKey="profit" stroke="#3B82F6" strokeWidth={2} name="Profit" />

</LineChart>

</ResponsiveContainer>

</CardContent>

</Card>

<Card>

<CardHeader>

<CardTitle>Profit by Category</CardTitle>

</CardHeader>

<CardContent>

<ResponsiveContainer width="100%" height={300}>

<PieChart>

<Pie

data={categoryData}

cx="50%"

cy="50%"

outerRadius={80}

dataKey="value"

label={({ name, value }) => `${name}: ${value}%`}

>

{categoryData.map((entry, index) => (

<Cell key={`cell-${index}`} fill={entry.color} />

))}

</Pie>

<Tooltip />

</PieChart>

</ResponsiveContainer>

</CardContent>

</Card>

</div>

</div>

);

};

src/components/reports/Reports.tsx

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";

import { StockMovementReport } from "./StockMovementReport";

import { ProfitLossReport } from "./ProfitLossReport";

import { ExpiryReport } from "./ExpiryReport";

export const Reports = () => {

return (

<div className="space-y-6">

<div>

<h1 className="text-3xl font-bold text-gray-900">Reports & Analytics</h1>

<p className="text-gray-500 mt-2">Track performance and analyze your inventory data.</p>

</div>

<Tabs defaultValue="stock-movement" className="space-y-6">

<TabsList className="grid w-full max-w-2xl grid-cols-3">

<TabsTrigger value="stock-movement">Stock Movement</TabsTrigger>

<TabsTrigger value="profit-loss">Profit & Loss</TabsTrigger>

<TabsTrigger value="expiry">Expiry Report</TabsTrigger>

</TabsList>

<TabsContent value="stock-movement">

<StockMovementReport />

</TabsContent>

<TabsContent value="profit-loss">

<ProfitLossReport />

</TabsContent>

<TabsContent value="expiry">

<ExpiryReport />

</TabsContent>

</Tabs>

</div>

);

};

src/components/reports/StockMovementReport.tsx

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, ResponsiveContainer } from "recharts";

const stockMovementData = [

{ product: "Headphones", inbound: 50, outbound: 35, current: 15 },

{ product: "Laptops", inbound: 25, outbound: 18, current: 7 },

{ product: "Chairs", inbound: 30, outbound: 28, current: 2 },

{ product: "Coffee", inbound: 100, outbound: 92, current: 8 },

{ product: "Desks", inbound: 15, outbound: 12, current: 3 },

];

export const StockMovementReport = () => {

return (

<div className="space-y-6">

<Card>

<CardHeader>

<CardTitle>Stock Movement Overview</CardTitle>

</CardHeader>

<CardContent>

<ResponsiveContainer width="100%" height={400}>

<BarChart data={stockMovementData}>

<CartesianGrid strokeDasharray="3 3" />

<XAxis dataKey="product" />

<YAxis />

<Tooltip />

<Bar dataKey="inbound" fill="#3B82F6" name="Inbound" />

<Bar dataKey="outbound" fill="#EF4444" name="Outbound" />

<Bar dataKey="current" fill="#10B981" name="Current Stock" />

</BarChart>

</ResponsiveContainer>

</CardContent>

</Card>

<Card>

<CardHeader>

<CardTitle>Detailed Stock Movement</CardTitle>

</CardHeader>

<CardContent>

<div className="overflow-x-auto">

<table className="w-full">

<thead>

<tr className="border-b border-gray-200">

<th className="text-left py-3 px-4 font-medium text-gray-600">Product</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Inbound</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Outbound</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Current Stock</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Movement Rate</th>

</tr>

</thead>

<tbody>

{stockMovementData.map((item) => (

<tr key={item.product} className="border-b border-gray-100 hover:bg-gray-50">

<td className="py-3 px-4 font-medium">{item.product}</td>

<td className="py-3 px-4 text-green-600">+{item.inbound}</td>

<td className="py-3 px-4 text-red-600">-{item.outbound}</td>

<td className="py-3 px-4">{item.current}</td>

<td className="py-3 px-4">

{((item.outbound / item.inbound) \* 100).toFixed(1)}%

</td>

</tr>

))}

</tbody>

</table>

</div>

</CardContent>

</Card>

</div>

);

};

src/components/users/UserManagement.tsx

import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";

import { Button } from "@/components/ui/button";

import { Badge } from "@/components/ui/badge";

import { Plus, Edit, Trash2, Shield, User } from "lucide-react";

const users = [

{ id: "1", name: "John Admin", email: "john@company.com", role: "admin", status: "active", lastLogin: "2024-01-15" },

{ id: "2", name: "Sarah Staff", email: "sarah@company.com", role: "staff", status: "active", lastLogin: "2024-01-14" },

{ id: "3", name: "Mike Manager", email: "mike@company.com", role: "admin", status: "active", lastLogin: "2024-01-13" },

{ id: "4", name: "Lisa Worker", email: "lisa@company.com", role: "staff", status: "inactive", lastLogin: "2024-01-10" },

];

const getRoleColor = (role: string) => {

switch (role) {

case "admin": return "bg-purple-100 text-purple-800";

case "staff": return "bg-blue-100 text-blue-800";

default: return "bg-gray-100 text-gray-800";

}

};

const getStatusColor = (status: string) => {

switch (status) {

case "active": return "bg-green-100 text-green-800";

case "inactive": return "bg-red-100 text-red-800";

default: return "bg-gray-100 text-gray-800";

}

};

export const UserManagement = () => {

return (

<div className="space-y-6">

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

<div>

<h1 className="text-3xl font-bold text-gray-900">User Management</h1>

<p className="text-gray-500 mt-2">Manage user accounts and permissions.</p>

</div>

<Button className="gap-2">

<Plus className="w-4 h-4" />

Add User

</Button>

</div>

<div className="grid grid-cols-1 md:grid-cols-2 gap-6">

<Card>

<CardHeader>

<CardTitle className="flex items-center gap-2">

<Shield className="w-5 h-5 text-purple-600" />

Admin Permissions

</CardTitle>

</CardHeader>

<CardContent>

<ul className="space-y-2 text-sm">

<li className="flex items-center gap-2">

<div className="w-2 h-2 bg-green-500 rounded-full"></div>

Full product management access

</li>

<li className="flex items-center gap-2">

<div className="w-2 h-2 bg-green-500 rounded-full"></div>

Create and manage orders

</li>

<li className="flex items-center gap-2">

<div className="w-2 h-2 bg-green-500 rounded-full"></div>

Generate and view all reports

</li>

<li className="flex items-center gap-2">

<div className="w-2 h-2 bg-green-500 rounded-full"></div>

User management access

</li>

<li className="flex items-center gap-2">

<div className="w-2 h-2 bg-green-500 rounded-full"></div>

System settings configuration

</li>

</ul>

</CardContent>

</Card>

<Card>

<CardHeader>

<CardTitle className="flex items-center gap-2">

<User className="w-5 h-5 text-blue-600" />

Staff Permissions

</CardTitle>

</CardHeader>

<CardContent>

<ul className="space-y-2 text-sm">

<li className="flex items-center gap-2">

<div className="w-2 h-2 bg-green-500 rounded-full"></div>

View product information

</li>

<li className="flex items-center gap-2">

<div className="w-2 h-2 bg-green-500 rounded-full"></div>

Add new products only

</li>

<li className="flex items-center gap-2">

<div className="w-2 h-2 bg-yellow-500 rounded-full"></div>

Limited order management

</li>

<li className="flex items-center gap-2">

<div className="w-2 h-2 bg-red-500 rounded-full"></div>

No user management access

</li>

<li className="flex items-center gap-2">

<div className="w-2 h-2 bg-red-500 rounded-full"></div>

No system settings access

</li>

</ul>

</CardContent>

</Card>

</div>

<Card>

<CardHeader>

<CardTitle>User Accounts</CardTitle>

</CardHeader>

<CardContent>

<div className="overflow-x-auto">

<table className="w-full">

<thead>

<tr className="border-b border-gray-200">

<th className="text-left py-3 px-4 font-medium text-gray-600">Name</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Email</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Role</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Status</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Last Login</th>

<th className="text-left py-3 px-4 font-medium text-gray-600">Actions</th>

</tr>

</thead>

<tbody>

{users.map((user) => (

<tr key={user.id} className="border-b border-gray-100 hover:bg-gray-50">

<td className="py-3 px-4 font-medium">{user.name}</td>

<td className="py-3 px-4 text-gray-600">{user.email}</td>

<td className="py-3 px-4">

<Badge className={getRoleColor(user.role)}>

{user.role}

</Badge>

</td>

<td className="py-3 px-4">

<Badge className={getStatusColor(user.status)}>

{user.status}

</Badge>

</td>

<td className="py-3 px-4 text-gray-500">{user.lastLogin}</td>

<td className="py-3 px-4">

<div className="flex gap-2">

<Button variant="ghost" size="sm">

<Edit className="w-4 h-4" />

</Button>

<Button variant="ghost" size="sm" className="text-red-600 hover:text-red-700">

<Trash2 className="w-4 h-4" />

</Button>

</div>

</td>

</tr>

))}

</tbody>

</table>

</div>

</CardContent>

</Card>

</div>

);

};

src/pages/Index.tsx

// Update this page (the content is just a fallback if you fail to update the page)

const Index = () => {

return (

<div className="min-h-screen flex items-center justify-center bg-gray-100">

<div className="text-center">

<h1 className="text-4xl font-bold mb-4">Welcome to Your Blank App</h1>

<p className="text-xl text-gray-600">Start building your amazing project here!</p>

</div>

</div>

import { useState } from "react";

import { InventoryLayout } from "@/components/layout/InventoryLayout";

import { Dashboard } from "@/components/dashboard/Dashboard";

import { ProductManagement } from "@/components/products/ProductManagement";

import { OrderManagement } from "@/components/orders/OrderManagement";

import { Reports } from "@/components/reports/Reports";

import { UserManagement } from "@/components/users/UserManagement";

const Index = () => {

const [currentView, setCurrentView] = useState("dashboard");

const renderContent = () => {

switch (currentView) {

case "dashboard":

return <Dashboard />;

case "products":

return <ProductManagement />;

case "orders":

return <OrderManagement />;

case "reports":

return <Reports />;

case "users":

return <UserManagement />;

default:

return <Dashboard />;

}

};

return (

<InventoryLayout currentView={currentView} onViewChange={setCurrentView}>

{renderContent()}

</InventoryLayout>

);

};

export default Index;

src/integrations/supabase/client.ts

// This file is automatically generated. Do not edit it directly.

import { createClient } from '@supabase/supabase-js';

import type { Database } from './types';

const SUPABASE\_URL = "https://lbnyutjajobqqawvgdsu.supabase.co";

const SUPABASE\_PUBLISHABLE\_KEY = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6Imxibnl1dGpham9icXFhd3ZnZHN1Iiwicm9sZSI6ImFub24iLCJpYXQiOjE3NDgxNTk2NjcsImV4cCI6MjA2MzczNTY2N30.3rX4KavxAy6xlLF4MjhDg8qaEwJSF4CQ8M7rCxUQEWs";

// Import the supabase client like this:

// import { supabase } from "@/integrations/supabase/client";

export const supabase = createClient<Database>(SUPABASE\_URL, SUPABASE\_PUBLISHABLE\_KEY);

src/integrations/supabase/types.ts

export type Json =

| string

| number

| boolean

| null

| { [key: string]: Json | undefined }

| Json[]

export type Database = {

public: {

Tables: {

[\_ in never]: never

}

Views: {

[\_ in never]: never

}

Functions: {

[\_ in never]: never

}

Enums: {

[\_ in never]: never

}

CompositeTypes: {

[\_ in never]: never

}

}

}

type DefaultSchema = Database[Extract<keyof Database, "public">]

export type Tables<

DefaultSchemaTableNameOrOptions extends

| keyof (DefaultSchema["Tables"] & DefaultSchema["Views"])

| { schema: keyof Database },

TableName extends DefaultSchemaTableNameOrOptions extends {

schema: keyof Database

}

? keyof (Database[DefaultSchemaTableNameOrOptions["schema"]]["Tables"] &

Database[DefaultSchemaTableNameOrOptions["schema"]]["Views"])

: never = never,

> = DefaultSchemaTableNameOrOptions extends { schema: keyof Database }

? (Database[DefaultSchemaTableNameOrOptions["schema"]]["Tables"] &

Database[DefaultSchemaTableNameOrOptions["schema"]]["Views"])[TableName] extends {

Row: infer R

}

? R

: never

: DefaultSchemaTableNameOrOptions extends keyof (DefaultSchema["Tables"] &

DefaultSchema["Views"])

? (DefaultSchema["Tables"] &

DefaultSchema["Views"])[DefaultSchemaTableNameOrOptions] extends {

Row: infer R

}

? R

: never

: never

export type TablesInsert<

DefaultSchemaTableNameOrOptions extends

| keyof DefaultSchema["Tables"]

| { schema: keyof Database },

TableName extends DefaultSchemaTableNameOrOptions extends {

schema: keyof Database

}

? keyof Database[DefaultSchemaTableNameOrOptions["schema"]]["Tables"]

: never = never,

> = DefaultSchemaTableNameOrOptions extends { schema: keyof Database }

? Database[DefaultSchemaTableNameOrOptions["schema"]]["Tables"][TableName] extends {

Insert: infer I

}

? I

: never

: DefaultSchemaTableNameOrOptions extends keyof DefaultSchema["Tables"]

? DefaultSchema["Tables"][DefaultSchemaTableNameOrOptions] extends {

Insert: infer I

}

? I

: never

: never

export type TablesUpdate<

DefaultSchemaTableNameOrOptions extends

| keyof DefaultSchema["Tables"]

| { schema: keyof Database },

TableName extends DefaultSchemaTableNameOrOptions extends {

schema: keyof Database

}

? keyof Database[DefaultSchemaTableNameOrOptions["schema"]]["Tables"]

: never = never,

> = DefaultSchemaTableNameOrOptions extends { schema: keyof Database }

? Database[DefaultSchemaTableNameOrOptions["schema"]]["Tables"][TableName] extends {

Update: infer U

}

? U

: never

: DefaultSchemaTableNameOrOptions extends keyof DefaultSchema["Tables"]

? DefaultSchema["Tables"][DefaultSchemaTableNameOrOptions] extends {

Update: infer U

}

? U

: never

: never

export type Enums<

DefaultSchemaEnumNameOrOptions extends

| keyof DefaultSchema["Enums"]

| { schema: keyof Database },

EnumName extends DefaultSchemaEnumNameOrOptions extends {

schema: keyof Database

}

? keyof Database[DefaultSchemaEnumNameOrOptions["schema"]]["Enums"]

: never = never,

> = DefaultSchemaEnumNameOrOptions extends { schema: keyof Database }

? Database[DefaultSchemaEnumNameOrOptions["schema"]]["Enums"][EnumName]

: DefaultSchemaEnumNameOrOptions extends keyof DefaultSchema["Enums"]

? DefaultSchema["Enums"][DefaultSchemaEnumNameOrOptions]

: never

export type CompositeTypes<

PublicCompositeTypeNameOrOptions extends

| keyof DefaultSchema["CompositeTypes"]

| { schema: keyof Database },

CompositeTypeName extends PublicCompositeTypeNameOrOptions extends {

schema: keyof Database

}

? keyof Database[PublicCompositeTypeNameOrOptions["schema"]]["CompositeTypes"]

: never = never,

> = PublicCompositeTypeNameOrOptions extends { schema: keyof Database }

? Database[PublicCompositeTypeNameOrOptions["schema"]]["CompositeTypes"][CompositeTypeName]

: PublicCompositeTypeNameOrOptions extends keyof DefaultSchema["CompositeTypes"]

? DefaultSchema["CompositeTypes"][PublicCompositeTypeNameOrOptions]

: never

export const Constants = {

public: {

Enums: {},

},

} as const

supabase/config.toml

project\_id = "lbnyutjajobqqawvgdsu"

backend

-- Enable UUID extension

CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

-- Create enum types

CREATE TYPE user\_role AS ENUM ('admin', 'staff');

CREATE TYPE order\_status AS ENUM ('pending', 'approved', 'ordered', 'received', 'cancelled');

CREATE TYPE sales\_order\_status AS ENUM ('pending', 'processing', 'shipped', 'completed', 'cancelled');

-- Create categories table

CREATE TABLE categories (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

name TEXT NOT NULL UNIQUE,

description TEXT,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Create suppliers table

CREATE TABLE suppliers (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

name TEXT NOT NULL,

contact\_person TEXT,

email TEXT,

phone TEXT,

address TEXT,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Create user profiles table

CREATE TABLE profiles (

id UUID PRIMARY KEY REFERENCES auth.users(id) ON DELETE CASCADE,

full\_name TEXT,

role user\_role DEFAULT 'staff',

email TEXT,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Create products table

CREATE TABLE products (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

sku TEXT NOT NULL UNIQUE,

name TEXT NOT NULL,

description TEXT,

category\_id UUID REFERENCES categories(id),

supplier\_id UUID REFERENCES suppliers(id),

price DECIMAL(10,2) NOT NULL DEFAULT 0,

quantity INTEGER NOT NULL DEFAULT 0,

low\_stock\_threshold INTEGER DEFAULT 10,

barcode TEXT,

expiry\_date DATE,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Create purchase orders table

CREATE TABLE purchase\_orders (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

order\_number TEXT NOT NULL UNIQUE,

supplier\_id UUID REFERENCES suppliers(id) NOT NULL,

status order\_status DEFAULT 'pending',

order\_date DATE NOT NULL DEFAULT CURRENT\_DATE,

expected\_delivery\_date DATE,

total\_amount DECIMAL(10,2) DEFAULT 0,

notes TEXT,

created\_by UUID REFERENCES auth.users(id),

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Create purchase order items table

CREATE TABLE purchase\_order\_items (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

purchase\_order\_id UUID REFERENCES purchase\_orders(id) ON DELETE CASCADE,

product\_id UUID REFERENCES products(id),

quantity INTEGER NOT NULL,

unit\_price DECIMAL(10,2) NOT NULL,

total\_price DECIMAL(10,2) GENERATED ALWAYS AS (quantity \* unit\_price) STORED,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Create customers table

CREATE TABLE customers (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

name TEXT NOT NULL,

email TEXT,

phone TEXT,

address TEXT,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Create sales orders table

CREATE TABLE sales\_orders (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

order\_number TEXT NOT NULL UNIQUE,

customer\_id UUID REFERENCES customers(id) NOT NULL,

status sales\_order\_status DEFAULT 'pending',

order\_date DATE NOT NULL DEFAULT CURRENT\_DATE,

delivery\_date DATE,

total\_amount DECIMAL(10,2) DEFAULT 0,

notes TEXT,

created\_by UUID REFERENCES auth.users(id),

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Create sales order items table

CREATE TABLE sales\_order\_items (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

sales\_order\_id UUID REFERENCES sales\_orders(id) ON DELETE CASCADE,

product\_id UUID REFERENCES products(id),

quantity INTEGER NOT NULL,

unit\_price DECIMAL(10,2) NOT NULL,

total\_price DECIMAL(10,2) GENERATED ALWAYS AS (quantity \* unit\_price) STORED,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Create stock movements table for tracking inventory changes

CREATE TABLE stock\_movements (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

product\_id UUID REFERENCES products(id) NOT NULL,

movement\_type TEXT NOT NULL, -- 'in', 'out', 'adjustment'

quantity INTEGER NOT NULL,

reference\_type TEXT, -- 'purchase\_order', 'sales\_order', 'adjustment'

reference\_id UUID,

notes TEXT,

created\_by UUID REFERENCES auth.users(id),

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Create low stock alerts table

CREATE TABLE low\_stock\_alerts (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

product\_id UUID REFERENCES products(id) NOT NULL,

current\_quantity INTEGER NOT NULL,

threshold INTEGER NOT NULL,

is\_resolved BOOLEAN DEFAULT FALSE,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

resolved\_at TIMESTAMP WITH TIME ZONE

);

-- Create function to handle new user profile creation

CREATE OR REPLACE FUNCTION handle\_new\_user()

RETURNS TRIGGER AS $$

BEGIN

INSERT INTO profiles (id, full\_name, email, role)

VALUES (

NEW.id,

COALESCE(NEW.raw\_user\_meta\_data->>'full\_name', NEW.email),

NEW.email,

'staff'

);

RETURN NEW;

END;

$$ LANGUAGE plpgsql SECURITY DEFINER;

-- Create trigger for new user profile creation

CREATE TRIGGER on\_auth\_user\_created

AFTER INSERT ON auth.users

FOR EACH ROW EXECUTE FUNCTION handle\_new\_user();

-- Create function to update stock quantity when purchase order items are received

CREATE OR REPLACE FUNCTION update\_stock\_on\_purchase\_received()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.status = 'received' AND OLD.status != 'received' THEN

-- Update product quantities

UPDATE products

SET quantity = quantity + poi.quantity,

updated\_at = NOW()

FROM purchase\_order\_items poi

WHERE poi.purchase\_order\_id = NEW.id

AND products.id = poi.product\_id;

-- Create stock movement records

INSERT INTO stock\_movements (product\_id, movement\_type, quantity, reference\_type, reference\_id, created\_by)

SELECT

poi.product\_id,

'in',

poi.quantity,

'purchase\_order',

NEW.id,

NEW.created\_by

FROM purchase\_order\_items poi

WHERE poi.purchase\_order\_id = NEW.id;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Create trigger for purchase order status changes

CREATE TRIGGER on\_purchase\_order\_status\_change

AFTER UPDATE ON purchase\_orders

FOR EACH ROW EXECUTE FUNCTION update\_stock\_on\_purchase\_received();

-- Create function to update stock quantity when sales order is completed

CREATE OR REPLACE FUNCTION update\_stock\_on\_sales\_completed()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.status = 'completed' AND OLD.status != 'completed' THEN

-- Update product quantities

UPDATE products

SET quantity = quantity - soi.quantity,

updated\_at = NOW()

FROM sales\_order\_items soi

WHERE soi.sales\_order\_id = NEW.id

AND products.id = soi.product\_id;

-- Create stock movement records

INSERT INTO stock\_movements (product\_id, movement\_type, quantity, reference\_type, reference\_id, created\_by)

SELECT

soi.product\_id,

'out',

soi.quantity,

'sales\_order',

NEW.id,

NEW.created\_by

FROM sales\_order\_items soi

WHERE soi.sales\_order\_id = NEW.id;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Create trigger for sales order status changes

CREATE TRIGGER on\_sales\_order\_status\_change

AFTER UPDATE ON sales\_orders

FOR EACH ROW EXECUTE FUNCTION update\_stock\_on\_sales\_completed();

-- Create function to check for low stock and create alerts

CREATE OR REPLACE FUNCTION check\_low\_stock()

RETURNS TRIGGER AS $$

BEGIN

-- Check if stock falls below threshold

IF NEW.quantity <= NEW.low\_stock\_threshold THEN

-- Create alert if one doesn't exist already

INSERT INTO low\_stock\_alerts (product\_id, current\_quantity, threshold)

VALUES (NEW.id, NEW.quantity, NEW.low\_stock\_threshold)

ON CONFLICT DO NOTHING;

ELSE

-- Resolve existing alerts if stock is above threshold

UPDATE low\_stock\_alerts

SET is\_resolved = TRUE, resolved\_at = NOW()

WHERE product\_id = NEW.id AND is\_resolved = FALSE;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Create trigger for low stock alerts

CREATE TRIGGER on\_product\_quantity\_change

AFTER UPDATE OF quantity ON products

FOR EACH ROW EXECUTE FUNCTION check\_low\_stock();

-- Enable Row Level Security

ALTER TABLE categories ENABLE ROW LEVEL SECURITY;

ALTER TABLE suppliers ENABLE ROW LEVEL SECURITY;

ALTER TABLE profiles ENABLE ROW LEVEL SECURITY;

ALTER TABLE products ENABLE ROW LEVEL SECURITY;

ALTER TABLE purchase\_orders ENABLE ROW LEVEL SECURITY;

ALTER TABLE purchase\_order\_items ENABLE ROW LEVEL SECURITY;

ALTER TABLE customers ENABLE ROW LEVEL SECURITY;

ALTER TABLE sales\_orders ENABLE ROW LEVEL SECURITY;

ALTER TABLE sales\_order\_items ENABLE ROW LEVEL SECURITY;

ALTER TABLE stock\_movements ENABLE ROW LEVEL SECURITY;

ALTER TABLE low\_stock\_alerts ENABLE ROW LEVEL SECURITY;

-- Create helper function to check user role

CREATE OR REPLACE FUNCTION get\_user\_role(user\_id UUID)

RETURNS user\_role AS $$

BEGIN

RETURN (SELECT role FROM profiles WHERE id = user\_id);

END;

$$ LANGUAGE plpgsql SECURITY DEFINER;

-- RLS Policies for profiles

CREATE POLICY "Users can view all profiles" ON profiles FOR SELECT TO authenticated USING (true);

CREATE POLICY "Users can update own profile" ON profiles FOR UPDATE TO authenticated USING (auth.uid() = id);

CREATE POLICY "Admins can update any profile" ON profiles FOR UPDATE TO authenticated USING (get\_user\_role(auth.uid()) = 'admin');

-- RLS Policies for categories

CREATE POLICY "Categories are viewable by authenticated users" ON categories FOR SELECT TO authenticated USING (true);

CREATE POLICY "Admins can insert categories" ON categories FOR INSERT TO authenticated WITH CHECK (get\_user\_role(auth.uid()) = 'admin');

CREATE POLICY "Admins can update categories" ON categories FOR UPDATE TO authenticated USING (get\_user\_role(auth.uid()) = 'admin');

CREATE POLICY "Admins can delete categories" ON categories FOR DELETE TO authenticated USING (get\_user\_role(auth.uid()) = 'admin');

-- RLS Policies for suppliers

CREATE POLICY "Suppliers are viewable by authenticated users" ON suppliers FOR SELECT TO authenticated USING (true);

CREATE POLICY "Admins can insert suppliers" ON suppliers FOR INSERT TO authenticated WITH CHECK (get\_user\_role(auth.uid()) = 'admin');

CREATE POLICY "Admins can update suppliers" ON suppliers FOR UPDATE TO authenticated USING (get\_user\_role(auth.uid()) = 'admin');

CREATE POLICY "Admins can delete suppliers" ON suppliers FOR DELETE TO authenticated USING (get\_user\_role(auth.uid()) = 'admin');

-- RLS Policies for products

CREATE POLICY "Products are viewable by authenticated users" ON products FOR SELECT TO authenticated USING (true);

CREATE POLICY "Staff can insert products" ON products FOR INSERT TO authenticated WITH CHECK (true);

CREATE POLICY "Staff can update products" ON products FOR UPDATE TO authenticated USING (get\_user\_role(auth.uid()) IN ('admin', 'staff'));

CREATE POLICY "Admins can delete products" ON products FOR DELETE TO authenticated USING (get\_user\_role(auth.uid()) = 'admin');

-- RLS Policies for customers

CREATE POLICY "Customers are viewable by authenticated users" ON customers FOR SELECT TO authenticated USING (true);

CREATE POLICY "Staff can insert customers" ON customers FOR INSERT TO authenticated WITH CHECK (true);

CREATE POLICY "Staff can update customers" ON customers FOR UPDATE TO authenticated USING (true);

CREATE POLICY "Staff can delete customers" ON customers FOR DELETE TO authenticated USING (true);

-- RLS Policies for purchase orders

CREATE POLICY "Purchase orders are viewable by authenticated users" ON purchase\_orders FOR SELECT TO authenticated USING (true);

CREATE POLICY "Staff can create purchase orders" ON purchase\_orders FOR INSERT TO authenticated WITH CHECK (true);

CREATE POLICY "Staff can update purchase orders" ON purchase\_orders FOR UPDATE TO authenticated USING (get\_user\_role(auth.uid()) IN ('admin', 'staff'));

CREATE POLICY "Admins can delete purchase orders" ON purchase\_orders FOR DELETE TO authenticated USING (get\_user\_role(auth.uid()) = 'admin');

-- RLS Policies for purchase order items

CREATE POLICY "Purchase order items are viewable by authenticated users" ON purchase\_order\_items FOR SELECT TO authenticated USING (true);

CREATE POLICY "Staff can insert purchase order items" ON purchase\_order\_items FOR INSERT TO authenticated WITH CHECK (true);

CREATE POLICY "Staff can update purchase order items" ON purchase\_order\_items FOR UPDATE TO authenticated USING (true);

CREATE POLICY "Staff can delete purchase order items" ON purchase\_order\_items FOR DELETE TO authenticated USING (true);

-- RLS Policies for sales orders

CREATE POLICY "Sales orders are viewable by authenticated users" ON sales\_orders FOR SELECT TO authenticated USING (true);

CREATE POLICY "Staff can create sales orders" ON sales\_orders FOR INSERT TO authenticated WITH CHECK (true);

CREATE POLICY "Staff can update sales orders" ON sales\_orders FOR UPDATE TO authenticated USING (get\_user\_role(auth.uid()) IN ('admin', 'staff'));

CREATE POLICY "Admins can delete sales orders" ON sales\_orders FOR DELETE TO authenticated USING (get\_user\_role(auth.uid()) = 'admin');

-- RLS Policies for sales order items

CREATE POLICY "Sales order items are viewable by authenticated users" ON sales\_order\_items FOR SELECT TO authenticated USING (true);

CREATE POLICY "Staff can insert sales order items" ON sales\_order\_items FOR INSERT TO authenticated WITH CHECK (true);

CREATE POLICY "Staff can update sales order items" ON sales\_order\_items FOR UPDATE TO authenticated USING (true);

CREATE POLICY "Staff can delete sales order items" ON sales\_order\_items FOR DELETE TO authenticated USING (true);

-- RLS Policies for stock movements

CREATE POLICY "Stock movements are viewable by authenticated users" ON stock\_movements FOR SELECT TO authenticated USING (true);

CREATE POLICY "System can create stock movements" ON stock\_movements FOR INSERT TO authenticated WITH CHECK (true);

-- RLS Policies for low stock alerts

CREATE POLICY "Low stock alerts are viewable by authenticated users" ON low\_stock\_alerts FOR SELECT TO authenticated USING (true);

CREATE POLICY "System can insert low stock alerts" ON low\_stock\_alerts FOR INSERT TO authenticated WITH CHECK (true);

CREATE POLICY "System can update low stock alerts" ON low\_stock\_alerts FOR UPDATE TO authenticated USING (true);

CREATE POLICY "System can delete low stock alerts" ON low\_stock\_alerts FOR DELETE TO authenticated USING (true);

-- Insert some sample data

INSERT INTO categories (name, description) VALUES

('Electronics', 'Electronic devices and components'),

('Furniture', 'Office and home furniture'),

('Food', 'Food and beverage items'),

('Accessories', 'Various accessories and supplies');

INSERT INTO suppliers (name, contact\_person, email, phone, address) VALUES

('TechCorp', 'John Smith', 'john@techcorp.com', '+1-555-0101', '123 Tech Street, Silicon Valley'),

('FurnishPro', 'Jane Doe', 'jane@furnishpro.com', '+1-555-0102', '456 Furniture Ave, Design City'),

('Bean Master', 'Mike Coffee', 'mike@beanmaster.com', '+1-555-0103', '789 Coffee Lane, Brew Town'),

('CompuTech', 'Sarah Wilson', 'sarah@computech.com', '+1-555-0104', '321 Computer Blvd, Tech Park');

Authentication

src/components/auth/AuthPage.tsx

import { useState } from "react";

import { supabase } from "@/integrations/supabase/client";

import { Button } from "@/components/ui/button";

import { Input } from "@/components/ui/input";

import { Card, CardContent, CardDescription, CardHeader, CardTitle } from "@/components/ui/card";

import { Label } from "@/components/ui/label";

import { Alert, AlertDescription } from "@/components/ui/alert";

import { Loader2, Box } from "lucide-react";

import { useToast } from "@/hooks/use-toast";

export const AuthPage = () => {

const [isLogin, setIsLogin] = useState(true);

const [email, setEmail] = useState("");

const [password, setPassword] = useState("");

const [fullName, setFullName] = useState("");

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

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

const { toast } = useToast();

const handleAuth = async (e: React.FormEvent) => {

e.preventDefault();

setLoading(true);

setError("");

try {

if (isLogin) {

const { error } = await supabase.auth.signInWithPassword({

email,

password,

});

if (error) throw error;

toast({

title: "Welcome back!",

description: "You have successfully logged in.",

});

} else {

const { error } = await supabase.auth.signUp({

email,

password,

options: {

data: {

full\_name: fullName,

},

},

});

if (error) throw error;

toast({

title: "Account created!",

description: "Welcome to InvenTrack inventory system.",

});

}

} catch (error: any) {

setError(error.message);

toast({

title: "Authentication Error",

description: error.message,

variant: "destructive",

});

} finally {

setLoading(false);

}

};

return (

<div className="min-h-screen bg-gray-50 flex items-center justify-center p-4">

<Card className="w-full max-w-md">

<CardHeader className="text-center">

<div className="flex items-center justify-center gap-3 mb-4">

<div className="w-10 h-10 bg-blue-600 rounded-lg flex items-center justify-center">

<Box className="w-6 h-6 text-white" />

</div>

<div>

<h1 className="text-2xl font-bold text-gray-900">InvenTrack</h1>

<p className="text-sm text-gray-500">Inventory System</p>

</div>

</div>

<CardTitle>{isLogin ? "Welcome Back" : "Create Account"}</CardTitle>

<CardDescription>

{isLogin

? "Sign in to access your inventory dashboard"

: "Sign up to get started with inventory management"

}

</CardDescription>

</CardHeader>

<CardContent>

<form onSubmit={handleAuth} className="space-y-4">

{!isLogin && (

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

<Label htmlFor="fullName">Full Name</Label>

<Input

id="fullName"

type="text"

placeholder="Enter your full name"

value={fullName}

onChange={(e) => setFullName(e.target.value)}

required={!isLogin}

/>

</div>

)}

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

<Label htmlFor="email">Email</Label>

<Input

id="email"

type="email"

placeholder="Enter your email"

value={email}

onChange={(e) => setEmail(e.target.value)}

required

/>

</div>

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

<Label htmlFor="password">Password</Label>

<Input

id="password"

type="password"

placeholder="Enter your password"

value={password}

onChange={(e) => setPassword(e.target.value)}

required

minLength={6}

/>

</div>

{error && (

<Alert variant="destructive">

<AlertDescription>{error}</AlertDescription>

</Alert>

)}

<Button type="submit" className="w-full" disabled={loading}>

{loading && <Loader2 className="mr-2 h-4 w-4 animate-spin" />}

{isLogin ? "Sign In" : "Sign Up"}

</Button>

<div className="text-center">

<Button

type="button"

variant="link"

onClick={() => {

setIsLogin(!isLogin);

setError("");

setEmail("");

setPassword("");

setFullName("");

}}

className="text-sm"

>

{isLogin

? "Don't have an account? Sign up"

: "Already have an account? Sign in"

}

</Button>

</div>

</form>

</CardContent>

</Card>

</div>

);

};

src/components/auth/ProtectedRoute.tsx

import { ReactNode } from "react";

import { useAuth } from "@/hooks/useAuth";

import { AuthPage } from "./AuthPage";

import { Loader2 } from "lucide-react";

interface ProtectedRouteProps {

children: ReactNode;

}

export const ProtectedRoute = ({ children }: ProtectedRouteProps) => {

const { user, loading } = useAuth();

if (loading) {

return (

<div className="min-h-screen flex items-center justify-center">

<Loader2 className="h-8 w-8 animate-spin" />

</div>

);

}

if (!user) {

return <AuthPage />;

}

return <>{children}</>;

};

src/hooks/useAuth.tsx

import { createContext, useContext, useEffect, useState, ReactNode } from "react";

import { User, Session } from "@supabase/supabase-js";

import { supabase } from "@/integrations/supabase/client";

interface AuthContextType {

user: User | null;

session: Session | null;

loading: boolean;

signOut: () => Promise<void>;

}

const AuthContext = createContext<AuthContextType | undefined>(undefined);

export const useAuth = () => {

const context = useContext(AuthContext);

if (context === undefined) {

throw new Error("useAuth must be used within an AuthProvider");

}

return context;

};

interface AuthProviderProps {

children: ReactNode;

}

export const AuthProvider = ({ children }: AuthProviderProps) => {

const [user, setUser] = useState<User | null>(null);

const [session, setSession] = useState<Session | null>(null);

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

useEffect(() => {

// Set up auth state listener

const { data: { subscription } } = supabase.auth.onAuthStateChange(

(event, session) => {

console.log("Auth state changed:", event, session?.user?.email);

setSession(session);

setUser(session?.user ?? null);

setLoading(false);

}

);

// Get initial session

supabase.auth.getSession().then(({ data: { session } }) => {

console.log("Initial session:", session?.user?.email);

setSession(session);

setUser(session?.user ?? null);

setLoading(false);

});

return () => subscription.unsubscribe();

}, []);

const signOut = async () => {

const { error } = await supabase.auth.signOut();

if (error) {

console.error("Error signing out:", error);

}

};

const value = {

user,

session,

loading,

signOut,

};

return (

<AuthContext.Provider value={value}>

{children}

</AuthContext.Provider>

);

};

src/App.tsx

import { Toaster } from "@/components/ui/toaster";

import { Toaster as Sonner } from "@/components/ui/sonner";

import { TooltipProvider } from "@/components/ui/tooltip";

import { QueryClient, QueryClientProvider } from "@tanstack/react-query";

import { BrowserRouter, Routes, Route } from "react-router-dom";

import { AuthProvider } from "@/hooks/useAuth";

import { ProtectedRoute } from "@/components/auth/ProtectedRoute";

import Index from "./pages/Index";

import NotFound from "./pages/NotFound";

const queryClient = new QueryClient();

const App = () => (

<QueryClientProvider client={queryClient}>

<TooltipProvider>

<Toaster />

<Sonner />

<AuthProvider>

<BrowserRouter>

<Routes>

<Route

path="/"

element={

<ProtectedRoute>

<Index />

</ProtectedRoute>

}

/>

{/\* ADD ALL CUSTOM ROUTES ABOVE THE CATCH-ALL "\*" ROUTE \*/}

<Route path="\*" element={<NotFound />} />

</Routes>

</BrowserRouter>

</AuthProvider>

</TooltipProvider>

</QueryClientProvider>

);

export default App;

src/components/layout/Header.tsx

import { Bell, Search, User, LogOut } from "lucide-react";

import { Button } from "@/components/ui/button";

import { Input } from "@/components/ui/input";

import { useAuth } from "@/hooks/useAuth";

import {

DropdownMenu,

DropdownMenuContent,

DropdownMenuItem,

DropdownMenuLabel,

DropdownMenuSeparator,

DropdownMenuTrigger,

} from "@/components/ui/dropdown-menu";

export const Header = () => {

const { user, signOut } = useAuth();

return (

<header className="bg-white border-b border-gray-200 px-6 py-4">

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

<div className="flex items-center gap-4">

<div className="relative">

<Search className="absolute left-3 top-1/2 transform -translate-y-1/2 text-gray-400 w-4 h-4" />

<Input

placeholder="Search products, orders..."

className="pl-10 w-80"

/>

</div>

</div>

<div className="flex items-center gap-4">

<Button variant="ghost" size="icon" className="relative">

<Bell className="w-5 h-5" />

<span className="absolute -top-1 -right-1 bg-red-500 text-white text-xs rounded-full w-5 h-5 flex items-center justify-center">

3

</span>

</Button>

<DropdownMenu>

<DropdownMenuTrigger asChild>

<Button variant="ghost" className="flex items-center gap-3">

<div className="text-right">

<p className="text-sm font-medium text-gray-900">

{user?.user\_metadata?.full\_name || user?.email}

</p>

<p className="text-xs text-gray-500">Staff</p>

</div>

<div className="w-8 h-8 bg-blue-100 rounded-full flex items-center justify-center">

<User className="w-4 h-4 text-blue-600" />

</div>

</Button>

</DropdownMenuTrigger>

<DropdownMenuContent align="end" className="w-56">

<DropdownMenuLabel>My Account</DropdownMenuLabel>

<DropdownMenuSeparator />

<DropdownMenuItem onClick={signOut} className="text-red-600">

<LogOut className="mr-2 h-4 w-4" />

<span>Log out</span>

</DropdownMenuItem>

</DropdownMenuContent>

</DropdownMenu>

</div>

</div>

</header>

);

};

src/components/ui/alert.tsx

import \* as React from "react"

import { cva, type VariantProps } from "class-variance-authority"

import { cn } from "@/lib/utils"

const alertVariants = cva(

"relative w-full rounded-lg border p-4 [&>svg~\*]:pl-7 [&>svg+div]:translate-y-[-3px] [&>svg]:absolute [&>svg]:left-4 [&>svg]:top-4 [&>svg]:text-foreground",

{

variants: {

variant: {

default: "bg-background text-foreground",

destructive:

"border-destructive/50 text-destructive dark:border-destructive [&>svg]:text-destructive",

},

},

defaultVariants: {

variant: "default",

},

}

)

const Alert = React.forwardRef<

HTMLDivElement,

React.HTMLAttributes<HTMLDivElement> & VariantProps<typeof alertVariants>

>(({ className, variant, ...props }, ref) => (

<div

ref={ref}

role="alert"

className={cn(alertVariants({ variant }), className)}

{...props}

/>

))

Alert.displayName = "Alert"

const AlertTitle = React.forwardRef<

HTMLParagraphElement,

React.HTMLAttributes<HTMLHeadingElement>

>(({ className, ...props }, ref) => (

<h5

ref={ref}

className={cn("mb-1 font-medium leading-none tracking-tight", className)}

{...props}

/>

))

AlertTitle.displayName = "AlertTitle"

const AlertDescription = React.forwardRef<

HTMLParagraphElement,

React.HTMLAttributes<HTMLParagraphElement>

>(({ className, ...props }, ref) => (

<div

ref={ref}

className={cn("text-sm [&\_p]:leading-relaxed", className)}

{...props}

/>

))

AlertDescription.displayName = "AlertDescription"

export { Alert, AlertTitle, AlertDescription }

src/components/ui/card.tsx

import \* as React from "react"

import { cn } from "@/lib/utils"

const Card = React.forwardRef<

HTMLDivElement,

React.HTMLAttributes<HTMLDivElement>

>(({ className, ...props }, ref) => (

<div

ref={ref}

className={cn(

"rounded-lg border bg-card text-card-foreground shadow-sm",

className

)}

{...props}

/>

))

Card.displayName = "Card"

const CardHeader = React.forwardRef<

HTMLDivElement,

React.HTMLAttributes<HTMLDivElement>

>(({ className, ...props }, ref) => (

<div ref={ref} className={cn("flex flex-col space-y-1.5 p-6", className)} {...props} />

))

CardHeader.displayName = "CardHeader"

const CardTitle = React.forwardRef<

HTMLDivElement,

React.HTMLAttributes<HTMLDivElement>

>(({ className, ...props }, ref) => (

<div

ref={ref}

className={cn(

"text-2xl font-semibold leading-none tracking-tight",

className

)}

{...props}

/>

))

CardTitle.displayName = "CardTitle"

const CardDescription = React.forwardRef<

HTMLDivElement,

React.HTMLAttributes<HTMLDivElement>

>(({ className, ...props }, ref) => (

<div ref={ref} className={cn("text-sm text-muted-foreground", className)} {...props} />

))

CardDescription.displayName = "CardDescription"

const CardContent = React.forwardRef<

HTMLDivElement,

React.HTMLAttributes<HTMLDivElement>

>(({ className, ...props }, ref) => (

<div ref={ref} className={cn("p-6 pt-0", className)} {...props} />

))

CardContent.displayName = "CardContent"

const CardFooter = React.forwardRef<

HTMLDivElement,

React.HTMLAttributes<HTMLDivElement>

>(({ className, ...props }, ref) => (

<div ref={ref} className={cn("flex items-center p-6 pt-0", className)} {...props} />

))

CardFooter.displayName = "CardFooter"

export { Card, CardHeader, CardFooter, CardTitle, CardDescription, CardContent }

src/components/ui/dropdown-menu.tsx

import \* as React from "react"

import \* as DropdownMenuPrimitive from "@radix-ui/react-dropdown-menu"

import { Check, ChevronRight, Circle } from "lucide-react"

import { cn } from "@/lib/utils"

const DropdownMenu = DropdownMenuPrimitive.Root

const DropdownMenuTrigger = DropdownMenuPrimitive.Trigger

const DropdownMenuGroup = DropdownMenuPrimitive.Group

const DropdownMenuPortal = DropdownMenuPrimitive.Portal

const DropdownMenuSub = DropdownMenuPrimitive.Sub

const DropdownMenuRadioGroup = DropdownMenuPrimitive.RadioGroup

const DropdownMenuSubTrigger = React.forwardRef<

React.ElementRef<typeof DropdownMenuPrimitive.SubTrigger>,

React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.SubTrigger> & {

inset?: boolean

}

>(({ className, inset, children, ...props }, ref) => (

<DropdownMenuPrimitive.SubTrigger

ref={ref}

className={cn(

"flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none focus:bg-accent data-[state=open]:bg-accent",

inset && "pl-8",

className

)}

{...props}

>

{children}

<ChevronRight className="ml-auto h-4 w-4" />

</DropdownMenuPrimitive.SubTrigger>

))

DropdownMenuSubTrigger.displayName =

DropdownMenuPrimitive.SubTrigger.displayName

const DropdownMenuSubContent = React.forwardRef<

React.ElementRef<typeof DropdownMenuPrimitive.SubContent>,

React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.SubContent>

>(({ className, ...props }, ref) => (

<DropdownMenuPrimitive.SubContent

ref={ref}

className={cn(

"z-50 min-w-[8rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-foreground shadow-lg data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",

className

)}

{...props}

/>

))

DropdownMenuSubContent.displayName =

DropdownMenuPrimitive.SubContent.displayName

const DropdownMenuContent = React.forwardRef<

React.ElementRef<typeof DropdownMenuPrimitive.Content>,

React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Content>

>(({ className, sideOffset = 4, ...props }, ref) => (

<DropdownMenuPrimitive.Portal>

<DropdownMenuPrimitive.Content

ref={ref}

sideOffset={sideOffset}

className={cn(

"z-50 min-w-[8rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-foreground shadow-md data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",

className

)}

{...props}

/>

</DropdownMenuPrimitive.Portal>

))

DropdownMenuContent.displayName = DropdownMenuPrimitive.Content.displayName

const DropdownMenuItem = React.forwardRef<

React.ElementRef<typeof DropdownMenuPrimitive.Item>,

React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Item> & {

inset?: boolean

}

>(({ className, inset, ...props }, ref) => (

<DropdownMenuPrimitive.Item

ref={ref}

className={cn(

"relative flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none transition-colors focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",

inset && "pl-8",

className

)}

{...props}

/>

))

DropdownMenuItem.displayName = DropdownMenuPrimitive.Item.displayName

const DropdownMenuCheckboxItem = React.forwardRef<

React.ElementRef<typeof DropdownMenuPrimitive.CheckboxItem>,

React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.CheckboxItem>

>(({ className, children, checked, ...props }, ref) => (

<DropdownMenuPrimitive.CheckboxItem

ref={ref}

className={cn(

"relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none transition-colors focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",

className

)}

checked={checked}

{...props}

>

<span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">

<DropdownMenuPrimitive.ItemIndicator>

<Check className="h-4 w-4" />

</DropdownMenuPrimitive.ItemIndicator>

</span>

{children}

</DropdownMenuPrimitive.CheckboxItem>

))

DropdownMenuCheckboxItem.displayName =

DropdownMenuPrimitive.CheckboxItem.displayName

const DropdownMenuRadioItem = React.forwardRef<

React.ElementRef<typeof DropdownMenuPrimitive.RadioItem>,

React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.RadioItem>

>(({ className, children, ...props }, ref) => (

<DropdownMenuPrimitive.RadioItem

ref={ref}

className={cn(

"relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none transition-colors focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",

className

)}

{...props}

>

<span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">

<DropdownMenuPrimitive.ItemIndicator>

<Circle className="h-2 w-2 fill-current" />

</DropdownMenuPrimitive.ItemIndicator>

</span>

{children}

</DropdownMenuPrimitive.RadioItem>

))

DropdownMenuRadioItem.displayName = DropdownMenuPrimitive.RadioItem.displayName

const DropdownMenuLabel = React.forwardRef<

React.ElementRef<typeof DropdownMenuPrimitive.Label>,

React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Label> & {

inset?: boolean

}

>(({ className, inset, ...props }, ref) => (

<DropdownMenuPrimitive.Label

ref={ref}

className={cn(

"px-2 py-1.5 text-sm font-semibold",

inset && "pl-8",

className

)}

{...props}

/>

))

DropdownMenuLabel.displayName = DropdownMenuPrimitive.Label.displayName

const DropdownMenuSeparator = React.forwardRef<

React.ElementRef<typeof DropdownMenuPrimitive.Separator>,

React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Separator>

>(({ className, ...props }, ref) => (

<DropdownMenuPrimitive.Separator

ref={ref}

className={cn("-mx-1 my-1 h-px bg-muted", className)}

{...props}

/>

))

DropdownMenuSeparator.displayName = DropdownMenuPrimitive.Separator.displayName

const DropdownMenuShortcut = ({

className,

...props

}: React.HTMLAttributes<HTMLSpanElement>) => {

return (

<span

className={cn("ml-auto text-xs tracking-widest opacity-60", className)}

{...props}

/>

)

}

DropdownMenuShortcut.displayName = "DropdownMenuShortcut"

export {

DropdownMenu,

DropdownMenuTrigger,

DropdownMenuContent,

DropdownMenuItem,

DropdownMenuCheckboxItem,

DropdownMenuRadioItem,

DropdownMenuLabel,

DropdownMenuSeparator,

DropdownMenuShortcut,

DropdownMenuGroup,

DropdownMenuPortal,

DropdownMenuSub,

DropdownMenuSubContent,

DropdownMenuSubTrigger,

DropdownMenuRadioGroup,

}

src/integrations/supabase/types.ts

export type Json =

| string

| number

| boolean

| null

| { [key: string]: Json | undefined }

| Json[]

export type Database = {

public: {

Tables: {

categories: {

Row: {

created\_at: string | null

description: string | null

id: string

name: string

updated\_at: string | null

}

Insert: {

created\_at?: string | null

description?: string | null

id?: string

name: string

updated\_at?: string | null

}

Update: {

created\_at?: string | null

description?: string | null

id?: string

name?: string

updated\_at?: string | null

}

Relationships: []

}

customers: {

Row: {

address: string | null

created\_at: string | null

email: string | null

id: string

name: string

phone: string | null

updated\_at: string | null

}

Insert: {

address?: string | null

created\_at?: string | null

email?: string | null

id?: string

name: string

phone?: string | null

updated\_at?: string | null

}

Update: {

address?: string | null

created\_at?: string | null

email?: string | null

id?: string

name?: string

phone?: string | null

updated\_at?: string | null

}

Relationships: []

}

low\_stock\_alerts: {

Row: {

created\_at: string | null

current\_quantity: number

id: string

is\_resolved: boolean | null

product\_id: string

resolved\_at: string | null

threshold: number

}

Insert: {

created\_at?: string | null

current\_quantity: number

id?: string

is\_resolved?: boolean | null

product\_id: string

resolved\_at?: string | null

threshold: number

}

Update: {

created\_at?: string | null

current\_quantity?: number

id?: string

is\_resolved?: boolean | null

product\_id?: string

resolved\_at?: string | null

threshold?: number

}

Relationships: [

{

foreignKeyName: "low\_stock\_alerts\_product\_id\_fkey"

columns: ["product\_id"]

isOneToOne: false

referencedRelation: "products"

referencedColumns: ["id"]

},

]

}

products: {

Row: {

barcode: string | null

category\_id: string | null

created\_at: string | null

description: string | null

expiry\_date: string | null

id: string

low\_stock\_threshold: number | null

name: string

price: number

quantity: number

sku: string

supplier\_id: string | null

updated\_at: string | null

}

Insert: {

barcode?: string | null

category\_id?: string | null

created\_at?: string | null

description?: string | null

expiry\_date?: string | null

id?: string

low\_stock\_threshold?: number | null

name: string

price?: number

quantity?: number

sku: string

supplier\_id?: string | null

updated\_at?: string | null

}

Update: {

barcode?: string | null

category\_id?: string | null

created\_at?: string | null

description?: string | null

expiry\_date?: string | null

id?: string

low\_stock\_threshold?: number | null

name?: string

price?: number

quantity?: number

sku?: string

supplier\_id?: string | null

updated\_at?: string | null

}

Relationships: [

{

foreignKeyName: "products\_category\_id\_fkey"

columns: ["category\_id"]

isOneToOne: false

referencedRelation: "categories"

referencedColumns: ["id"]

},

{

foreignKeyName: "products\_supplier\_id\_fkey"

columns: ["supplier\_id"]

isOneToOne: false

referencedRelation: "suppliers"

referencedColumns: ["id"]

},

]

}

profiles: {

Row: {

created\_at: string | null

email: string | null

full\_name: string | null

id: string

role: Database["public"]["Enums"]["user\_role"] | null

updated\_at: string | null

}

Insert: {

created\_at?: string | null

email?: string | null

full\_name?: string | null

id: string

role?: Database["public"]["Enums"]["user\_role"] | null

updated\_at?: string | null

}

Update: {

created\_at?: string | null

email?: string | null

full\_name?: string | null

id?: string

role?: Database["public"]["Enums"]["user\_role"] | null

updated\_at?: string | null

}

Relationships: []

}

purchase\_order\_items: {

Row: {

created\_at: string | null

id: string

product\_id: string | null

purchase\_order\_id: string | null

quantity: number

total\_price: number | null

unit\_price: number

}

Insert: {

created\_at?: string | null

id?: string

product\_id?: string | null

purchase\_order\_id?: string | null

quantity: number

total\_price?: number | null

unit\_price: number

}

Update: {

created\_at?: string | null

id?: string

product\_id?: string | null

purchase\_order\_id?: string | null

quantity?: number

total\_price?: number | null

unit\_price?: number

}

Relationships: [

{

foreignKeyName: "purchase\_order\_items\_product\_id\_fkey"

columns: ["product\_id"]

isOneToOne: false

referencedRelation: "products"

referencedColumns: ["id"]

},

{

foreignKeyName: "purchase\_order\_items\_purchase\_order\_id\_fkey"

columns: ["purchase\_order\_id"]

isOneToOne: false

referencedRelation: "purchase\_orders"

referencedColumns: ["id"]

},

]

}

purchase\_orders: {

Row: {

created\_at: string | null

created\_by: string | null

expected\_delivery\_date: string | null

id: string

notes: string | null

order\_date: string

order\_number: string

status: Database["public"]["Enums"]["order\_status"] | null

supplier\_id: string

total\_amount: number | null

updated\_at: string | null

}

Insert: {

created\_at?: string | null

created\_by?: string | null

expected\_delivery\_date?: string | null

id?: string

notes?: string | null

order\_date?: string

order\_number: string

status?: Database["public"]["Enums"]["order\_status"] | null

supplier\_id: string

total\_amount?: number | null

updated\_at?: string | null

}

Update: {

created\_at?: string | null

created\_by?: string | null

expected\_delivery\_date?: string | null

id?: string

notes?: string | null

order\_date?: string

order\_number?: string

status?: Database["public"]["Enums"]["order\_status"] | null

supplier\_id?: string

total\_amount?: number | null

updated\_at?: string | null

}

Relationships: [

{

foreignKeyName: "purchase\_orders\_supplier\_id\_fkey"

columns: ["supplier\_id"]

isOneToOne: false

referencedRelation: "suppliers"

referencedColumns: ["id"]

},

]

}

sales\_order\_items: {

Row: {

created\_at: string | null

id: string

product\_id: string | null

quantity: number

sales\_order\_id: string | null

total\_price: number | null

unit\_price: number

}

Insert: {

created\_at?: string | null

id?: string

product\_id?: string | null

quantity: number

sales\_order\_id?: string | null

total\_price?: number | null

unit\_price: number

}

Update: {

created\_at?: string | null

id?: string

product\_id?: string | null

quantity?: number

sales\_order\_id?: string | null

total\_price?: number | null

unit\_price?: number

}

Relationships: [

{

foreignKeyName: "sales\_order\_items\_product\_id\_fkey"

columns: ["product\_id"]

isOneToOne: false

referencedRelation: "products"

referencedColumns: ["id"]

},

{

foreignKeyName: "sales\_order\_items\_sales\_order\_id\_fkey"

columns: ["sales\_order\_id"]

isOneToOne: false

referencedRelation: "sales\_orders"

referencedColumns: ["id"]

},

]

}

sales\_orders: {

Row: {

created\_at: string | null

created\_by: string | null

customer\_id: string

delivery\_date: string | null

id: string

notes: string | null

order\_date: string

order\_number: string

status: Database["public"]["Enums"]["sales\_order\_status"] | null

total\_amount: number | null

updated\_at: string | null

}

Insert: {

created\_at?: string | null

created\_by?: string | null

customer\_id: string

delivery\_date?: string | null

id?: string

notes?: string | null

order\_date?: string

order\_number: string

status?: Database["public"]["Enums"]["sales\_order\_status"] | null

total\_amount?: number | null

updated\_at?: string | null

}

Update: {

created\_at?: string | null

created\_by?: string | null

customer\_id?: string

delivery\_date?: string | null

id?: string

notes?: string | null

order\_date?: string

order\_number?: string

status?: Database["public"]["Enums"]["sales\_order\_status"] | null

total\_amount?: number | null

updated\_at?: string | null

}

Relationships: [

{

foreignKeyName: "sales\_orders\_customer\_id\_fkey"

columns: ["customer\_id"]

isOneToOne: false

referencedRelation: "customers"

referencedColumns: ["id"]

},

]

}

stock\_movements: {

Row: {

created\_at: string | null

created\_by: string | null

id: string

movement\_type: string

notes: string | null

product\_id: string

quantity: number

reference\_id: string | null

reference\_type: string | null

}

Insert: {

created\_at?: string | null

created\_by?: string | null

id?: string

movement\_type: string

notes?: string | null

product\_id: string

quantity: number

reference\_id?: string | null

reference\_type?: string | null

}

Update: {

created\_at?: string | null

created\_by?: string | null

id?: string

movement\_type?: string

notes?: string | null

product\_id?: string

quantity?: number

reference\_id?: string | null

reference\_type?: string | null

}

Relationships: [

{

foreignKeyName: "stock\_movements\_product\_id\_fkey"

columns: ["product\_id"]

isOneToOne: false

referencedRelation: "products"

referencedColumns: ["id"]

},

]

}

suppliers: {

Row: {

address: string | null

contact\_person: string | null

created\_at: string | null

email: string | null

id: string

name: string

phone: string | null

updated\_at: string | null

}

Insert: {

address?: string | null

contact\_person?: string | null

created\_at?: string | null

email?: string | null

id?: string

name: string

phone?: string | null

updated\_at?: string | null

}

Update: {

address?: string | null

contact\_person?: string | null

created\_at?: string | null

email?: string | null

id?: string

name?: string

phone?: string | null

updated\_at?: string | null

}

Relationships: []

}

}

Views: {

[\_ in never]: never

}

Functions: {

get\_user\_role: {

Args: { user\_id: string }

Returns: Database["public"]["Enums"]["user\_role"]

}

}

Enums: {

order\_status:

| "pending"

| "approved"

| "ordered"

| "received"

| "cancelled"

sales\_order\_status:

| "pending"

| "processing"

| "shipped"

| "completed"

| "cancelled"

user\_role: "admin" | "staff"

}

CompositeTypes: {

[\_ in never]: never

}

}

}

type DefaultSchema = Database[Extract<keyof Database, "public">]

export type Tables<

DefaultSchemaTableNameOrOptions extends

| keyof (DefaultSchema["Tables"] & DefaultSchema["Views"])

| { schema: keyof Database },

TableName extends DefaultSchemaTableNameOrOptions extends {

schema: keyof Database

}

? keyof (Database[DefaultSchemaTableNameOrOptions["schema"]]["Tables"] &

Database[DefaultSchemaTableNameOrOptions["schema"]]["Views"])

: never = never,

> = DefaultSchemaTableNameOrOptions extends { schema: keyof Database }

? (Database[DefaultSchemaTableNameOrOptions["schema"]]["Tables"] &

Database[DefaultSchemaTableNameOrOptions["schema"]]["Views"])[TableName] extends {

Row: infer R

}

? R

: never

: DefaultSchemaTableNameOrOptions extends keyof (DefaultSchema["Tables"] &

DefaultSchema["Views"])

? (DefaultSchema["Tables"] &

DefaultSchema["Views"])[DefaultSchemaTableNameOrOptions] extends {

Row: infer R

}

? R

: never

: never

export type TablesInsert<

DefaultSchemaTableNameOrOptions extends

| keyof DefaultSchema["Tables"]

| { schema: keyof Database },

TableName extends DefaultSchemaTableNameOrOptions extends {

schema: keyof Database

}

? keyof Database[DefaultSchemaTableNameOrOptions["schema"]]["Tables"]

: never = never,

> = DefaultSchemaTableNameOrOptions extends { schema: keyof Database }

? Database[DefaultSchemaTableNameOrOptions["schema"]]["Tables"][TableName] extends {

Insert: infer I

}

? I

: never

: DefaultSchemaTableNameOrOptions extends keyof DefaultSchema["Tables"]

? DefaultSchema["Tables"][DefaultSchemaTableNameOrOptions] extends {

Insert: infer I

}

? I

: never

: never

export type TablesUpdate<

DefaultSchemaTableNameOrOptions extends

| keyof DefaultSchema["Tables"]

| { schema: keyof Database },

TableName extends DefaultSchemaTableNameOrOptions extends {

schema: keyof Database

}

? keyof Database[DefaultSchemaTableNameOrOptions["schema"]]["Tables"]

: never = never,

> = DefaultSchemaTableNameOrOptions extends { schema: keyof Database }

? Database[DefaultSchemaTableNameOrOptions["schema"]]["Tables"][TableName] extends {

Update: infer U

}

? U

: never

: DefaultSchemaTableNameOrOptions extends keyof DefaultSchema["Tables"]

? DefaultSchema["Tables"][DefaultSchemaTableNameOrOptions] extends {

Update: infer U

}

? U

: never

: never

export type Enums<

DefaultSchemaEnumNameOrOptions extends

| keyof DefaultSchema["Enums"]

| { schema: keyof Database },

EnumName extends DefaultSchemaEnumNameOrOptions extends {

schema: keyof Database

}

? keyof Database[DefaultSchemaEnumNameOrOptions["schema"]]["Enums"]

: never = never,

> = DefaultSchemaEnumNameOrOptions extends { schema: keyof Database }

? Database[DefaultSchemaEnumNameOrOptions["schema"]]["Enums"][EnumName]

: DefaultSchemaEnumNameOrOptions extends keyof DefaultSchema["Enums"]

? DefaultSchema["Enums"][DefaultSchemaEnumNameOrOptions]

: never

export type CompositeTypes<

PublicCompositeTypeNameOrOptions extends

| keyof DefaultSchema["CompositeTypes"]

| { schema: keyof Database },

CompositeTypeName extends PublicCompositeTypeNameOrOptions extends {

schema: keyof Database

}

? keyof Database[PublicCompositeTypeNameOrOptions["schema"]]["CompositeTypes"]

: never = never,

> = PublicCompositeTypeNameOrOptions extends { schema: keyof Database }

? Database[PublicCompositeTypeNameOrOptions["schema"]]["CompositeTypes"][CompositeTypeName]

: PublicCompositeTypeNameOrOptions extends keyof DefaultSchema["CompositeTypes"]

? DefaultSchema["CompositeTypes"][PublicCompositeTypeNameOrOptions]

: never

export const Constants = {

public: {

Enums: {

order\_status: ["pending", "approved", "ordered", "received", "cancelled"],

sales\_order\_status: [

"pending",

"processing",

"shipped",

"completed",

"cancelled",

],

user\_role: ["admin", "staff"],

},

},

} as const