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

import { initializeApp } from 'firebase/app';

import { getAuth, signInWithCustomToken, signInAnonymously } from 'firebase/auth';

import { getFirestore, doc, getDoc, setDoc, onSnapshot, collection, query, where, addDoc } from 'firebase/firestore';

import { setLogLevel } from 'firebase/firestore';

// Set Firebase debug logging

setLogLevel('debug');

const firebaseConfig = typeof \_firebase\_config !== 'undefined' ? JSON.parse(\_firebase\_config) : {};

const appId = typeof \_\_app\_id !== 'undefined' ? \_\_app\_id : 'default-app-id';

const initialAuthToken = typeof \_\_initial\_auth\_token !== 'undefined' ? \_\_initial\_auth\_token : null;

// Icons

const ShoppingBagIcon = ({ className = 'w-6 h-6' }) => (

<svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" strokeWidth="2" strokeLinecap="round" strokeLinejoin="round" className={className}>

<path d="M6 2L3 6v14a2 2 0 0 0 2 2h14a2 2 0 0 0 2-2V6l-3-4z"></path>

<line x1="3" y1="6" x2="21" y2="6"></line>

<path d="M16 10a4 4 0 0 1-8 0"></path>

</svg>

);

const TrashIcon = ({ className = 'w-6 h-6' }) => (

<svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" strokeWidth="2" strokeLinecap="round" strokeLinejoin="round" className={className}>

<polyline points="3 6 5 6 21 6"></polyline>

<path d="M19 6v14a2 2 0 0 1-2 2H7a2 2 0 0 1-2-2V6m3 0V4a2 2 0 0 1 2-2h4a2 2 0 0 1 2 2v2"></path>

</svg>

);

const ChevronLeftIcon = ({ className = 'w-6 h-6' }) => (

<svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" strokeWidth="2" strokeLinecap="round" strokeLinejoin="round" className={className}>

<polyline points="15 18 9 12 15 6"></polyline>

</svg>

);

const App = () => {

const [db, setDb] = useState(null);

const [auth, setAuth] = useState(null);

const [userId, setUserId] = useState(null);

const [products, setProducts] = useState([]);

const [cart, setCart] = useState([]);

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

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

const [view, setView] = useState('products'); // 'products', 'productDetail', 'cart'

const [selectedProduct, setSelectedProduct] = useState(null);

const [showModal, setShowModal] = useState(false);

const [modalMessage, setModalMessage] = useState('');

// Handle modal display

const showMessageBox = (message) => {

setModalMessage(message);

setShowModal(true);

};

const closeModal = () => {

setShowModal(false);

setModalMessage('');

};

useEffect(() => {

try {

if (!firebaseConfig.apiKey) {

console.error("Firebase config is missing API key.");

setError("Firebase configuration is missing. Cannot initialize the app.");

return;

}

const app = initializeApp(firebaseConfig);

const firestore = getFirestore(app);

const firebaseAuth = getAuth(app);

setDb(firestore);

setAuth(firebaseAuth);

const unsubscribeAuth = firebaseAuth.onAuthStateChanged(async (user) => {

if (user) {

setUserId(user.uid);

} else {

try {

if (initialAuthToken) {

const userCredential = await signInWithCustomToken(firebaseAuth, initialAuthToken);

setUserId(userCredential.user.uid);

} else {

const userCredential = await signInAnonymously(firebaseAuth);

setUserId(userCredential.user.uid);

}

} catch (e) {

console.error("Authentication failed:", e);

setError("Authentication failed. Please check your network connection.");

}

}

});

return () => unsubscribeAuth();

} catch (e) {

console.error("Failed to initialize Firebase:", e);

setError("Failed to initialize Firebase. Please check your network connection.");

}

}, []);

// Firestore listeners for products and cart

useEffect(() => {

if (!db || !userId) return;

const productsCollectionRef = collection(db, 'artifacts', appId, 'public', 'data', 'products');

const cartDocRef = doc(db, 'artifacts', appId, 'users', userId, 'carts', 'user\_cart');

// Seed products if the collection is empty

const checkAndSeedProducts = async () => {

const productsSnapshot = await getDoc(doc(productsCollectionRef, 'seed\_check'));

if (!productsSnapshot.exists()) {

try {

const sampleProducts = [

{ id: '1', name: 'Wireless Headphones', price: 99.99, imageUrl: 'https://placehold.co/400x300/a855f7/ffffff?text=Headphones', description: 'Experience crystal-clear audio with these comfortable, noise-cancelling headphones.' },

{ id: '2', name: 'Smart Watch', price: 149.99, imageUrl: 'https://placehold.co/400x300/ec4899/ffffff?text=Smart+Watch', description: 'Track your fitness, receive notifications, and stay connected on the go with this stylish smart watch.' },

{ id: '3', name: 'Mechanical Keyboard', price: 75.00, imageUrl: 'https://placehold.co/400x300/3b82f6/ffffff?text=Keyboard', description: 'A durable and tactile keyboard designed for gamers and professionals. Features customizable RGB lighting.' },

{ id: '4', name: 'Portable Speaker', price: 55.00, imageUrl: 'https://placehold.co/400x300/22c55e/ffffff?text=Speaker', description: 'Compact and powerful, this portable speaker delivers rich sound with a long-lasting battery life.' },

{ id: '5', name: 'HD Webcam', price: 65.00, imageUrl: 'https://placehold.co/400x300/ef4444/ffffff?text=Webcam', description: 'Capture sharp, high-definition video for video calls and streaming with this easy-to-use webcam.' },

];

for (const product of sampleProducts) {

await setDoc(doc(productsCollectionRef, product.id), product);

}

await setDoc(doc(productsCollectionRef, 'seed\_check'), { seeded: true });

} catch (e) {

console.error("Error seeding products:", e);

}

}

};

checkAndSeedProducts();

const unsubscribeProducts = onSnapshot(productsCollectionRef, (snapshot) => {

const productsData = snapshot.docs.map(doc => ({ ...doc.data(), id: doc.id }));

setProducts(productsData);

setLoading(false);

}, (e) => {

console.error("Error getting products:", e);

setError("Failed to load products. Please refresh.");

setLoading(false);

});

const unsubscribeCart = onSnapshot(cartDocRef, (docSnap) => {

if (docSnap.exists()) {

const cartData = docSnap.data().items || [];

setCart(cartData);

} else {

setCart([]);

}

}, (e) => {

console.error("Error getting cart:", e);

setError("Failed to load cart data.");

});

return () => {

unsubscribeProducts();

unsubscribeCart();

};

}, [db, userId]);

const addToCart = async (product) => {

if (!db || !userId) {

showMessageBox("Error: Not authenticated. Please refresh the page.");

return;

}

const cartDocRef = doc(db, 'artifacts', appId, 'users', userId, 'carts', 'user\_cart');

const existingCartItem = cart.find(item => item.id === product.id);

let newCart;

if (existingCartItem) {

newCart = cart.map(item =>

item.id === product.id ? { ...item, quantity: item.quantity + 1 } : item

);

} else {

newCart = [...cart, { ...product, quantity: 1 }];

}

try {

await setDoc(cartDocRef, { items: newCart }, { merge: true });

showMessageBox(${product.name} added to cart!);

} catch (e) {

console.error("Error adding to cart:", e);

showMessageBox("Failed to add item to cart. Please try again.");

}

};

const removeFromCart = async (productId) => {

if (!db || !userId) {

showMessageBox("Error: Not authenticated. Please refresh the page.");

return;

}

const cartDocRef = doc(db, 'artifacts', appId, 'users', userId, 'carts', 'user\_cart');

const newCart = cart.filter(item => item.id !== productId);

try {

await setDoc(cartDocRef, { items: newCart }, { merge: true });

showMessageBox("Item removed from cart.");

} catch (e) {

console.error("Error removing from cart:", e);

showMessageBox("Failed to remove item from cart. Please try again.");

}

};

const getCartTotal = () => {

return cart.reduce((total, item) => total + (item.price \* item.quantity), 0).toFixed(2);

};

const renderView = () => {

switch (view) {

case 'cart':

return (

<div className="p-4 sm:p-8">

<h2 className="text-3xl font-bold mb-6 text-gray-800">Your Cart</h2>

{cart.length === 0 ? (

<p className="text-lg text-gray-600">Your cart is empty.</p>

) : (

<>

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

{cart.map(item => (

<li key={item.id} className="flex items-center justify-between p-4 bg-white rounded-lg shadow-md">

<div className="flex items-center space-x-4">

<img src={item.imageUrl} alt={item.name} className="w-16 h-16 object-cover rounded-lg" />

<div>

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

<p className="text-sm text-gray-600">Quantity: {item.quantity}</p>

<p className="text-sm text-gray-600">Price: ${item.price}</p>

</div>

</div>

<button onClick={() => removeFromCart(item.id)} className="p-2 text-red-500 hover:text-red-700 transition-colors duration-200">

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

</button>

</li>

))}

</ul>

<div className="mt-8 pt-4 border-t border-gray-200 flex justify-between items-center">

<p className="text-xl font-bold text-gray-800">Total: ${getCartTotal()}</p>

<button className="bg-green-500 text-white font-bold py-3 px-6 rounded-full shadow-lg hover:bg-green-600 transition-colors duration-200">

Checkout

</button>

</div>

</>

)}

</div>

);

case 'productDetail':

if (!selectedProduct) {

return null;

}

return (

<div className="p-4 sm:p-8">

<button onClick={() => setView('products')} className="flex items-center text-gray-600 mb-6 hover:text-gray-800 transition-colors duration-200">

<ChevronLeftIcon className="w-5 h-5 mr-2" /> Back to products

</button>

<div className="bg-white rounded-lg shadow-lg p-6 flex flex-col md:flex-row items-center md:items-start">

<img src={selectedProduct.imageUrl} alt={selectedProduct.name} className="w-full md:w-1/2 rounded-lg object-cover mb-6 md:mb-0 md:mr-6" />

<div className="flex-1">

<h2 className="text-4xl font-extrabold text-gray-900 mb-2">{selectedProduct.name}</h2>

<p className="text-2xl font-bold text-gray-700 mb-4">${selectedProduct.price}</p>

<p className="text-gray-600 text-lg mb-6">{selectedProduct.description}</p>

<button onClick={() => addToCart(selectedProduct)} className="w-full md:w-auto bg-purple-600 text-white font-bold py-3 px-8 rounded-full shadow-lg hover:bg-purple-700 transition-colors duration-200">

Add to Cart

</button>

</div>

</div>

</div>

);

case 'products':

default:

return (

<div className="p-4 sm:p-8 grid gap-6 grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4">

{products.map(product => (

<div key={product.id} className="bg-white rounded-xl shadow-lg overflow-hidden flex flex-col transition-transform transform hover:scale-105">

<img src={product.imageUrl} alt={product.name} className="w-full h-48 object-cover" />

<div className="p-5 flex flex-col justify-between flex-1">

<div>

<h3 className="text-lg font-bold text-gray-900 mb-1">{product.name}</h3>

<p className="text-gray-700 font-semibold mb-3">${product.price}</p>

</div>

<div className="flex flex-col sm:flex-row space-y-2 sm:space-y-0 sm:space-x-2">

<button

onClick={() => {

setSelectedProduct(product);

setView('productDetail');

}}

className="flex-1 bg-gray-200 text-gray-800 font-semibold py-2 px-4 rounded-full hover:bg-gray-300 transition-colors duration-200"

>

View

</button>

<button

onClick={() => addToCart(product)}

className="flex-1 bg-purple-600 text-white font-bold py-2 px-4 rounded-full hover:bg-purple-700 transition-colors duration-200"

>

Add to Cart

</button>

</div>

</div>

</div>

))}

</div>

);

}

};

if (loading) {

return (

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

<div className="text-center text-gray-600">

<p className="text-xl">Loading...</p>

</div>

</div>

);

}

if (error) {

return (

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

<div className="text-center text-red-500 p-8 bg-white rounded-lg shadow-md">

<p className="text-lg font-semibold">{error}</p>

</div>

</div>

);

}

return (

<div className="bg-gray-100 min-h-screen font-sans antialiased">

<style>{`

@import url('https://fonts.googleapis.com/css2?family=Inter:wght@400;600;700&display=swap');

body { font-family: 'Inter', sans-serif; }

`}</style>

<script src="https://cdn.tailwindcss.com"></script>

{/\* Modal for messages \*/}

{showModal && (

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

<div className="bg-white p-6 rounded-lg shadow-lg max-w-sm w-full text-center">

<h3 className="text-lg font-semibold mb-4">{modalMessage}</h3>

<button onClick={closeModal} className="bg-purple-600 text-white font-bold py-2 px-6 rounded-full hover:bg-purple-700 transition-colors duration-200">

OK

</button>

</div>

</div>

)}

{/\* Header \*/}

<header className="bg-white shadow-md sticky top-0 z-40">

<div className="container mx-auto px-4 py-4 flex justify-between items-center flex-wrap">

<h1 className="text-2xl font-extrabold text-gray-900 cursor-pointer" onClick={() => setView('products')}>

E-Commerce Store

</h1>

<div className="flex items-center space-x-4 mt-2 sm:mt-0">

{userId && (

<span className="bg-gray-200 text-gray-800 text-sm font-medium px-3 py-1 rounded-full hidden sm:block">

User: {userId}

</span>

)}

<button onClick={() => setView('cart')} className="relative p-2 rounded-full bg-gray-100 hover:bg-gray-200 transition-colors duration-200">

<ShoppingBagIcon className="text-gray-600" />

{cart.length > 0 && (

<span className="absolute top-0 right-0 inline-flex items-center justify-center px-2 py-1 text-xs font-bold leading-none text-red-100 transform translate-x-1/2 -translate-y-1/2 bg-red-600 rounded-full">{cart.length}</span>

)}

</button>

</div>

</div>

</header>

{/\* Main Content \*/}

<main className="container mx-auto">

{renderView()}

</main>

</div>

);

};

export default App;