package.json

{

"name": "ecommerce-cart-node",

"version": "1.0.0",

"description": "Simple e-commerce cart system (Node.js + Express + lowdb)",

"main": "server.js",

"scripts": {

"start": "node server.js",

"dev": "nodemon server.js"

},

"author": "",

"license": "MIT",

"dependencies": {

"cors": "^2.8.5",

"express": "^4.18.2",

"lowdb": "^6.0.1",

"nanoid": "^4.0.0"

},

"devDependencies": {

"nodemon": "^2.0.22"

}

}

---

server.js

const express = require('express');

const cors = require('cors');

const productsRouter = require('./routes/products');

const cartRouter = require('./routes/cart');

const app = express();

app.use(cors());

app.use(express.json());

// Simple middleware to extract user-id header for cart ownership (for demo)

app.use((req, res, next) => {

// In a real app you'd use authentication and extract user id from token.

// For this demo, clients should send header: x-user-id: some-id

req.userId = req.header('x-user-id') || 'guest';

next();

});

// Routes

app.use('/api/products', productsRouter);

app.use('/api/cart', cartRouter);

// Health check

app.get('/', (req, res) => res.send({ ok: true, msg: 'Ecommerce Cart API running' }));

const PORT = process.env.PORT || 4000;

app.listen(PORT, () => {

console.log(`Server listening on http://localhost:${PORT}`);

});

---

db.js

// lowdb setup (file-based JSON DB)

const { Low } = require('lowdb');

const { JSONFile } = require('lowdb/node');

const path = require('path');

const fs = require('fs');

const file = path.join(\_\_dirname, 'data', 'db.json');

// ensure data dir and file exist

const dataDir = path.dirname(file);

if (!fs.existsSync(dataDir)) fs.mkdirSync(dataDir, { recursive: true });

if (!fs.existsSync(file)) fs.writeFileSync(file, JSON.stringify({ products: [], carts: {} }, null, 2));

const adapter = new JSONFile(file);

const db = new Low(adapter);

async function init() {

await db.read();

db.data = db.data || { products: [], carts: {} };

// seed products if empty

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

db.data.products = [

{ id: 'p1', title: 'T-shirt', price: 299.0, stock: 50, description: 'Comfortable cotton T-shirt' },

{ id: 'p2', title: 'Sneakers', price: 2499.0, stock: 25, description: 'Stylish running shoes' },

{ id: 'p3', title: 'Coffee Mug', price: 199.0, stock: 100, description: 'Ceramic mug 350ml' }

];

await db.write();

}

}

init();

module.exports = db;

---

routes/products.js

const express = require('express');

const router = express.Router();

const db = require('../db');

// GET /api/products - list all products

router.get('/', async (req, res) => {

await db.read();

res.json(db.data.products || []);

});

// GET /api/products/:id - get product by id

router.get('/:id', async (req, res) => {

await db.read();

const p = (db.data.products || []).find(x => x.id === req.params.id);

if (!p) return res.status(404).json({ error: 'Product not found' });

res.json(p);

});

module.exports = router;

---

routes/cart.js

const express = require('express');

const { nanoid } = require('nanoid');

const router = express.Router();

const db = require('../db');

/\*

Cart structure stored in db.data.carts:

{

userId1: {

id: 'cart-xxx',

items: [

{ productId: 'p1', qty: 2, priceAt: 299.0, title: 'T-shirt' },

...

],

createdAt: '...',

updatedAt: '...'

},

userId2: { ... }

}

\*/

// helper

async function getCartForUser(userId) {

await db.read();

db.data.carts = db.data.carts || {};

if (!db.data.carts[userId]) {

db.data.carts[userId] = {

id: `cart\_${nanoid(8)}`,

items: [],

createdAt: new Date().toISOString(),

updatedAt: new Date().toISOString()

};

await db.write();

}

return db.data.carts[userId];

}

// GET /api/cart - get current user's cart

router.get('/', async (req, res) => {

const userId = req.userId;

const cart = await getCartForUser(userId);

res.json(cart);

});

// POST /api/cart/add - add product to cart { productId, qty }

router.post('/add', async (req, res) => {

const userId = req.userId;

const { productId, qty } = req.body;

const quantity = Math.max(1, parseInt(qty || 1, 10));

await db.read();

const product = (db.data.products || []).find(p => p.id === productId);

if (!product) return res.status(404).json({ error: 'Product not found' });

const cart = await getCartForUser(userId);

const existing = cart.items.find(i => i.productId === productId);

if (existing) {

existing.qty += quantity;

} else {

cart.items.push({

productId,

qty: quantity,

priceAt: product.price,

title: product.title

});

}

cart.updatedAt = new Date().toISOString();

await db.write();

res.json(cart);

});

// POST /api/cart/update - update qty { productId, qty }

router.post('/update', async (req, res) => {

const userId = req.userId;

const { productId, qty } = req.body;

const quantity = parseInt(qty, 10);

if (isNaN(quantity) || quantity < 0) return res.status(400).json({ error: 'Invalid qty' });

const cart = await getCartForUser(userId);

const itemIndex = cart.items.findIndex(i => i.productId === productId);

if (itemIndex === -1) return res.status(404).json({ error: 'Item not in cart' });

if (quantity === 0) {

cart.items.splice(itemIndex, 1);

} else {

cart.items[itemIndex].qty = quantity;

}

cart.updatedAt = new Date().toISOString();

await db.write();

res.json(cart);

});

// DELETE /api/cart/remove/:productId

router.delete('/remove/:productId', async (req, res) => {

const userId = req.userId;

const productId = req.params.productId;

const cart = await getCartForUser(userId);

const before = cart.items.length;

cart.items = cart.items.filter(i => i.productId !== productId);

cart.updatedAt = new Date().toISOString();

await db.write();

res.json({ removed: before - cart.items.length, cart });

});

// POST /api/cart/clear

router.post('/clear', async (req, res) => {

const userId = req.userId;

const cart = await getCartForUser(userId);

cart.items = [];

cart.updatedAt = new Date().toISOString();

await db.write();

res.json(cart);

});

// POST /api/cart/checkout - demo endpoint that just returns order summary and clears cart

router.post('/checkout', async (req, res) => {

const userId = req.userId;

const cart = await getCartForUser(userId);

if (!cart.items || cart.items.length === 0) {

return res.status(400).json({ error: 'Cart is empty' });

}

const total = cart.items.reduce((s, it) => s + (it.priceAt \* it.qty), 0);

// Dummy 'order' - in real app you'd create order, charge payment, adjust stock, etc.

const order = {

orderId: `order\_${nanoid(10)}`,

userId,

items: cart.items,

total,

createdAt: new Date().toISOString()

};

// Clear cart

cart.items = [];

cart.updatedAt = new Date().toISOString();

await db.write();

res.json({ success: true, order });

});

module.exports = router;