Sumkam Supermarket — Complete Full-Stack Starter

Project: Sumkam Supermarket (Njoro) **What you get here:** A ready-to-copy full system skeleton: `backend` (Express + Postgres + Stripe + M-Pesa), `web` (Next.js storefront), `mobile` (Expo React Native app), `docker-compose` for local dev, SQL seed, .env examples, and README with run instructions.

Repo layout

```
sumkam/
├─ backend/
│  ├─ src/
│  │  ├─ controllers/
│  │  │  ├─ stripe.js
│  │  │  └─ mpesa.js
│  │  ├─ routes/
│  │  │  ├─ products.js
│  │  │  └─ payments.js
│  │  ├─ db.js
│  │  ├─ index.js
│  │  └─ seed.sql
│  ├─ package.json
│  └─ .env.example
├─ web/
│  ├─ pages/
│  │  ├─ index.js
│  │  ├─ product/[id].js
│  │  └─ checkout.js
│  ├─ components/
│  │  ├─ ProductCard.js
│  │  └─ CartContext.js
│  ├─ package.json
│  └─ .env.local.example
├─ mobile/
│  ├─ App.js
│  ├─ screens/
│  │  ├─ Home.js
│  │  └─ Checkout.js
│  └─ package.json
├─ docker-compose.yml
```

```
├─ README.md
└─ LICENSE
```

---

1) Backend — Express + Postgres + Stripe + M-Pesa

`backend/package.json`

```
{
  "name":"sumkam-backend",
  "version":"1.0.0",
  "main":"src/index.js",
  "scripts":{
    "start":"node src/index.js",
    "dev":"nodemon src/index.js"
  },
  "dependencies":{
    "axios":"^1.4.0",
    "body-parser":"^1.20.2",
    "cors":"^2.8.5",
    "dotenv":"^16.0.3",
    "express":"^4.18.2",
    "pg":"^8.11.0",
    "stripe":"^11.20.0",
    "bcrypt":"^5.1.0",
    "jsonwebtoken":"^9.0.0",
    "moment":"^2.29.4"
  },
  "devDependencies":{
    "nodemon":"^2.0.22"
  }
}
```

`backend/src/db.js` **— simple Postgres client**

```
const { Pool } = require('pg');
require('dotenv').config();

const pool = new Pool({ connectionString: process.env.DATABASE_URL });
module.exports = { query: (text, params) => pool.query(text, params) };
```

backend/src/index.js

```
require('dotenv').config();
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');

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

const app = express();
app.use(cors());
app.use(bodyParser.json());

app.get('/health', (req, res) => res.json({ status: 'ok', time: new Date() }));
app.use('/api/products', productsRouter);
app.use('/api/payments', paymentsRouter);

const PORT = process.env.PORT || 4000;
app.listen(PORT, () => console.log(`Backend running on ${PORT}`));
```

backend/src/routes/products.js

```
const express = require('express');
const db = require('../db');
const router = express.Router();

router.get('/', async (req, res) => {
  const result = await db.query('SELECT * FROM products ORDER BY id');
  res.json({ products: result.rows });
});

router.get('/:id', async (req, res) => {
  const { id } = req.params;
  const result = await db.query('SELECT * FROM products WHERE id=$1', [id]);
  if (!result.rows.length) return res.status(404).json({ error: 'Not found' });
  res.json({ product: result.rows[0] });
});

module.exports = router;
```

`backend/src/routes/payments.js`

```
const express = require('express');
const stripeCtrl = require('../controllers/stripe');
const mpesaCtrl = require('../controllers/mpesa');
const router = express.Router();

router.post('/stripe/create-payment-intent', stripeCtrl.createPaymentIntent);
router.post('/stripe/webhook', stripeCtrl.webhookHandler);

router.post('/mpesa/stkpush', mpesaCtrl.stkPush);
router.post('/mpesa/callback', mpesaCtrl.callback);

module.exports = router;
```

`backend/src/controllers/stripe.js`

```
const Stripe = require('stripe');
const stripe = new Stripe(process.env.STRIPE_SECRET_KEY);

exports.createPaymentIntent = async (req, res) => {
  try {
    const { amount, currency='kes', orderId } = req.body;
    const unit = Math.round(Number(amount) * 100); // smallest unit
    const pi = await stripe.paymentIntents.create({ amount: unit, currency:
currency.toLowerCase(), metadata: { orderId } });
    res.json({ clientSecret: pi.client_secret });
  } catch (err) { console.error(err); res.status(500).json({ error:
err.message }); }
};

exports.webhookHandler = async (req, res) => {
  // optional: verify signature and update order in DB
  res.json({ received: true });
};
```

`backend/src/controllers/mpesa.js` **(Daraja STK push - sandbox)**

```
const axios = require('axios');
const moment = require('moment');

async function getAccessToken() {
  const url = `${process.env.MPESA_BASE_URL}/oauth/v1/generate?
grant_type=client_credentials`;
```

```
  const res = await axios.get(url, { auth: { username:
process.env.MPESA_CONSUMER_KEY, password:
process.env.MPESA_CONSUMER_SECRET } });
  return res.data.access_token;
}

exports.stkPush = async (req, res) => {
  try {
    const { phone, amount, orderId } = req.body; // phone 2547XXXXXXXX
    const token = await getAccessToken();

    const shortcode = process.env.MPESA_SHORTCODE;
    const passkey = process.env.MPESA_PASSKEY;
    const timestamp = moment().format('YYYYMMDDHHmmss');
    const password = Buffer.from(shortcode + passkey +
timestamp).toString('base64');

    const payload = {
      BusinessShortCode: shortcode,
      Password: password,
      Timestamp: timestamp,
      TransactionType: 'CustomerPayBillOnline',
      Amount: amount,
      PartyA: phone,
      PartyB: shortcode,
      PhoneNumber: phone,
      CallBackURL: `${process.env.BACKEND_URL}/api/payments/mpesa/callback`,
      AccountReference: orderId,
      TransactionDesc: `Order ${orderId}`
    };

    const resp = await axios.post(`${process.env.MPESA_BASE_URL}/mpesa/stkpush/
v1/processrequest`, payload, { headers: { Authorization: `Bearer ${token}` } });
    res.json(resp.data);
  } catch (err) { console.error(err.response?.data || err.message);
res.status(500).json({ error: err.response?.data || err.message }); }
};

exports.callback = async (req, res) => {
  // Safaricom will POST payment result here. Save to DB and respond 200.
  console.log('MPESA CALLBACK', JSON.stringify(req.body).slice(0,1000));
  res.status(200).send('OK');
};
```

`backend/.env.example`

```
PORT=4000
DATABASE_URL=postgres://postgres:postgres@db:5432/sumkam_db
JWT_SECRET=your_jwt_secret
STRIPE_SECRET_KEY=sk_test_xxx
MPESA_BASE_URL=https://sandbox.safaricom.co.ke
MPESA_CONSUMER_KEY=your_consumer_key
MPESA_CONSUMER_SECRET=your_consumer_secret
MPESA_SHORTCODE=174379
MPESA_PASSKEY=your_passkey
BACKEND_URL=http://localhost:4000
```

`backend/src/seed.sql` **— schema + seed (run to create tables)**

```
CREATE TABLE IF NOT EXISTS users ( id SERIAL PRIMARY KEY, full_name
VARCHAR(255), email VARCHAR(255) UNIQUE, phone VARCHAR(20), password_hash
VARCHAR(255), role VARCHAR(20) DEFAULT 'customer', created_at TIMESTAMP DEFAULT
now() );

CREATE TABLE IF NOT EXISTS categories ( id SERIAL PRIMARY KEY, name
VARCHAR(100) UNIQUE, slug VARCHAR(120) UNIQUE );

CREATE TABLE IF NOT EXISTS products ( id SERIAL PRIMARY KEY, name VARCHAR(255),
slug VARCHAR(255) UNIQUE, description TEXT, price NUMERIC(10,2), sku
VARCHAR(100), stock INTEGER DEFAULT 0, category_id INTEGER REFERENCES
categories(id) ON DELETE SET NULL, image_url TEXT, created_at TIMESTAMP DEFAULT
now() );

CREATE TABLE IF NOT EXISTS orders ( id SERIAL PRIMARY KEY, user_id INTEGER
REFERENCES users(id), total_amount NUMERIC(10,2), currency VARCHAR(10) DEFAULT
'KES', status VARCHAR(50) DEFAULT 'pending', payment_method VARCHAR(50),
metadata JSONB, created_at TIMESTAMP DEFAULT now() );

CREATE TABLE IF NOT EXISTS order_items ( id SERIAL PRIMARY KEY, order_id
INTEGER REFERENCES orders(id) ON DELETE CASCADE, product_id INTEGER REFERENCES
products(id), quantity INTEGER, unit_price NUMERIC(10,2) );

CREATE TABLE IF NOT EXISTS payments ( id SERIAL PRIMARY KEY, order_id INTEGER
REFERENCES orders(id), provider VARCHAR(50), provider_payment_id VARCHAR(255),
amount NUMERIC(10,2), status VARCHAR(50), raw_response JSONB, created_at
TIMESTAMP DEFAULT now() );

-- seed categories & products
INSERT INTO categories (name, slug) VALUES ('Groceries','groceries') ON
CONFLICT DO NOTHING;
```

```
INSERT INTO categories (name, slug) VALUES ('Beverages','beverages') ON
CONFLICT DO NOTHING;
INSERT INTO categories (name, slug) VALUES ('Household','household') ON
CONFLICT DO NOTHING;

INSERT INTO products (name, slug, description, price, sku, stock, category_id,
image_url) VALUES
('Sunrise Maize Flour 2kg','sunrise-maize-2kg','Maize flour - 2kg pack',180.
00,'SMF-2KG',50,1,'/images/sunrise-maize-2kg.jpg') ON CONFLICT DO NOTHING,
('Tusker Lager 500ml','tusker-500ml','Bottle 500ml',150.00,'TUS-500',120,2,'/
images/tusker-500ml.jpg') ON CONFLICT DO NOTHING,
('Omo Washing Powder 2kg','omo-2kg','Laundry powder',400.00,'OMO-2KG',80,3,'/
images/omo-2kg.jpg') ON CONFLICT DO NOTHING;

-- admin user placeholder (hash later)
INSERT INTO users (full_name, email, phone, password_hash, role) VALUES
('Sumkam
Admin','admin@sumkamsupermarket.co.ke','+254700000000','$2b$12$EXAMPLEHASH','admin')
ON CONFLICT DO NOTHING;
```

---

2) Frontend — Next.js Storefront

`web/package.json`

```
{
  "name":"sumkam-web",
  "version":"1.0.0",
  "scripts":{ "dev":"next dev -p 3000", "build":"next build", "start":"next
start -p 3000" },
  "dependencies":{ "next":"13.4.10","react":"18.2.0","react-
dom":"18.2.0","swr":"2.1.0","stripe-js":"1.62.0","@stripe/react-stripe-
js":"2.1.0" }
}
```

`web/pages/index.js` **(product list)**

```
import useSWR from 'swr';
import Link from 'next/link';
const fetcher = (url) => fetch(url).then(r => r.json());
export default function Home() {
  const { data } = useSWR(`${process.env.NEXT_PUBLIC_API_URL}/api/products`,
fetcher);
  if (!data) return <div>Loading...</div>;
```

```
  return (
    <div>
      <h1>Sumkam Supermarket — Njoro</h1>
      <div style={{display:'grid',gridTemplateColumns:'repeat(3,1fr)',gap:16}}>
        {data.products.map(p => (
          <div key={p.id} style={{border:'1px solid #eee',padding:12}}>
            <img src={p.image_url || '/placeholder.png'} alt=''
style={{width:'100%',height:150,objectFit:'cover'}}/>
            <h3>{p.name}</h3>
            <p>KES {p.price}</p>
            <Link href={`/product/${p.id}`}><a>View</a></Link>
          </div>
        ))}
      </div>
    </div>
  );
}
```

`web/pages/product/[id].js` **(single product)**

```
import { useRouter } from 'next/router';
import useSWR from 'swr';
const fetcher = (url) => fetch(url).then(r => r.json());
export default function Product() {
  const router = useRouter();
  const { id } = router.query;
  const { data } = useSWR(id ? `${process.env.NEXT_PUBLIC_API_URL}/api/products/
${id}` : null, fetcher);
  if (!data) return <div>Loading...</div>;
  const p = data.product;
  return (
    <div>
      <h2>{p.name}</h2>
      <img src={p.image_url || '/placeholder.png'} style={{width:300}}/>
      <p>{p.description}</p>
      <p>KES {p.price}</p>
      <button onClick={() => { if (typeof window !== 'undefined') { const cart
= JSON.parse(localStorage.getItem('cart')||'[]'); cart.push({id:p.id,
name:p.name, price:p.price}); localStorage.setItem('cart',
JSON.stringify(cart)); alert('Added to cart'); }}}>Add to cart</button>
    </div>
  );
}
```

`web/pages/checkout.js` **(simplified: choose Stripe or M-Pesa)**

```
import { useState } from 'react';
export default function Checkout(){
  const [phone,setPhone] = useState('2547');
  const cart = (typeof window!=='undefined') ?
JSON.parse(localStorage.getItem('cart')||'[]') : [];
  const total = cart.reduce((s,i)=>s + Number(i.price || 0),0);

  async function payWithMpesa(){
    const res = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/payments/
mpesa/stkpush`,{method:'POST',headers:{'Content-Type':'application/
json'},body:JSON.stringify({phone,amount:total,orderId:Date.now()})});
    const j = await res.json(); alert(JSON.stringify(j));
  }

  async function payWithCard(){
    const res = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/payments/
stripe/create-payment-intent`,{method:'POST',headers:{'Content-
Type':'application/
json'},body:JSON.stringify({amount:total,orderId:Date.now()})});
    const j = await res.json(); alert('Card flow not fully implemented in this
stub. clientSecret='+j.clientSecret);
  }

  return (
    <div>
      <h2>Checkout</h2>
      <div>Items: {cart.length} — Total: KES {total}</div>
      <div>
        <h3>Pay with M-Pesa</h3>
        <input value={phone} onChange={e=>setPhone(e.target.value)} />
        <button onClick={payWithMpesa}>Pay with M-Pesa (STK Push)</button>
      </div>
      <div>
        <h3>Pay with Card</h3>
        <button onClick={payWithCard}>Pay with Card (Stripe)</button>
      </div>
    </div>
  );
}
```

`web/.env.local.example`

```
NEXT_PUBLIC_API_URL=http://localhost:4000
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_test_xxx
```

---

3) Mobile — Expo React Native (calls backend endpoints)

`mobile/package.json`

```
{ "name":"sumkam-mobile","version":"1.0.0","main":"node_modules/expo/
AppEntry.js","scripts":{"start":"expo start"},"dependencies":
{"expo":"48.0.0","react":"18.2.0","react-native":"0.71.8","@react-navigation/
native":"^6.1.6","@react-navigation/stack":"^6.3.16","axios":"^1.4.0"}}
```

`mobile/App.js` **(minimal navigation)**

```
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';
import Home from './screens/Home';
import Checkout from './screens/Checkout';

const Stack = createStackNavigator();
export default function App(){
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="Home" component={Home} />
        <Stack.Screen name="Checkout" component={Checkout} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

`mobile/screens/Home.js` **(fetch products)**

```
import React, {useEffect, useState} from 'react';
import { View, Text, FlatList, TouchableOpacity } from 'react-native';
import axios from 'axios';

export default function Home({navigation}){
```

```
  const [products,setProducts] = useState([]);
  useEffect(()=>{ axios.get('http://10.0.2.2:4000/api/
products').then(r=>setProducts(r.data.products)).catch(e=>console.log(e)); },
[]);
  return (
    <View style={{flex:1,padding:16}}>
      <Text style={{fontSize:20}}>Sumkam Supermarket (Njoro)</Text>
      <FlatList data={products} keyExtractor={i=>String(i.id)}
renderItem={({item})=> (
        <TouchableOpacity onPress={()=> alert('Add to cart logic here')}
style={{padding:12,borderBottomWidth:1}}>
          <Text>{item.name}</Text>
          <Text>KES {item.price}</Text>
        </TouchableOpacity>
      )} />
    </View>
  );
}
```

`mobile/screens/Checkout.js` **(call mpesa)**

```
import React, {useState} from 'react';
import { View, Text, TextInput, Button } from 'react-native';

export default function Checkout(){
  const [phone,setPhone] = useState('2547');
  async function pay(){
    const res = await fetch('http://10.0.2.2:4000/api/payments/mpesa/stkpush',
{method:'POST',headers:{'Content-Type':'application/
json'},body:JSON.stringify({phone,amount:100,orderId:Date.now()})});
    const j = await res.json(); alert(JSON.stringify(j));
  }

  return (
    <View style={{padding:16}}>
      <Text>Checkout (mobile)</Text>
      <TextInput value={phone} onChangeText={setPhone} style={{borderWidth:
1,padding:8}}/>
      <Button title="Pay with M-Pesa" onPress={pay} />
    </View>
  );
}
```

---

4) Docker Compose for local dev

`docker-compose.yml`

```
version: '3.8'
services:
  db:
    image: postgres:15
    environment:
      POSTGRES_DB: sumkam_db
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
    ports:
      - '5432:5432'
    volumes:
      - db_data:/var/lib/postgresql/data
  backend:
    build: ./backend
    command: sh -c "npm install && npm run dev"
    volumes:
      - ./backend:/usr/src/app
    ports:
      - '4000:4000'
    environment:
      DATABASE_URL: postgres://postgres:postgres@db:5432/sumkam_db
      PORT: 4000
      MPESA_BASE_URL: https://sandbox.safaricom.co.ke
      BACKEND_URL: http://localhost:4000
    depends_on: [db]
volumes:
  db_data:
```

Note: backend build here uses default Node image; for production create a proper Dockerfile in `backend/`.

---

5) README (quickstart)

1. Install Docker & Docker Compose.
2. Clone repo and open terminal at repo root.
3. Run `docker-compose up`.
4. Exec into db and run seed: `docker exec -it <dbcontainer> psql -U postgres -d sumkam_db -f /usr/src/app/src/seed.sql` OR run seed from backend using `psql` on host.
5. Open `http://localhost:3000` for web (Next.js), `http://localhost:4000` for API.

6. Mobile: run `expo start` inside `mobile/` and use emulator; for API calls in Android emulator use `10.0.2.2` (as example). For iOS or physical device use host IP.

Admin account (seed) - Email: `admin@sumkamsupermarket.co.ke` - Password: `SumkamAdmin2025!` (replace / hash before production)

Env vars: fill `backend/.env` from `.env.example` and `web/.env.local`.

---

6) Security & production notes (short)

- Hash passwords with bcrypt before storing.
- Never commit `.env`.
- Use Stripe Checkout or Elements to avoid card data on your server.
- Use HTTPS (required for Daraja callbacks). Use a real domain and TLS.
- For M-Pesa production, register with Safaricom and get production credentials; sandbox credentials differ.
- Implement webhooks verification and update order/payment records transactionally.

---

7) Next steps I can produce now (pick any):

- Full **copy-paste complete backend files** (I can paste every file content here).
- Full **complete Next.js app** files (pages/components/styles).
- Full **Expo mobile app** files ready to paste.
- **Dockerfile** for backend and production readiness.
- Provide a ZIP of the project (I can produce file contents here to copy).

Tell me which artifact you want me to *paste fully right now* (e.g., "Paste entire backend src files", or "Paste full web pages and components", or "Give me Dockerfile and production deploy scripts").

Generated for you: Sumkam Supermarket — Njoro.