ShopClues Clone — Frontend + Backend

This repository is a simplified, functional clone of the ShopClues homepage and basic e-commerce flows (catalog, product detail, cart, checkout, auth). It's **not** a pixel-perfect copy of ShopClues; instead it provides a production-ready starting point you can extend.

Project structure

```
shopclues-clone/
├─ frontend/                 # React + Tailwind UI
│  ├─ package.json
│  ├─ tailwind.config.js
│  ├─ postcss.config.js
│  └─ src/
│     ├─ main.jsx
│     ├─ App.jsx
│     ├─ index.css
│     ├─ pages/
│     │  ├─ Home.jsx
│     │  ├─ Category.jsx
│     │  ├─ Product.jsx
│     │  ├─ Cart.jsx
│     │  └─ Checkout.jsx
│     ├─ components/
│     │  ├─ Header.jsx
│     │  ├─ Footer.jsx
│     │  ├─ ProductCard.jsx
│     │  └─ SearchBar.jsx
│     └─ api.js

├─ backend/                  # Node + Express + MongoDB (Mongoose)
│  ├─ package.json
│  ├─ server.js
│  ├─ .env.example
│  ├─ models/
│  │  ├─ Product.js
│  │  ├─ User.js
│  │  └─ Order.js
│  ├─ routes/
│  │  ├─ products.js
│  │  ├─ auth.js
│  │  └─ cart.js
│  └─ controllers/
│     ├─ productCtrl.js
│     └─ authCtrl.js
```

```
└─ README.md
```

How to use

1. Install and run backend:

```
cd backend
npm install
# copy .env.example -> .env and set MONGODB_URI and JWT_SECRET
node server.js
```

1. Install and run frontend:

```
cd frontend
npm install
npm run dev
```

Open http://localhost:5173 (Vite default) for frontend and backend at http://localhost:5000

Backend — `backend/server.js` (Node + Express)

```
// server.js
import express from 'express';
import cors from 'cors';
import mongoose from 'mongoose';
import dotenv from 'dotenv';
import productsRouter from './routes/products.js';
import authRouter from './routes/auth.js';
import cartRouter from './routes/cart.js';

dotenv.config();
const app = express();
app.use(cors());
app.use(express.json());

const PORT = process.env.PORT || 5000;

mongoose.connect(process.env.MONGODB_URI, { useNewUrlParser: true,
useUnifiedTopology: true })
  .then(() => console.log('MongoDB connected'))
  .catch(err => console.error(err));

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

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

app.listen(PORT, () => console.log(`Server listening on ${PORT}`));
```

backend/models/Product.js

```
import mongoose from 'mongoose';

const ProductSchema = new mongoose.Schema({
  title: String,
  description: String,
  price: Number,
  mrp: Number,
  category: String,
  images: [String],
  stock: { type: Number, default: 100 },
  rating: { type: Number, default: 4.0 }
});

export default mongoose.model('Product', ProductSchema);
```

backend/models/User.js

```
import mongoose from 'mongoose';

const UserSchema = new mongoose.Schema({
  name: String,
  email: { type: String, unique: true },
  passwordHash: String,
  createdAt: { type: Date, default: Date.now }
});

export default mongoose.model('User', UserSchema);
```

backend/routes/products.js

```
import express from 'express';
import Product from '../models/Product.js';
const router = express.Router();

// GET /api/products?q=&category=&limit=&offset=
router.get('/', async (req, res) => {
  const { q, category, limit = 20, offset = 0 } = req.query;
  const filter = {};
  if (q) filter.title = { $regex: q, $options: 'i' };
  if (category) filter.category = category;
  const products = await
Product.find(filter).skip(Number(offset)).limit(Number(limit));
```

```
  res.json(products);
});

// GET /api/products/:id
router.get('/:id', async (req, res) => {
  const p = await Product.findById(req.params.id);
  if (!p) return res.status(404).json({ message: 'Not found' });
  res.json(p);
});

// POST /api/products (admin seeding)
router.post('/', async (req, res) => {
  const p = new Product(req.body);
  await p.save();
  res.status(201).json(p);
});

export default router;
```

`backend/routes/auth.js` **(simple JWT auth)**

```
import express from 'express';
import bcrypt from 'bcryptjs';
import jwt from 'jsonwebtoken';
import User from '../models/User.js';

const router = express.Router();

router.post('/register', async (req, res) => {
  const { name, email, password } = req.body;
  const salt = await bcrypt.genSalt(10);
  const hash = await bcrypt.hash(password, salt);
  const user = new User({ name, email, passwordHash: hash });
  await user.save();
  res.json({ message: 'ok' });
});

router.post('/login', async (req, res) => {
  const { email, password } = req.body;
  const user = await User.findOne({ email });
  if (!user) return res.status(401).json({ message: 'Invalid' });
  const ok = await bcrypt.compare(password, user.passwordHash);
  if (!ok) return res.status(401).json({ message: 'Invalid' });
  const token = jwt.sign({ id: user._id, email: user.email },
process.env.JWT_SECRET, { expiresIn: '7d' });
  res.json({ token, name: user.name });
});

export default router;
```

`backend/routes/cart.js` **(stateless cart endpoints — normally you'd store cart per user)**

```
import express from 'express';
import Product from '../models/Product.js';
const router = express.Router();

// This example calculates totals for a client-submitted cart
router.post('/calculate', async (req, res) => {
  const { items } = req.body; // [{productId, qty}]
  const ids = items.map(i => i.productId);
  const products = await Product.find({ _id: { $in: ids } });
  const byId = {};
  products.forEach(p => (byId[p._id] = p));

  const result = items.map(it => ({
    product: byId[it.productId],
    qty: it.qty,
    lineTotal: (byId[it.productId].price || 0) * it.qty
  }));
  const subtotal = result.reduce((s, r) => s + r.lineTotal, 0);
  res.json({ items: result, subtotal });
});

export default router;
```

Frontend — React + Tailwind (Vite)

`frontend/package.json` should include Vite, React, axios, react-router-dom, tailwind.

`frontend/src/api.js`

```
import axios from 'axios';
const API = axios.create({ baseURL: import.meta.env.VITE_API_URL || 'http://
localhost:5000/api' });
export default API;
```

`frontend/src/main.jsx`

```
import React from 'react'
import { createRoot } from 'react-dom/client'
import { BrowserRouter, Routes, Route } from 'react-router-dom'
import App from './App'
import './index.css'

createRoot(document.getElementById('root')).render(
```

```
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
)
```

frontend/src/App.jsx

```
import React from 'react'
import { Routes, Route } from 'react-router-dom'
import Home from './pages/Home'
import Product from './pages/Product'
import Cart from './pages/Cart'
import Header from './components/Header'
import Footer from './components/Footer'

export default function App(){
  return (
    <div className="min-h-screen flex flex-col">
      <Header />
      <main className="flex-1 container mx-auto p-4">
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/product/:id" element={<Product />} />
          <Route path="/cart" element={<Cart />} />
        </Routes>
      </main>
      <Footer />
    </div>
  )
}
```

frontend/src/components/Header.jsx

```
import React from 'react'
import { Link } from 'react-router-dom'

export default function Header(){
  return (
    <header className="bg-white shadow">
      <div className="container mx-auto flex items-center gap-4 p-4">
        <Link to="/" className="text-2xl font-bold">ShopClues-Clone</Link>
        <div className="flex-1">
          <input className="w-full border rounded p-2" placeholder="Search
for products, brands and more" />
        </div>
        <nav className="flex gap-4 items-center">
          <Link to="/cart">Cart</Link>
```

```
          <a href="#">Sign In</a>
        </nav>
      </div>
    </header>
  )
}
```

frontend/src/pages/Home.jsx

```
import React, {useEffect, useState} from 'react'
import API from '../api'
import ProductCard from '../components/ProductCard'

export default function Home(){
  const [products,setProducts] = useState([])
  useEffect(()=>{ API.get('/
products').then(r=>setProducts(r.data)).catch(()=>{}); },[])
  return (
    <div>
      <section className="mb-6">
        <h2 className="text-xl font-semibold mb-2">Deals of the Day</h2>
        <div className="grid grid-cols-2 md:grid-cols-4 gap-4">
          {products.slice(0,8).map(p=> <ProductCard key={p._id} product={p} /
>)}
        </div>
      </section>

      <section>
        <h2 className="text-xl font-semibold mb-2">Recommended for you</h2>
        <div className="grid grid-cols-2 md:grid-cols-4 gap-4">
          {products.map(p=> <ProductCard key={p._id} product={p} />)}
        </div>
      </section>
    </div>
  )
}
```

frontend/src/components/ProductCard.jsx

```
import React from 'react'
import { Link } from 'react-router-dom'

export default function ProductCard({product}){
  return (
    <div className="border rounded p-3 bg-white">
      <Link to={`/product/${product._id}`}>
        <img src={product.images?.[0] || 'https://via.placeholder.com/200'}
alt={product.title} className="w-full h-40 object-contain" />
        <h3 className="text-sm mt-2 truncate">{product.title}</h3>
```

```
      </Link>
      <div className="mt-2 flex items-center justify-between">
        <div>
          <div className="font-bold">₹{product.price}</div>
          <div className="text-xs line-through text-gray-400">₹{product.mrp}
</div>
        </div>
        <button className="px-3 py-1 bg-blue-600 text-white rounded">Add</
button>
      </div>
    </div>
  )
}
```

frontend/src/pages/Product.jsx

```
import React, {useEffect,useState} from 'react'
import { useParams } from 'react-router-dom'
import API from '../api'

export default function Product(){
  const { id } = useParams();
  const [product,setProduct] = useState(null)
  useEffect(()=>{ if(id) API.get(`/products/${id}
`).then(r=>setProduct(r.data)) },[id])
  if(!product) return <div>Loading...</div>
  return (
    <div className="grid md:grid-cols-3 gap-6">
      <div className="col-span-2">
        <img src={product.images?.[0]||'https://via.placeholder.com/600'}
className="w-full h-96 object-contain" />
      </div>
      <div>
        <h1 className="text-2xl font-semibold">{product.title}</h1>
        <p className="mt-2">{product.description}</p>
        <div className="mt-4">
          <div className="text-2xl font-bold">₹{product.price}</div>
          <div className="text-sm line-through">₹{product.mrp}</div>
        </div>
        <button className="mt-6 px-4 py-2 bg-yellow-500 rounded">Add to
cart</button>
      </div>
    </div>
  )
}
```

`frontend/src/pages/Cart.jsx`

```
import React from 'react'

export default function Cart(){
  return (
    <div>
      <h1 className="text-2xl mb-4">Your Cart</h1>
      <div className="p-4 bg-white border">Cart logic goes here — this repo
includes a stateless `cart/calculate` endpoint to compute totals from
items.</div>
    </div>
  )
}
```

Notes, extensions & deployment

- This clone intentionally keeps business logic minimal but shows a realistic separation of concerns: React frontend, REST backend, MongoDB for persistence.
- To extend:
- Add admin panel to seed products.
- Add payments integration (Razorpay/Stripe) in `checkout` flow.
- Add product filtering, sorting, pagination, caching.
- Harden auth (refresh tokens), rate limiting, input validation.

If you'd like, I can: - Provide full `package.json`, `tailwind.config.js`, and other exact files inside this project. - Generate seed data (a JSON file) for products to quickly populate the database. - Convert the backend to use SQLite + Prisma if you prefer a file-based DB.

Tell me which of the above you'd like next and I will add it directly into this project file.