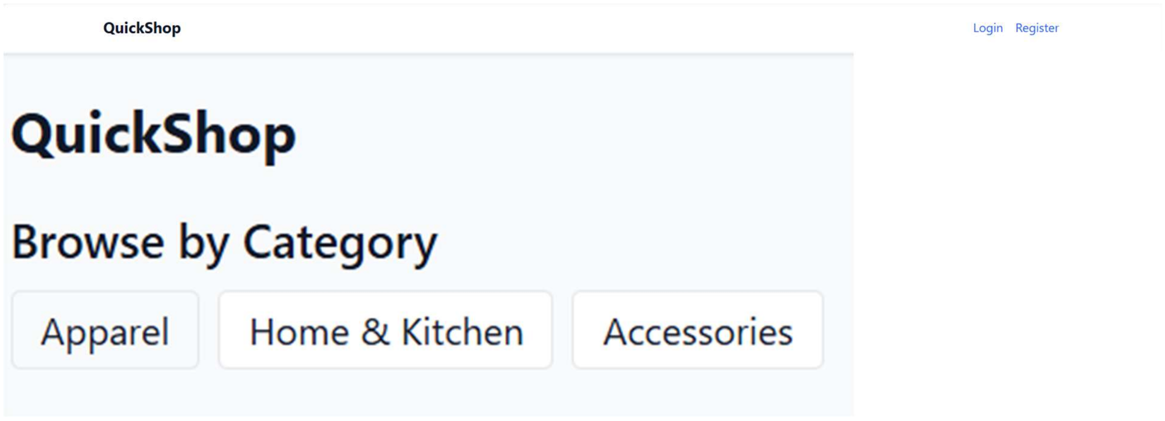
QuickShop — A Lightweight E-commerce Storefront
Submission Document for Site Guru Pvt. Ltd.

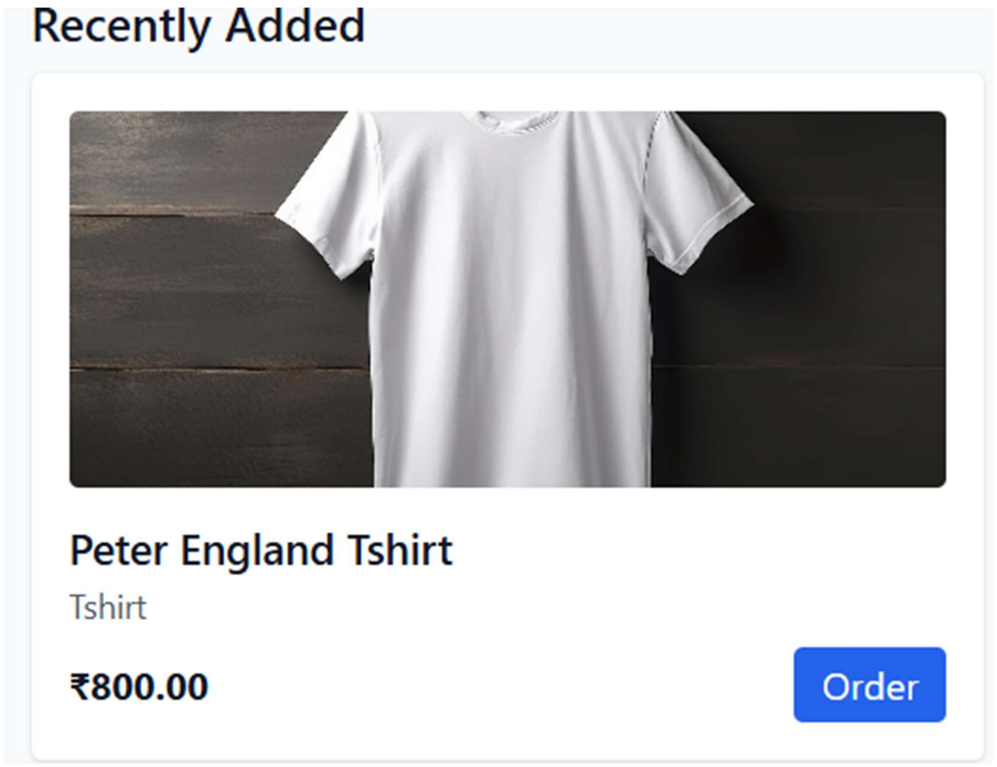
==
TITLE PAGE
==
Project: QuickShop — A Lightweight E-commerce Storefront
Author: [Your Name]
Email: [Your Email]
Date: [Submission Date]

Short Description:
QuickShop is a lightweight e-commerce storefront where customers browse products and place orders, while admins manage products and oversee orders. Built with React + Vite + Tailwind (frontend), Node.js + Express (backend), MongoDB + Mongoose (DB), JWT + bcrypt for authentication.

==
PROJECT TITLE & SHORT DESCRIPTION
==
Title: QuickShop
Summary: A full-stack e-commerce MVP with authentication, product catalog, category filtering, order placement, and admin management for products and orders. Clean architecture and production-ready scaffolding.

Screenshots:

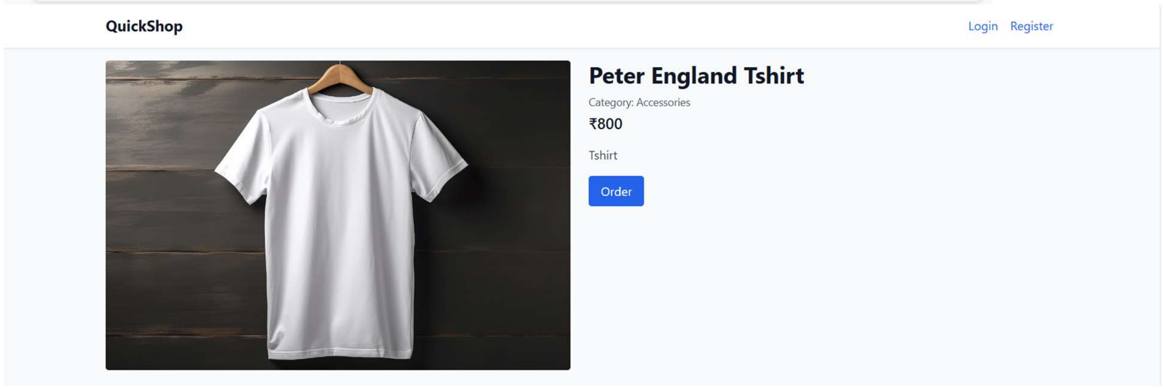



Recently Added

Peter England Tshirt

Tshirt

₹800.00

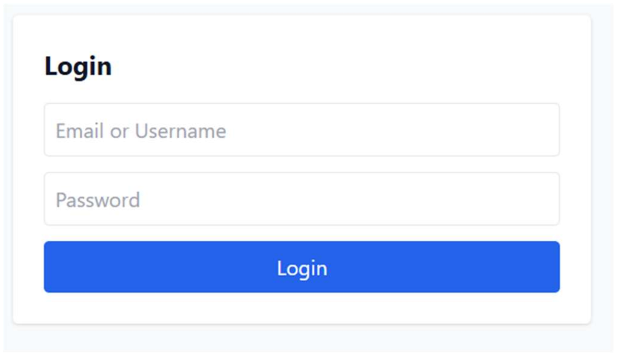

QuickShop

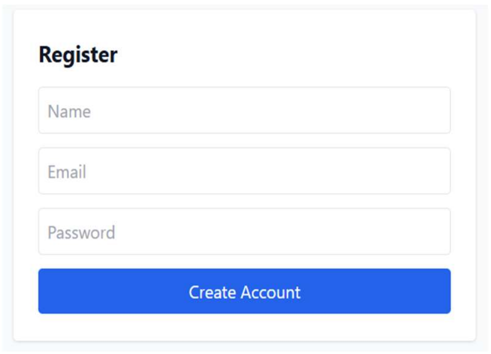
Peter England Tshirt

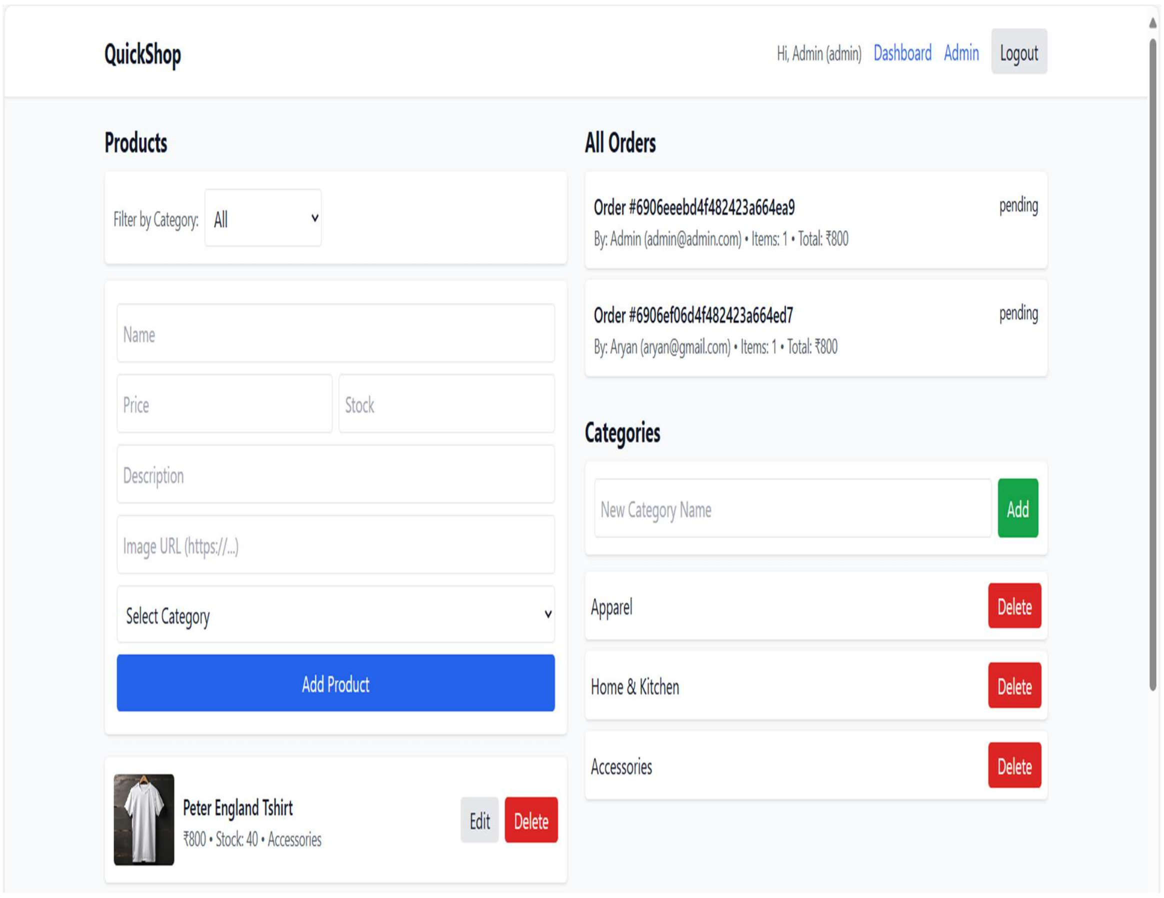
Category: Accessories

₹800

Tshirt


Login


Register


QuickShop

Hi, Admin (admin) Dashboard Admin Logout

Products

Filter by Category: All

Name

Price Stock

Description

Image URL (https://...)

Select Category

Add Product

Peter England Tshirt
₹800 • Stock: 40 • Accessories
Edit Delete

All Orders

Order #6906eeebd4f482423a664ea9 pending
By: Admin (admin@admin.com) • Items: 1 • Total: ₹800

Order #6906ef06d4f482423a664ed7 pending
By: Aryan (aryan@gmail.com) • Items: 1 • Total: ₹800

Categories

New Category Name Add

Apparel Delete

Home & Kitchen Delete

Accessories Delete

==
OVERVIEW OF PROJECT — TYPES OF USERS
==

Customer
- Responsibilities: Browse products, place orders, view own orders.
- Actions: Register, login, view products, create order, view my orders.
- Access: Home, Login, Register, Dashboard (My Orders).
- Restrictions: Cannot manage products; cannot view other users' orders.

Admin
- Responsibilities: Manage product catalog and view all orders.
- Actions: Add/edit/delete products, view all orders, manage categories.
- Access: Admin Dashboard, Home, Login.
- Restrictions: Must be authenticated with role: admin.

==

DATABASE DESIGN / STRUCTURE

==

Collections and fields:

- Users (users)
  _id:ObjectId (PK), name:String, email:String (unique), password:String (bcrypt), role:String(enum: customer,admin, default customer), timestamps
- Categories (categories)
  _id:ObjectId (PK), name:String(unique), slug:String(unique), timestamps
- Products (products)
  _id:ObjectId (PK), name:String, description:String, price:Number, imageUrl:String, stock:Number, category:ObjectId(ref Category), timestamps
- Orders (orders)
  _id:ObjectId (PK), user:ObjectId(ref User), items:[{ product:ObjectId(ref Product), qty:Number }], total:Number, status:String(enum: pending,paid,shipped,completed,cancelled), timestamps

Relationships:

- users (1) — (many) orders
- categories (1) — (many) products
- products (many) — (many) orders.items

==
FRONTEND COMPONENT STRUCTURE
==
App Shell
- App, Navbar, ProtectedRoute, AdminRoute, Loading

Auth
- Pages: Login, Register
- Context: AuthContext (user, token, role; login/logout)

Catalog
- Pages: Home (recent + by category), CategoryPage (/category/:slug), ProductPage (/product/:id)
- Components: ProductCard

Orders
- Pages: Dashboard (customer: my orders)
- Pages: AdminDashboard (admin: manage categories and products; view all orders)

==
BACKEND CONTROLLERS / FUNCTIONS
==
Auth
- POST /api/auth/register → create user, hash password, return JWT
- POST /api/auth/login → accepts identifier (email or username) + password; returns JWT
- GET /api/auth/me → returns profile (auth)

Products
- GET   /api/products → list (supports ?category=slugOrId)
- GET   /api/products/:id → details
- POST   /api/products (admin) → create (name, price, stock, description, imageUrl, category)
- PUT   /api/products/:id (admin) → update fields
- DELETE /api/products/:id (admin) → delete

Categories
- GET   /api/categories → list
- POST   /api/categories (admin) → create by name (slug auto)
- DELETE /api/categories/:id (admin) → delete

Orders
- POST /api/orders (auth) → create from items [{product, qty}] (total computed)
- GET /api/orders/my (auth) → current user orders
- GET /api/orders (admin) → all orders
- GET /api/orders/:id (owner/admin) → details

==
ROLES & PERMISSIONS (AUTHORIZATION SYSTEM)

==
- protect middleware: verifies JWT, sets req.user.
- adminOnly middleware: requires req.user.role === 'admin'.
- Order ownership: allow if admin or req.user._id matches order.user.

==
MODULE CONNECTIVITY & FLOW
==
- Auth: React → /api/auth/login|register → store JWT (localStorage) → attach Authorization header via axios.
- Catalog: React → /api/categories + /api/products → Home shows recent and sections by category; CategoryPage filters; ProductPage details.
- Orders: Create via /api/orders; Dashboard fetches /api/orders/my; Admin fetches /api/orders.

==
BONUS SECTION: Authentication & Security
==
- JWT (7d expiry), bcrypt password hashing.
- CORS restricted to frontend origin; morgan for logging.
- Ideas: rate limiting, helmet, strong password policy, refresh tokens.

Scaling Ideas:
- Stateless services behind a load balancer; MongoDB Atlas; CDN for images; cache product list via HTTP caching and LRU.
- CI/CD with GitHub Actions; deploy API to Render/Fly; frontend to Netlify/Vercel.

==
TECHNOLOGIES USED
==
- Frontend: React 18, Vite 5, Tailwind CSS, React Router
- Backend: Node.js, Express (ES Modules)
- Database: MongoDB, Mongoose
- Auth: JWT, bcrypt
- Tooling: dotenv, cors, morgan, axios

==
HOW TO RUN LOCALLY
==

Backend
1) Copy backend/.env.example to backend/.env
2) Set env:

```
MONGO_URI=mongodb://localhost:27017/quickshop
JWT_SECRET=supersecretjwtkeychangeit
PORT=5000
CORS_ORIGIN=http://localhost:5173
```

3) Install & run:

```
cd backend
npm i
npm run seed   (creates admin@admin.com / 12345678 and demo data)
npm run dev
```

Frontend
1) Copy frontend/.env.example to frontend/.env
2) Ensure: VITE_API_BASE=http://localhost:5000
3) Install & run:

```
cd frontend
npm i
npm run dev
```

Login
- Admin: admin@admin.com / 12345678 → auto-redirect to /admin
- Customer: register/login → redirect to /dashboard

==
CONCLUSION
==

QuickShop delivers a clean, interview-ready e-commerce MVP with a clear separation of concerns, robust JWT auth, category-driven catalog, order flow, and an admin panel capable of managing categories and products (including images). The stack and structure are production-lean and easily extensible for future features such as payments, inventory analytics, and SEO optimizations.