# Building a Free E-commerce Website (Clothing Store)

## 1. Project Overview

You’ll build a full-stack clothing e-commerce website with login, cart, and payments. The project will include a frontend for users, a backend API, and a cloud-hosted database.

## 2. Database Design

Core tables for the e-commerce website include:

* Users

**Relationships:**

* One user → many carts
* One user → many orders
* One user → many shipping\_details

**Script**  
CREATE TABLE users (

id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(150) UNIQUE NOT NULL,

password\_hash VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

* Products

**Relationships:**

* One product → many cart\_items
* One product → many order\_items

Script  
CREATE TABLE products (

id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

description TEXT,

price DECIMAL(10, 2) NOT NULL,

image\_url VARCHAR(255),

stock INT DEFAULT 0

);

* Cart

**Relationships:**

* One cart → many cart\_items
* Script

CREATE TABLE cart (

id SERIAL PRIMARY KEY,

user\_id INT NOT NULL,

status VARCHAR(50) DEFAULT 'ACTIVE',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE

);

* Cart\_items

**Relationships:**

* Belongs to one cart
* References one product

Script

CREATE TABLE cart\_items (

id SERIAL PRIMARY KEY,

cart\_id INT NOT NULL,

product\_id INT NOT NULL,

quantity INT DEFAULT 1,

added\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (cart\_id) REFERENCES cart(id) ON DELETE CASCADE,

FOREIGN KEY (product\_id) REFERENCES products(id) ON DELETE CASCADE

);

* Orders

**Relationships:**

* One order → many order\_items
* One order → one shipping\_details

Script

CREATE TABLE orders (

id SERIAL PRIMARY KEY,

user\_id INT NOT NULL,

total\_price DECIMAL(10, 2) NOT NULL,

payment\_status VARCHAR(50) DEFAULT 'PENDING',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE

);

* Order\_items

**Relationships:**

* Belongs to one order
* References one product

Script

CREATE TABLE order\_items (

id SERIAL PRIMARY KEY,

order\_id INT NOT NULL,

product\_id INT NOT NULL,

quantity INT DEFAULT 1,

price DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (order\_id) REFERENCES orders(id) ON DELETE CASCADE,

FOREIGN KEY (product\_id) REFERENCES products(id) ON DELETE CASCADE

);

* Shipping\_details

**Relationships:**

* Belongs to one user
* Belongs to one order
* Script

CREATE TABLE shipping\_details (

id SERIAL PRIMARY KEY,

order\_id INT NOT NULL,

user\_id INT NOT NULL,

name VARCHAR(100) NOT NULL,

address TEXT NOT NULL,

city VARCHAR(100) NOT NULL,

province VARCHAR(100),

postal\_code VARCHAR(20),

phone\_number VARCHAR(20),

delivery\_status VARCHAR(50) DEFAULT 'Pending',

FOREIGN KEY (order\_id) REFERENCES orders(id) ON DELETE CASCADE,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE

);

Extra tables

* Categories

Script

CREATE TABLE categories (

id SERIAL PRIMARY KEY,

name VARCHAR(100) UNIQUE NOT NULL,

description TEXT

);

ALTER TABLE products

ADD COLUMN category\_id INT,

ADD FOREIGN KEY (category\_id) REFERENCES categories(id) ON DELETE SET NULL;

* Payments

Script

CREATE TABLE payments (

id SERIAL PRIMARY KEY,

order\_id INT NOT NULL,

amount DECIMAL(10,2) NOT NULL,

method VARCHAR(50),

transaction\_id VARCHAR(255),

payment\_status VARCHAR(50) DEFAULT 'PENDING',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (order\_id) REFERENCES orders(id) ON DELETE CASCADE

);

* Product\_images

Script

CREATE TABLE product\_images (

id SERIAL PRIMARY KEY,

product\_id INT NOT NULL,

image\_url VARCHAR(255) NOT NULL,

FOREIGN KEY (product\_id) REFERENCES products(id) ON DELETE CASCADE

);

* Reviews
* Script

CREATE TABLE reviews (

id SERIAL PRIMARY KEY,

user\_id INT NOT NULL,

product\_id INT NOT NULL,

rating INT CHECK (rating BETWEEN 1 AND 5),

comment TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES users(id),

FOREIGN KEY (product\_id) REFERENCES products(id)

);

* Inventory\_log

Script

CREATE TABLE inventory\_log (

id SERIAL PRIMARY KEY,

product\_id INT NOT NULL,

change\_type VARCHAR(50), -- e.g. 'SALE', 'RESTOCK'

quantity\_changed INT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (product\_id) REFERENCES products(id)

);

## 3. Tech Stack (Free Tools)

|  |  |  |
| --- | --- | --- |
| Part | Technology | Free Hosting Option |
| Frontend | React | Vercel |
| Backend | Node.js (Express.js) | Render, Railway, or Cyclic |
| Database | PostgreSQL | Neon, Supabase, or Railway |
| Auth | JWT or Firebase Auth | Firebase |
| Payments | Stripe | Free developer account |
| File Hosting | Cloudinary | Free plan for image storage |

## 4. Setup Flow

Step 1. Database

Create a PostgreSQL database using Neon.tech or Supabase and get the connection string.

Step 2. Backend (Node + Express)

Use Sequelize to connect to the PostgreSQL database and build API routes for users, products, cart, and orders.

Step 3. Frontend (React)

Create pages[and deploy the site to Vercel.

|  |  |  |
| --- | --- | --- |
| Page / Route | Backend API(s) | Purpose / Flow |
| Home / | GET /api/products?featured=true  GET /api/categories | Display featured products, top categories, promotional banners. User can click products or categories to navigate. |
| Products List /products | GET /api/products | Show all products with filters (price, category). Pagination support. Clicking a product navigates to Product Details page. |
| Category /category/:id | GET /api/categories/:id  GET /api/categories/:id/products | Show all products under a specific category. User can filter/sort products. |
| Product Details /product/:id | GET /api/products/:id  GET /api/products/:id/reviews  GET /api/inventory-logs?product\_id=:id  POST /api/reviews | Display product info, stock, multiple images, and reviews. User can add to cart and submit a review. Inventory logs are visible for admin (optional). |
| Cart /cart | GET /api/cart?user\_id=:id  POST /api/cart-items  PUT /api/cart-items/:id  DELETE /api/cart-items/:id | Show items in the user’s cart. Allow quantity updates, removals, and proceed to checkout. |
| Checkout /checkout | POST /api/orders  POST /api/shipping-details  POST /api/payments | Collect shipping info and payment. Create order and order\_items, generate payment, and redirect to Payment Status page. |
| Payment Status /payment-status | GET /api/payments/:transaction\_id | Display confirmation of payment success/failure. Link to invoice or order history. |
| Login / Register /login, /register | POST /api/auth/login  POST /api/auth/register | Authenticate users and create accounts. JWT token returned or Firebase handled authentication. |
| Forgot Password /forgot-password | POST /api/auth/forgot-password  POST /api/auth/reset-password | Allow user to request password reset and update password securely. |
| Order History /orders | GET /api/orders?user\_id=:id | List all past orders of the logged-in user with summary info (status, total). Click an order to view invoice/details. |
| Invoice / Order Details /orders/:id | GET /api/orders/:id  GET /api/order-items?order\_id=:id  GET /api/shipping-details?order\_id=:id  GET /api/payments?order\_id=:id | Detailed view of a single order, including purchased products, quantities, shipping info, payment info, and total amount. Option to download PDF invoice. |
| Admin Inventory Logs /admin/inventory | GET /api/inventory-logs | Admin view of all stock changes per product. Shows change type, quantity, reason, and timestamp. |
| Admin Products Management (Optional) /admin/products | GET /api/products  POST /api/products  PUT /api/products/:id  DELETE /api/products/:id | Add, edit, or delete products. Manage stock levels. |
| Settings/  Profile /settings | GET /api/users/:id  PUT /api/users/:id  PUT /api/auth/change-password  GET/POST/PUT/DELETE /api/shipping-details | - Display user profile info (name, email, shipping addresses).  Allow user to update profile details.  Option to change password (calls change-password API).  Optionally manage saved addresses (shipping\_details). |
|  |  |  |

**Complete Data Flow Summary**

**User visits Home / Products / Category pages**

* Frontend calls GET /api/products?featured=true and GET /api/categories.
* Data displayed as cards/grid.
* User clicks a product or category to navigate.

**User views Product Details**

* Fetch product info → GET /api/products/:id.
* Fetch product images → GET /api/product-images?product\_id=:id.
* Fetch reviews → GET /api/products/:id/reviews.
* Fetch inventory logs (optional for admin) → GET /api/inventory-logs?product\_id=:id.
* Add review → POST /api/reviews.
* Add to cart → POST /api/cart-items.

**User manages Cart**

* Fetch current cart items → GET /api/cart?user\_id=:id.
* Update quantity → PUT /api/cart-items/:id.
* Remove item → DELETE /api/cart-items/:id.

**User proceeds to Checkout**

* Submit shipping details → POST /api/shipping-details.
* Create order → POST /api/orders.
* Create order items → POST /api/order-items.
* Process payment → POST /api/payments.
* Redirect to Payment Status page → GET /api/payments/:transaction\_id.

**User checks Order History or Invoice**

* Fetch all orders → GET /api/orders?user\_id=:id.
* View single order / invoice → GET /api/orders/:id + related GET /api/order-items?order\_id=:id + GET /api/shipping-details?order\_id=:id + GET /api/payments?order\_id=:id.

**User Account / Settings**

* Fetch user profile → GET /api/users/:id.
* Update profile → PUT /api/users/:id.
* Change password → PUT /api/auth/change-password.
* Manage saved addresses → GET/POST/PUT/DELETE /api/shipping-details.

**Authentication**

* Login → POST /api/auth/login.
* Register → POST /api/auth/register.
* Forgot password → POST /api/auth/forgot-password and reset → POST /api/auth/reset-password.

**Admin views Inventory Logs**

* Fetch inventory changes → GET /api/inventory-logs.

**Admin manages Products (optional)**

* Fetch products → GET /api/products.
* Add product → POST /api/products.
* Update product → PUT /api/products/:id.
* Delete product → DELETE /api/products/:id.

**Reviews Flow**

* Fetch product reviews → GET /api/products/:id/reviews.
* Add review → POST /api/reviews.
* Admin can moderate (optional).

Step 4. Payment Integration (Stripe)

Create a Stripe account and integrate test payment keys with your backend.

## 5. Free Hosting Summary

|  |  |  |
| --- | --- | --- |
| Component | Free Option | Notes |
| Database | Neon.tech / Supabase | PostgreSQL with free tier |
| Backend | Render / Railway / Cyclic.sh | Node.js hosting with free plan |
| Frontend | Vercel / Netlify | Easy GitHub deploys |
| Images | Cloudinary | Free 25GB/month |
| Payments | Stripe (test mode) | Use sandbox until real launch |

## 6. Authentication (Login/Signup)

Option A: Custom authentication using bcrypt for password hashing and JWT for login tokens.

Option B: Use Firebase Authentication for a faster and simpler setup.

## 7. Example Free Full-Stack Stack

|  |  |
| --- | --- |
| Part | Choice |
| Frontend | React (Vercel) |
| Backend | Node.js + Express (Render) |
| Database | PostgreSQL (Neon) |
| Images | Cloudinary |
| Auth | JWT |
| Payments | Stripe (test mode) |