**Complete Luxury Fashion E-Commerce Setup Guide**

**📁 Complete Project Structure**

luxury-fashion-ecommerce/

│

├── backend/

│ ├── config/

│ │ ├── database.js

│ │ └── redis.js

│ │

│ ├── controllers/

│ │ ├── auth.controller.js

│ │ ├── product.controller.js

│ │ ├── cart.controller.js

│ │ ├── order.controller.js

│ │ ├── user.controller.js

│ │ ├── reward.controller.js

│ │ ├── story.controller.js

│ │ └── admin.controller.js

│ │

│ ├── middleware/

│ │ ├── auth.middleware.js

│ │ ├── error.middleware.js

│ │ ├── upload.middleware.js

│ │ └── validation.middleware.js

│ │

│ ├── models/

│ │ ├── User.js

│ │ ├── Product.js

│ │ ├── Order.js

│ │ ├── Cart.js

│ │ ├── Category.js

│ │ ├── Review.js

│ │ ├── Story.js

│ │ ├── Reward.js

│ │ └── Coupon.js

│ │

│ ├── routes/

│ │ ├── auth.routes.js

│ │ ├── product.routes.js

│ │ ├── cart.routes.js

│ │ ├── order.routes.js

│ │ ├── user.routes.js

│ │ ├── reward.routes.js

│ │ ├── story.routes.js

│ │ └── admin.routes.js

│ │

│ ├── services/

│ │ ├── email.service.js

│ │ ├── payment.service.js

│ │ ├── cloudinary.service.js

│ │ └── analytics.service.js

│ │

│ ├── utils/

│ │ ├── helpers.js

│ │ ├── constants.js

│ │ └── validators.js

│ │

│ ├── .env

│ ├── .env.example

│ ├── .gitignore

│ ├── package.json

│ ├── package-lock.json

│ ├── server.js

│ └── Dockerfile

│

├── frontend/

│ ├── app/

│ │ ├── (auth)/

│ │ │ ├── login/

│ │ │ │ └── page.tsx

│ │ │ ├── register/

│ │ │ │ └── page.tsx

│ │ │ └── forgot-password/

│ │ │ └── page.tsx

│ │ │

│ │ ├── (shop)/

│ │ │ ├── products/

│ │ │ │ ├── page.tsx

│ │ │ │ └── [id]/

│ │ │ │ └── page.tsx

│ │ │ ├── category/

│ │ │ │ └── [slug]/

│ │ │ │ └── page.tsx

│ │ │ ├── cart/

│ │ │ │ └── page.tsx

│ │ │ ├── checkout/

│ │ │ │ └── page.tsx

│ │ │ └── wishlist/

│ │ │ └── page.tsx

│ │ │

│ │ ├── (user)/

│ │ │ ├── profile/

│ │ │ │ └── page.tsx

│ │ │ ├── orders/

│ │ │ │ ├── page.tsx

│ │ │ │ └── [id]/

│ │ │ │ └── page.tsx

│ │ │ └── rewards/

│ │ │ └── page.tsx

│ │ │

│ │ ├── admin/

│ │ │ ├── dashboard/

│ │ │ │ └── page.tsx

│ │ │ ├── products/

│ │ │ │ ├── page.tsx

│ │ │ │ ├── new/

│ │ │ │ │ └── page.tsx

│ │ │ │ └── edit/

│ │ │ │ └── [id]/

│ │ │ │ └── page.tsx

│ │ │ ├── orders/

│ │ │ │ └── page.tsx

│ │ │ ├── stories/

│ │ │ │ └── page.tsx

│ │ │ └── users/

│ │ │ └── page.tsx

│ │ │

│ │ ├── layout.tsx

│ │ ├── page.tsx

│ │ └── globals.css

│ │

│ ├── components/

│ │ ├── layout/

│ │ │ ├── Header.tsx

│ │ │ ├── Footer.tsx

│ │ │ ├── Navigation.tsx

│ │ │ └── MobileMenu.tsx

│ │ │

│ │ ├── home/

│ │ │ ├── HeroBanner.tsx

│ │ │ ├── StoryReel.tsx

│ │ │ ├── RewardsBar.tsx

│ │ │ ├── FeaturedProducts.tsx

│ │ │ ├── ShopByOccasion.tsx

│ │ │ ├── BestSellers.tsx

│ │ │ ├── InfluencerSection.tsx

│ │ │ └── Newsletter.tsx

│ │ │

│ │ ├── product/

│ │ │ ├── ProductCard.tsx

│ │ │ ├── ProductGrid.tsx

│ │ │ ├── ProductFilters.tsx

│ │ │ ├── QuickViewModal.tsx

│ │ │ ├── SizeGuide.tsx

│ │ │ ├── ProductGallery.tsx

│ │ │ └── ReviewSection.tsx

│ │ │

│ │ ├── cart/

│ │ │ ├── CartItem.tsx

│ │ │ ├── CartSummary.tsx

│ │ │ └── CartDrawer.tsx

│ │ │

│ │ ├── checkout/

│ │ │ ├── ShippingForm.tsx

│ │ │ ├── PaymentForm.tsx

│ │ │ ├── OrderSummary.tsx

│ │ │ └── CheckoutSteps.tsx

│ │ │

│ │ ├── ui/

│ │ │ ├── Button.tsx

│ │ │ ├── Input.tsx

│ │ │ ├── Modal.tsx

│ │ │ ├── Spinner.tsx

│ │ │ ├── Badge.tsx

│ │ │ └── Toast.tsx

│ │ │

│ │ └── admin/

│ │ ├── StatsCard.tsx

│ │ ├── RevenueChart.tsx

│ │ ├── OrdersTable.tsx

│ │ └── ProductForm.tsx

│ │

│ ├── lib/

│ │ ├── api.ts

│ │ ├── utils.ts

│ │ ├── constants.ts

│ │ └── hooks.ts

│ │

│ ├── store/

│ │ ├── index.ts

│ │ ├── authSlice.ts

│ │ ├── cartSlice.ts

│ │ ├── wishlistSlice.ts

│ │ └── types.ts

│ │

│ ├── styles/

│ │ └── globals.css

│ │

│ ├── public/

│ │ ├── images/

│ │ ├── fonts/

│ │ └── icons/

│ │

│ ├── .env.local

│ ├── .env.local.example

│ ├── .gitignore

│ ├── next.config.js

│ ├── tailwind.config.js

│ ├── tsconfig.json

│ ├── package.json

│ ├── package-lock.json

│ └── Dockerfile

│

├── docker-compose.yml

├── .gitignore

└── README.md

**📋 Step-by-Step Setup Instructions**

**Step 1: Create Project Directory**

# Create main project directory

mkdir luxury-fashion-ecommerce

cd luxury-fashion-ecommerce

# Create backend and frontend directories

mkdir backend frontend

**Step 2: Backend Setup**

**2.1 Initialize Backend**

cd backend

npm init -y

**2.2 Create Backend Directory Structure**

# Create all backend directories

mkdir config controllers middleware models routes services utils

# Create subdirectories if needed

cd ..

**2.3 Extract Backend Files from Artifact**

From the main artifact, copy the following sections to their respective files:

**package.json (backend/package.json)**

{

"name": "luxury-fashion-backend",

"version": "1.0.0",

"main": "server.js",

"scripts": {

"start": "node server.js",

"dev": "nodemon server.js"

},

"dependencies": {

"express": "^4.18.2",

"cors": "^2.8.5",

"dotenv": "^16.3.1",

"bcryptjs": "^2.4.3",

"jsonwebtoken": "^9.0.2",

"mongoose": "^7.5.0",

"multer": "^1.4.5-lts.1",

"cloudinary": "^1.40.0",

"stripe": "^13.5.0",

"nodemailer": "^6.9.4",

"express-validator": "^7.0.1",

"express-rate-limit": "^6.10.0",

"helmet": "^7.0.0",

"compression": "^1.7.4",

"redis": "^4.6.8",

"socket.io": "^4.5.2"

},

"devDependencies": {

"nodemon": "^3.0.1"

}

}

**server.js (backend/server.js)**

* Copy the content between // === server.js === from the artifact
* Remove the backticks and save

**models/User.js (backend/models/User.js)**

* Copy the content between // === models/User.js === from the artifact
* Remove the backticks and save

**models/Product.js (backend/models/Product.js)**

* Copy the content between // === models/Product.js === from the artifact
* Remove the backticks and save

**models/Order.js (backend/models/Order.js)**

* Copy the content between // === models/Order.js === from the artifact
* Remove the backticks and save

**models/Story.js (backend/models/Story.js)**

* Copy the content between // === models/Story.js === from the artifact
* Remove the backticks and save

**controllers/auth.controller.js (backend/controllers/auth.controller.js)**

* Copy the content between // === controllers/auth.controller.js === from the artifact
* Remove the backticks and save

**controllers/product.controller.js (backend/controllers/product.controller.js)**

* Copy the content between // === controllers/product.controller.js === from the artifact
* Remove the backticks and save

**middleware/auth.middleware.js (backend/middleware/auth.middleware.js)**

* Copy the content between // === middleware/auth.middleware.js === from the artifact
* Remove the backticks and save

**.env (backend/.env)**

NODE\_ENV=development

PORT=5000

MONGODB\_URI=mongodb://localhost:27017/luxury-fashion

REDIS\_URL=redis://localhost:6379

JWT\_SECRET=your-super-secret-jwt-key-change-this-in-production

JWT\_EXPIRES\_IN=7d

CLIENT\_URL=http://localhost:3000

# Cloudinary (for image uploads)

CLOUDINARY\_CLOUD\_NAME=your-cloud-name

CLOUDINARY\_API\_KEY=your-api-key

CLOUDINARY\_API\_SECRET=your-api-secret

# Stripe (for payments)

STRIPE\_SECRET\_KEY=your-stripe-secret-key

STRIPE\_WEBHOOK\_SECRET=your-webhook-secret

# Email (Nodemailer)

EMAIL\_HOST=smtp.gmail.com

EMAIL\_PORT=587

EMAIL\_USER=your-email@gmail.com

EMAIL\_PASS=your-app-password

**2.4 Create Missing Route Files**

Since not all route files were in the artifact, create these basic structures:

**routes/auth.routes.js (backend/routes/auth.routes.js)**

const express = require('express');

const router = express.Router();

const authController = require('../controllers/auth.controller');

router.post('/register', authController.register);

router.post('/login', authController.login);

router.get('/verify-email/:token', authController.verifyEmail);

module.exports = router;

**routes/product.routes.js (backend/routes/product.routes.js)**

const express = require('express');

const router = express.Router();

const productController = require('../controllers/product.controller');

const { protect } = require('../middleware/auth.middleware');

router.get('/', productController.getProducts);

router.get('/featured', productController.getFeaturedProducts);

router.get('/best-sellers', productController.getBestSellers);

router.get('/:id', productController.getProductById);

router.get('/by-occasion/:occasion', productController.getProductsByOccasion);

router.get('/by-video/:videoId', productController.getProductsByVideo);

module.exports = router;

**routes/cart.routes.js (backend/routes/cart.routes.js)**

const express = require('express');

const router = express.Router();

const { protect } = require('../middleware/auth.middleware');

router.use(protect); // All cart routes require authentication

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

res.json({ message: 'Get cart' });

});

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

res.json({ message: 'Add to cart' });

});

router.put('/items/:id', (req, res) => {

res.json({ message: 'Update cart item' });

});

router.delete('/items/:id', (req, res) => {

res.json({ message: 'Remove from cart' });

});

module.exports = router;

**Step 3: Frontend Setup**

**3.1 Initialize Frontend**

cd ../frontend

npx create-next-app@latest . --typescript --tailwind --app

When prompted:

* Would you like to use ESLint? → Yes
* Would you like to use src/ directory? → No
* Would you like to customize the default import alias? → No

**3.2 Install Additional Dependencies**

npm install @reduxjs/toolkit react-redux @tanstack/react-query axios framer-motion react-hot-toast react-hook-form swiper react-intersection-observer @stripe/stripe-js socket.io-client date-fns react-icons

**3.3 Extract Frontend Files from Artifact**

**package.json (Update frontend/package.json dependencies)**

Add these to your existing package.json:

{

"dependencies": {

"@reduxjs/toolkit": "^1.9.5",

"react-redux": "^8.1.2",

"@tanstack/react-query": "^4.35.0",

"axios": "^1.5.0",

"framer-motion": "^10.16.0",

"react-hot-toast": "^2.4.1",

"react-hook-form": "^7.46.0",

"swiper": "^10.2.0",

"react-intersection-observer": "^9.5.2",

"@stripe/stripe-js": "^2.1.0",

"socket.io-client": "^4.5.2",

"date-fns": "^2.30.0",

"react-icons": "^4.11.0"

}

}

**tailwind.config.js (frontend/tailwind.config.js)**

* Copy the content between // === tailwind.config.js === from the artifact
* Remove the backticks and save

**app/page.tsx (frontend/app/page.tsx)**

* Copy the content between // === app/page.tsx (Homepage) === from the artifact
* Remove the backticks and save

**components/home/StoryReel.tsx (frontend/components/home/StoryReel.tsx)**

* Create the directory: mkdir -p components/home
* Copy the content between // === components/StoryReel.tsx === from the artifact
* Remove the backticks and save

**components/product/ProductCard.tsx (frontend/components/product/ProductCard.tsx)**

* Create the directory: mkdir -p components/product
* Copy the content between // === components/ProductCard.tsx === from the artifact
* Remove the backticks and save

**components/home/RewardsBar.tsx (frontend/components/home/RewardsBar.tsx)**

* Copy the content between // === components/RewardsBar.tsx === from the artifact
* Remove the backticks and save

**store/index.ts (frontend/store/index.ts)**

* Create the directory: mkdir store
* Copy the content between // === store/index.ts (Redux Store) === from the artifact
* Remove the backticks and save

**store/cartSlice.ts (frontend/store/cartSlice.ts)**

* Copy the content between // === store/cartSlice.ts === from the artifact
* Remove the backticks and save

**lib/api.ts (frontend/lib/api.ts)**

* Create the directory: mkdir lib
* Copy the content between // === lib/api.ts (API Client) === from the artifact
* Remove the backticks and save

**components/checkout/Checkout.tsx (frontend/components/checkout/Checkout.tsx)**

* Create the directory: mkdir -p components/checkout
* Copy the content between // === components/Checkout.tsx === from the artifact
* Remove the backticks and save

**components/admin/AdminDashboard.tsx (frontend/components/admin/AdminDashboard.tsx)**

* Create the directory: mkdir -p components/admin
* Copy the content between // === Admin Dashboard Component === from the artifact
* Remove the backticks and save

**.env.local (frontend/.env.local)**

NEXT\_PUBLIC\_API\_URL=http://localhost:5000/api

NEXT\_PUBLIC\_STRIPE\_PUBLIC\_KEY=your-stripe-public-key

NEXT\_PUBLIC\_GOOGLE\_ANALYTICS\_ID=G-XXXXXXXXXX

**Step 4: Create Root Files**

**docker-compose.yml (root directory)**

version: '3.8'

services:

backend:

build: ./backend

ports:

- "5000:5000"

environment:

- NODE\_ENV=production

- MONGODB\_URI=mongodb://mongo:27017/luxury-fashion

- REDIS\_URL=redis://redis:6379

depends\_on:

- mongo

- redis

frontend:

build: ./frontend

ports:

- "3000:3000"

environment:

- NEXT\_PUBLIC\_API\_URL=http://backend:5000/api

depends\_on:

- backend

mongo:

image: mongo:6

volumes:

- mongo-data:/data/db

ports:

- "27017:27017"

redis:

image: redis:7-alpine

ports:

- "6379:6379"

volumes:

mongo-data:

**.gitignore (root directory)**

# Dependencies

node\_modules/

.pnp

.pnp.js

# Testing

coverage/

# Production

build/

dist/

.next/

out/

# Misc

.DS\_Store

\*.pem

# Debug

npm-debug.log\*

yarn-debug.log\*

yarn-error.log\*

# Local env files

.env

.env.local

.env.development.local

.env.test.local

.env.production.local

# Vercel

.vercel

# TypeScript

\*.tsbuildinfo

next-env.d.ts

# IDE

.vscode/

.idea/

\*.swp

\*.swo

# OS

Thumbs.db

**🚀 Running the Application**

**Development Mode**

**Terminal 1: Start MongoDB**

# If using Docker

docker run -d -p 27017:27017 --name mongodb mongo:6

# Or if MongoDB is installed locally

mongod

**Terminal 2: Start Redis**

# If using Docker

docker run -d -p 6379:6379 --name redis redis:7-alpine

# Or if Redis is installed locally

redis-server

**Terminal 3: Start Backend**

cd backend

npm install

npm run dev

**Terminal 4: Start Frontend**

cd frontend

npm install

npm run dev

**Production Mode with Docker**

# From root directory

docker-compose up -d

**📝 Additional Setup Steps**

**1. Install MongoDB and Redis locally (if not using Docker)**

**MongoDB:**

* Download from: https://www.mongodb.com/try/download/community
* Follow installation instructions for your OS

**Redis:**

* Download from: https://redis.io/download
* Follow installation instructions for your OS

**2. Configure Cloudinary Account**

* Sign up at: https://cloudinary.com
* Get your Cloud Name, API Key, and API Secret
* Update .env file

**3. Configure Stripe Account**

* Sign up at: https://stripe.com
* Get your Secret Key and Publishable Key
* Update .env and .env.local files

**4. Configure Email Service**

* Enable 2-factor authentication in Gmail
* Generate an App Password
* Update EMAIL\_USER and EMAIL\_PASS in .env

**🎨 Creating Missing Components**

For components not included in the artifact, here's a template structure:

**Example: HeroBanner.tsx**

import { motion } from 'framer-motion';

import Link from 'next/link';

export default function HeroBanner() {

return (

<section className="relative h-[70vh] bg-gradient-to-r from-luxury-blue to-luxury-gold">

<div className="absolute inset-0 bg-black/20" />

<div className="relative z-10 flex items-center justify-center h-full text-white">

<motion.div

initial={{ opacity: 0, y: 20 }}

animate={{ opacity: 1, y: 0 }}

className="text-center px-4"

>

<h1 className="text-5xl md:text-7xl font-serif mb-4">

Luxury Redefined

</h1>

<p className="text-xl mb-8">

Discover the Latest Collection

</p>

<Link href="/products">

<button className="bg-white text-luxury-blue px-8 py-3 rounded-full font-medium hover:shadow-lg transition-all">

Shop Now

</button>

</Link>

</motion.div>

</div>

</section>

);

}

**Example: FeaturedProducts.tsx**

import { useQuery } from '@tanstack/react-query';

import { productAPI } from '@/lib/api';

import ProductCard from '@/components/product/ProductCard';

export default function FeaturedProducts() {

const { data: products, isLoading } = useQuery({

queryKey: ['featured-products'],

queryFn: productAPI.getFeaturedProducts,

});

if (isLoading) {

return <div>Loading...</div>;

}

return (

<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6">

{products?.data?.map((product) => (

<ProductCard key={product.id} product={product} />

))}

</div>

);

}

**🔧 Troubleshooting**

**Common Issues and Solutions**

1. **MongoDB Connection Error**
   * Ensure MongoDB is running
   * Check MONGODB\_URI in .env file
   * Try mongodb://127.0.0.1:27017/luxury-fashion instead of localhost
2. **Redis Connection Error**
   * Ensure Redis is running
   * Check REDIS\_URL in .env file
   * Try redis://127.0.0.1:6379 instead of localhost
3. **CORS Issues**
   * Update CLIENT\_URL in backend .env
   * Ensure frontend URL matches in CORS configuration
4. **Module Not Found Errors**
   * Run npm install in both directories
   * Delete node\_modules and package-lock.json, then reinstall
5. **TypeScript Errors**
   * Create missing type definitions
   * Use any type temporarily for quick fixes
   * Run npm run build to check for errors

**📚 Next Steps**

1. **Database Seeding**
   * Create seed scripts for initial data
   * Add sample products, categories, and users
2. **Testing**
   * Add Jest for unit testing
   * Add Cypress for E2E testing
3. **Performance Optimization**
   * Implement image optimization
   * Add PWA support
   * Configure CDN
4. **Security Enhancements**
   * Add CSRF protection
   * Implement rate limiting per user
   * Add input sanitization
5. **Analytics Integration**
   * Set up Google Analytics
   * Add custom event tracking
   * Implement conversion tracking

**🎉 Congratulations!**

Your luxury fashion e-commerce platform is now set up and ready for development. The modular structure allows for easy customization and scaling as your business grows.