

# **E-Commerce Project Documentation**

**Title:** E-Commerce Web Application with Admin Seller Functionality

**Date:** December 2024

**Student Name:** Malik Stewart

**Major:** Applied Computing

**Degree:** Bachelor of Science in Applied Computing: Business Concentration

**Project Advisor:** Professor Henderson

---

## **Statement of Purpose**

This project aims to develop a versatile e-commerce platform that is initially tailored to meet the needs of small clothing brand owners, providing them with a credible space to showcase and sell their products. However, the design and structure of this platform are intended to be adaptable, allowing it to be used by any seller, in any niche, to create a unique online store. For the current phase, the platform supports a single admin user as the sole seller, but it is designed to scale and accommodate future enhancements such as the ability for users to create their own shops and sell products.

## **Problem Statement:**

Small businesses, particularly clothing brand owners, often struggle to set up professional e-commerce websites due to high costs and technical barriers. This project solves that problem by providing an easy-to-use platform where sellers can display and sell their products while ensuring a seamless shopping experience for customers. The system is designed to be flexible, so in the future, it can support sellers from any industry.

---

## **Research & Background**

E-commerce platforms are an essential tool for businesses, especially for small retailers who might not have the resources to build custom websites. This project

focuses on creating an e-commerce solution for small clothing brand owners but ensures that the system is adaptable to any niche market. The flexible design will allow users in different industries to use the platform to set up their stores and sell products.

The platform uses **Ruby on Rails** for backend development, **Stripe API** for secure payment processing, and **Tailwind CSS** for modern, responsive frontend design. These technologies were chosen for their reliability, scalability, and ease of use, ensuring that the platform can evolve to serve a broader audience as more sellers join.

---

## Project Language(s), Software, and Hardware

- **Languages:**

- Ruby (Rails framework) for backend development
- HTML, CSS (Tailwind CSS) for frontend styling
- JavaScript for front-end interactivity

- **Software:**

- Ruby on Rails (version 7.x)
- PostgreSQL for database management
- Stripe API for payment gateway integration
- Git and GitHub for version control
- Visual Studio Code as IDE
- WSL2 and Ubuntu

- **Hardware:**

- Development environment: Local machine (Windows)
- Possible Deployment: Heroku/Production server with PostgreSQL

---

## Project Requirements

- **Functional Requirements:**

- **Product Catalog:**

The product catalog allows users to browse available products, view their details, and add them to the cart. This catalog is designed to be modular and scalable, supporting multiple product types and categories in the future.

**Technical Detail:** The catalog uses ActiveRecord models to retrieve products from the PostgreSQL database and display them on the frontend. Product categories and filters are implemented using query parameters.

- **Shopping Cart System:**

The shopping cart enables customers to add products to their cart, view item details, and proceed to checkout.

**Technical Detail:** The cart uses Rails sessions to store user-selected items temporarily. Items are saved with product IDs, quantities, and total prices, allowing for easy updates and checkout calculations.

- **Payment Integration (Stripe):**

The platform integrates **Stripe API** for payment processing, ensuring secure and efficient transactions.

**Technical Detail:** Stripe's Checkout API is integrated, handling the card processing, with webhook handling to confirm successful payments and update order statuses. The payment flow is secured using HTTPS and Stripe's encryption protocols.

- **Admin Interface:**

The admin manages product listings, inventory, and order processing. The current version supports a single admin, but the code structure accommodates future multi-seller features.

**Technical Detail:** Rails Admin is used to manage products and orders, providing a simple, web-based interface to interact with the backend database.

- 

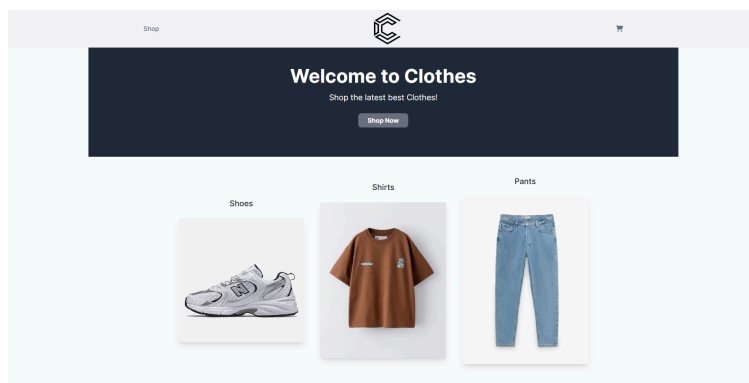
- **Non-functional Requirements:**

- **Scalability:**  
The platform is designed to scale for future multi-seller functionality.  
**Technical Detail:** Models like Product and Order are structured to allow for multiple users. Future development will include adding relationships between users (sellers) and their products, ensuring the system can handle more vendors efficiently.
- **Responsive Design:**  
Tailwind CSS ensures that the platform is fully responsive, providing an optimal experience for both mobile and desktop users.  
**Technical Detail:** The frontend design uses Tailwind's utility classes to manage layout changes dynamically, ensuring the site adapts to different screen sizes without the need for custom media queries.
- 

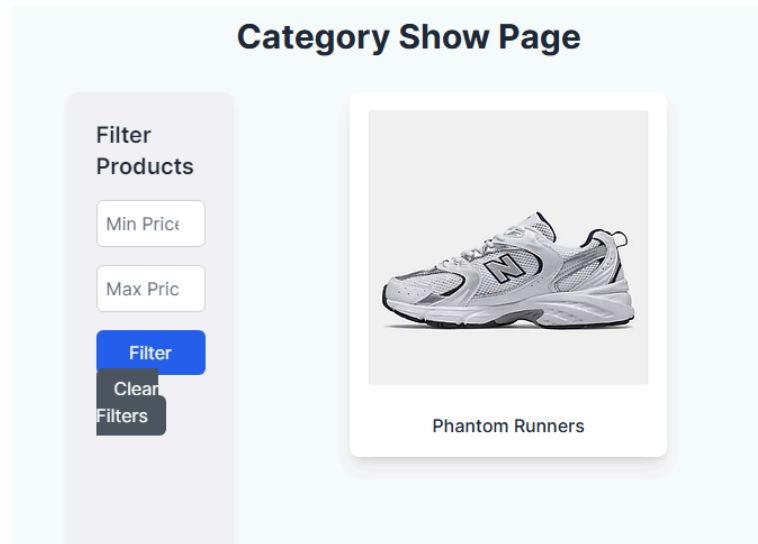
---

## Project Implementation Description & Explanation

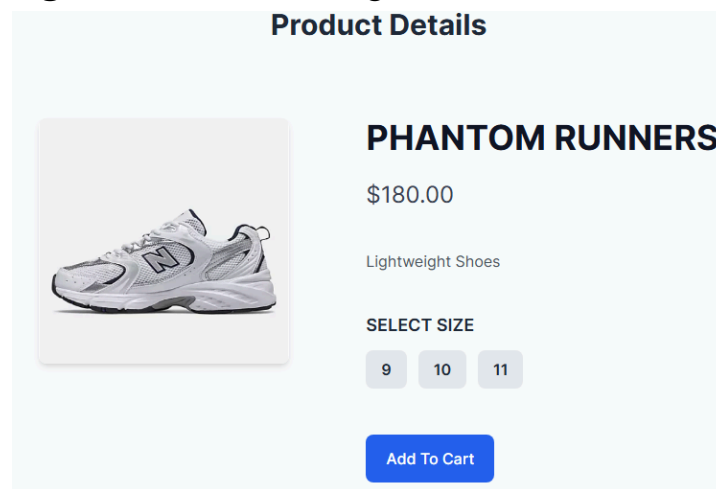
1. **Product Catalog:** The product catalog allows users to browse available products, view details, and add items to their shopping cart.
  - **Technical Details:** The catalog is populated using a simple Rails controller that pulls product data from PostgreSQL. Filtering options are implemented using ActiveRecord queries to display products by price range.
  - **Fig 1:** User home page



- Features Navbar with link to cart and home page
- Features categories the user can visit
- **Fig 2: Category Show Page**



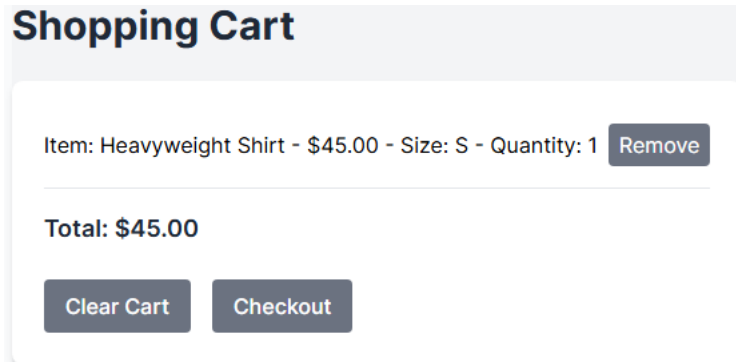
- - Features products in that category
  - Has a filter that allows the user to filter products by price.
- **Fig 3: Product Show Page**



- Has product details (Price, description, sizes, name, picture)
  - User can select sizes and add to cart
- 2. **Shopping Cart and Checkout:** Users can add items to the cart, update quantities, and proceed to checkout. The checkout process integrates with Stripe for payment processing.
  - **Technical Details:** Cart updates are handled through JavaScript (AJAX) for smooth, client-side interactions. At checkout, the total

price is calculated dynamically, and Stripe's frontend library handles secure payment submission.

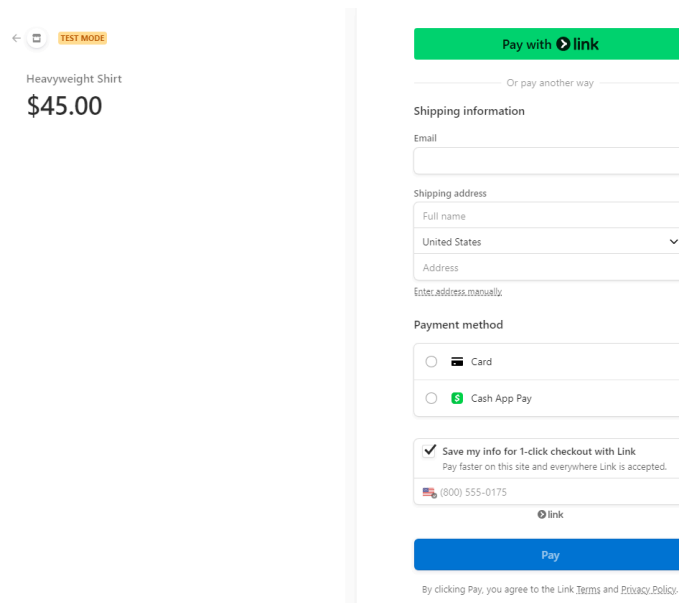
- **Fig 4:** Shopping cart with products and total price.



The image shows a shopping cart interface. At the top, it says "Shopping Cart" in a bold, dark blue font. Below this, there is a list of items. The first item is "Item: Heavyweight Shirt - \$45.00 - Size: S - Quantity: 1". To the right of this item is a "Remove" button. Below the item list, the total price is displayed as "Total: \$45.00". At the bottom of the cart, there are two buttons: "Clear Cart" and "Checkout".

- Features products added to cart, product attributes and total price.
- Ability to remove individual items, the whole cart, or continue checkout.

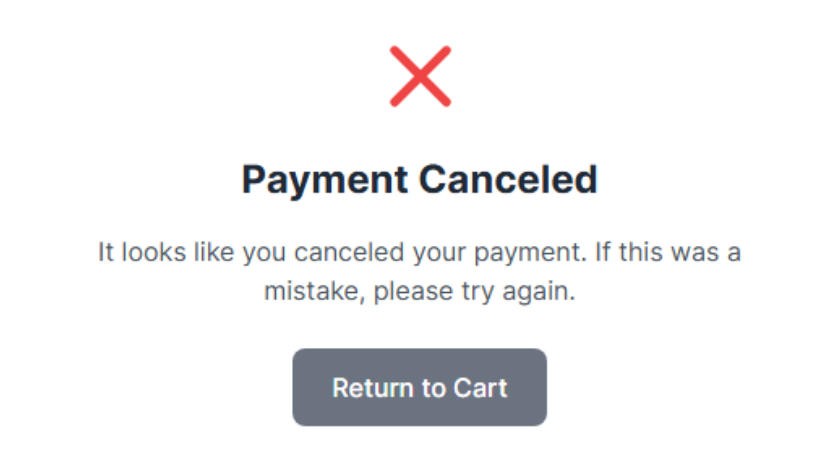
- **Fig 4:** Stripe Checkout Page



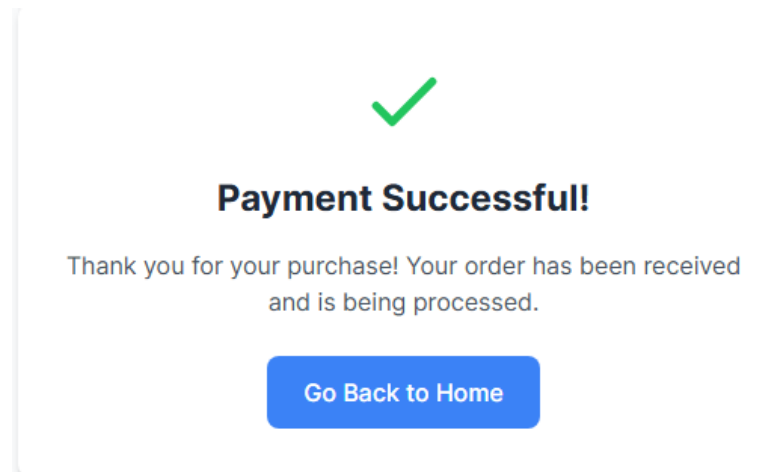
The image shows the Stripe Checkout page. On the left, there is a summary of the order: "Heavyweight Shirt" and "\$45.00". On the right, there is a form for shipping and payment information. The form has a green "Pay with link" button at the top. Below this, there is a section for "Shipping information" with fields for "Email", "Shipping address" (with sub-fields for "Full name", "United States" (a dropdown menu), and "Address"), and a link to "Enter address manually". Below the shipping information, there is a section for "Payment method" with radio buttons for "Card" and "Cash App Pay". At the bottom, there is a checkbox for "Save my info for 1-click checkout with Link" and a "Pay" button. A small note at the bottom says "By clicking Pay, you agree to the Link Terms and Privacy Policy."

- Asks user for all relevant shipping and billing info with multiple payment options
- Displays total order cost.
- Allows the user to cancel checkout process or continue to pay
- Information is updated in STRIPE'S API database and local SQL database

- Decrements stock quantity when order is processed
- **Fig 5: Cancel Page**



- Displays when the buyer cancels the checkout process.
- Allow them to return to the cart.
- **Fig 6: Success Page**



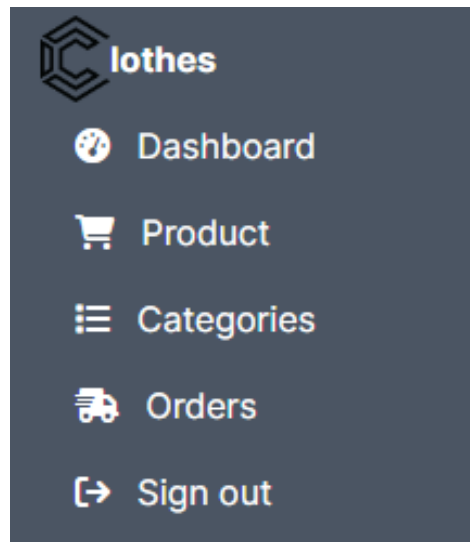
- Informs user that their order was success
- Allows them to go back to home page
- **Fig 7: Stripe Transactions**

Transactions									
<div><div>All17</div><div>Succeeded17</div><div>Refunded0</div><div>Disputed0</div><div>Failed0</div><div>Uncaptured0</div></div>									
<div><div>Date and time</div><div>Amount</div><div>Currency</div><div>Status</div><div>Payment method</div><div>More filters</div><div>Export</div><div>Edit columns</div></div>									
<input type="checkbox"/>	Amount	Payment method	Description	Customer	Date	Refunded date	Dispute amount	Dispute reason	Disputes
<input type="checkbox"/>	\$45.00 USD <span>Succeeded</span>	visa **** 4242	pi_3QSNzmRuoYYU1Ica0nD1eoe5	Jamie@example.com	Dec 4, 2:12 PM	—	—	—	—
<input type="checkbox"/>	\$45.00 USD <span>Succeeded</span>	visa **** 4242	pi_3QSNxeRuoYYU1Ica1dbHk1ek	malik@example.com	Dec 4, 2:10 PM	—	—	—	—
<input type="checkbox"/>	\$45.00 USD <span>Succeeded</span>	visa **** 4242	pi_3QSNpaRuoYYU1Ica0XB38hJ0	malik@example.com	Dec 4, 2:01 PM	—	—	—	—

- Displays transaction info on Stipe's website

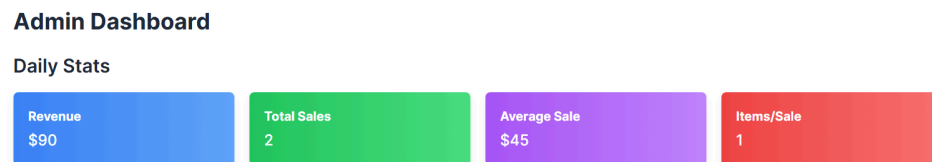
3. **Admin Seller Side:** The admin interface allows the single admin user to manage product listings, inventory, and orders.

- **Technical Details:** Admin functionality is handled via Rails Admin, a gem that simplifies administrative dashboards. Future plans will involve expanding this to allow multiple admins or sellers to have their own interfaces.
- **Fig 8:** Admin navbar



- - Features links to Dashboard, Product, Categories, Orders, Sign Out.
  - Easy Navigation.

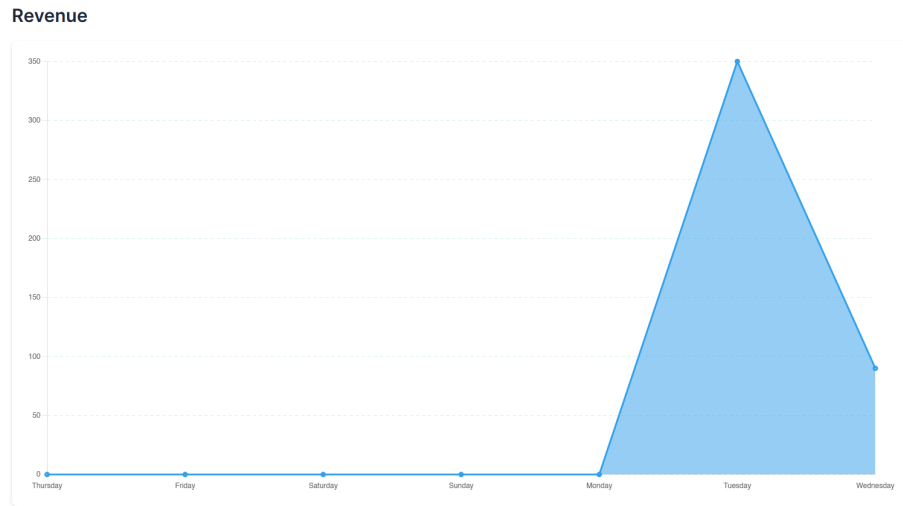
- **Fig 9:** Admin Dashboard



- Display useful info to admin seller (Revenue, Total Sales, Average Sale, Item Per Sale).



○ **Fig 10: Admin Revenue Chart**



- Gives visual representation of sales

○ **Fig 11: Unfulfilled Orders**

Recent Unfulfilled Orders

Order ID	Customer	Date	Amount
<a href="#">20</a>	Jamie@example.com	December 04, 2024 19:12	\$45
<a href="#">19</a>	malik@example.com	December 04, 2024 19:10	\$45
<a href="#">18</a>	malik@example.com	December 03, 2024 18:45	\$180
<a href="#">17</a>	malik@example.com	December 03, 2024 18:38	\$45
<a href="#">16</a>	malik@example.com	December 03, 2024 18:30	\$45

- Displays unfulfilled orders on Admin Dashboard Page
- Indicates high priority

○ **Fig 12: Admin Product Page**

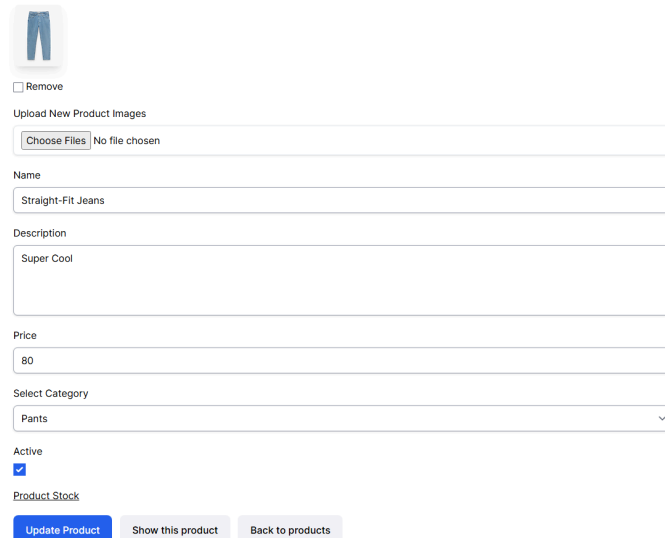
Products New product

Images	Name	Description	Price
	<a href="#">Straight-Fit Jeans</a>	<a href="#">Super Cool</a>	\$80.00
	<a href="#">Phantom Runners</a>	<a href="#">Lightweight Shoes</a>	\$180.00
	<a href="#">Heavyweight Shirt</a>	<a href="#">Soft and cozy</a>	\$45.00

- Displays products in the sellers shop and all related attributes
- Ability to create a new product

○ **Fig 13: Admin Product Edit Page**

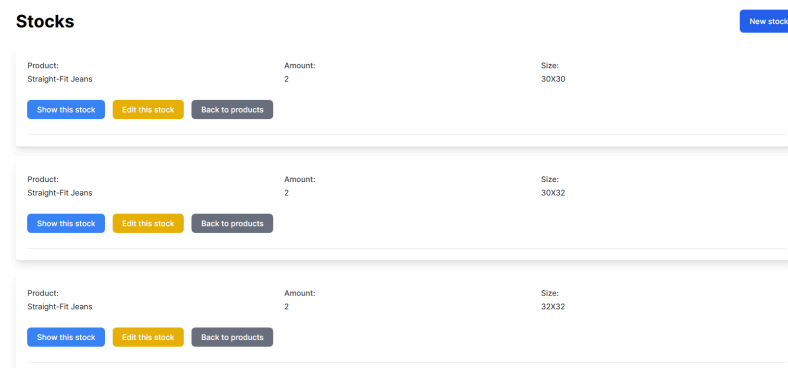
**Editing product**



The form for editing a product includes a product image placeholder with a 'Remove' checkbox. Below is a section for 'Upload New Product Images' with a 'Choose Files' button and 'No file chosen' text. The form contains input fields for 'Name' (filled with 'Straight-Fit Jeans'), 'Description' (filled with 'Super Cool'), and 'Price' (filled with '80'). A 'Select Category' dropdown is set to 'Pants'. An 'Active' checkbox is checked. At the bottom, there are three buttons: 'Update Product' (blue), 'Show this product' (light gray), and 'Back to products' (light gray).

- Allows users to edit all attributes related to product (Price, Image, Description, Name, Product Stock).

○ **Fig 14: Admin Stock Page**






The Admin Stock Page features a table with three rows of stock data for 'Straight-Fit Jeans'. Each row displays the product name, amount (2), and size (30X30, 30X32, and 32X32). Each row has three buttons: 'Show this stock' (blue), 'Edit this stock' (yellow), and 'Back to products' (gray). A 'New stock' button is located in the top right corner.

Product	Amount	Size	Actions
Straight-Fit Jeans	2	30X30	<a href="#">Show this stock</a> <a href="#">Edit this stock</a> <a href="#">Back to products</a>
Straight-Fit Jeans	2	30X32	<a href="#">Show this stock</a> <a href="#">Edit this stock</a> <a href="#">Back to products</a>
Straight-Fit Jeans	2	32X32	<a href="#">Show this stock</a> <a href="#">Edit this stock</a> <a href="#">Back to products</a>

- Displays different stocks for that particular product and all of stocks attributes (Product, Size, Amount)
- Sellers can edit stock (delete, update).
- Sellers can create new stock for that product.

○ **Fig 15: Admin Categories Page**

Categories			New category
	Name	Description	
	Shoes	Quality Shoes	
	Shirts	Quality Shirts	
	Pants	Quality Pants	

- Displays all admin categories and its related attributes (image, name, description)
- Ability to create new categories.

○ **Fig 16: Edit Admin Categories Page**

**Editing category**

Name

Description

Image

Choose File No file chosen

Update CategoryShow this categoryBack to categories

- Displays category attributes.
- Ability to edit categories (update, destroy).

○ **Fig 17: Admin Orders Page**

Orders				New order
Unfulfilled Orders				
Order ID	Customer Email	Address	Total	
18	malik@example.com	12 North York Street Mechanicsburg, PA 17055	\$45.00	
12	malik@example.com	12 North Street West Haverstraw, NY 10993	\$45.00	
18	malik@example.com	12 North York Street Wheeling, WV 26003	\$180.00	
18	malik@example.com	12 North York Street Mechanicsburg, PA 17055	\$45.00	
20	Jamie@example.com	12 North Avenue Staten Island, NY 10302	\$45.00	
Fulfilled Orders				
Order ID	Customer Email	Address	Total	
15	malik@example.com	12 North York Road Hatboro, PA 19040	\$80.00	

- Displays all orders places and its attributes
- Ability to create a new order

○ **Fig 18: Admin Order Show Page**

- Displays a specific order and all its related attributes
- Ability to edit order (update, destroy)

---

## Test Plan

- **Unit Testing:**

Tests have been developed for individual models (e.g., **Product**, **Cart**, **Order**) to ensure proper functionality of core features such as adding items to the cart, processing payments, and managing inventory.

- **Integration Testing:**

Integration tests have been performed to verify that the product catalog, shopping cart, and checkout process work seamlessly together.

- **Technical Details:** Integration tests cover user behavior scenarios, ensuring that adding products to the cart updates session data and proceeds to Stripe checkout correctly.

- **UI/UX Testing:**

The platform's user interface has been tested for responsiveness, ensuring it works smoothly on different devices and screen sizes.

- **Security Testing:**

Security testing focused on the protection of sensitive data, including implementing secure user authentication and ensuring that payment data is handled through Stripe's secure channels.

---

## Test Results

- **Product Catalog and Shopping Cart:**

The product catalog displays items accurately, and users can add/remove items from the shopping cart without issues.

- **Checkout and Payment Processing:**

The checkout process functions correctly, and payments are securely

processed through the Stripe API.

- **Admin Side:**

The admin can manage products and view orders, with the interface being intuitive and easy to navigate.

---

## Challenges Overcome

- **Design Flexibility:**

One challenge was ensuring that the platform, while initially focused on small clothing brands, could be easily adapted to other types of sellers. This was achieved by creating a modular and scalable design.

- **Stripe Integration:**

The process of integrating Stripe's API and ensuring secure payment processing was challenging, but it was resolved by carefully setting up error handling and ensuring proper API configurations.

- **Responsive Design:**

Ensuring the platform was fully responsive for both customers and future sellers required extensive testing and debugging, but it was successfully implemented using Tailwind CSS.

---

## Future Enhancements

- **Multi-Seller Functionality:**

In future phases, the platform will allow multiple sellers to create their own shops, upload products, and manage their inventory, transforming the platform into a multi-vendor marketplace.

- **Mobile App:**

A mobile app could be developed to offer a more convenient shopping

experience for customers and a seller interface for managing stores on the go.

---

### **Defense Presentation Slides**

The defense presentation will include:

- Overview of the project goals and objectives
- Live demo of the shopping experience for customers and the admin seller functionality
- Discussion on how the platform design supports any seller and can be scaled for future sellers
- Challenges faced and how they were addressed
- Plans for future enhancements to expand the platform's capabilities
- Q&A session