ITMGT 45.03 - YZW | HandaCraftPH | ALZATE, BOÑAGA, HILARIO, PAMPLONA | [GC2]
Business: HandaCraftPH

Requirements and Deliverables

1. Platform Architecture Plan
    - System Diagram
        - Front-End → HTML (structure), CSS (visuals), and JS (interactivity); Back-end → Django; Database → SQLite; External APIs / services → Twilio, Easypost API, Quotable.io
    - Tech Stack Justification
        - The HTML, CSS, JS on the front-end ensure fast load times, responsive design, and broad compatibility, making it easy to customize product pages and supports responsive design across devices. The back-end is built with **Django**, a Python-based framework that helps manage user login, cart, and more. Django is secure, scalable, and comes with built-in tools that speed up development and keep the code organized. **SQLite** as the database because it's lightweight, fast, and easy to set up. It stores all data in a single file, making it ideal for small to medium size projects. SQLite is great for quick development and testing, works well with Django and can handle user profiles and other information efficiently. For the APIs, **Twilio** is used to send SMS notifications to users, such as order updates, improves communication, and keeps users informed. The **Easypost API** is used for shipping tasks like tracking packages. It supports many carriers, making delivery smoother and more flexible for users. **Quotable.io** displays random quotes on the site, enhancing the user experience by adding thoughtful messages that keep the content engaging. It is free and easy to integrate.

    - Hosting & Deployment
        - Render is used to host and deploy all the components because it's easy to set up, supports the Django framework, and automates deployment from GitHub. It offers free plans, making it ideal for this project.

Components	Platform
Front-end	Render
Back-end	Render
Database	Render

    - Data Flow Description
        1. User registers. The user fills out a form. JS checks the inputs. Django saves the new user and tier profile info, then logs them in.
        2. User log in. The user enters their username and password. Django checks if it matches and logs them in by creating a session.
        3. Open the website. The browser asks Django for the homepage. Django sends back a page with product lists, categories, and user login information.
        4. Browses products. When the user clicks a category, Django gets the matching products from the database.
        5. Views a product. Clicking a product shows full details like images, seller info, and description. Django fetches this from the database and sends it to the browser.
        6. User adds to cart. Clicking "Add to Cart" sends the product ID and quantity to Django. Who then saves it in the CartItem table and updates the cart count.

7. User views cart. Django gets all the items in the user's cart, calculates totals, and sends the info to the cart page.
8. User checks out. Django gets the cart items and user's saved address. The checkout page shows items, total price, and address form.
9. Order confirmation. Django sends an email and SMS (using Twilio) with order details. The user sees a confirmation page.
10. Seller gets notified. Django tells the seller someone bought their product, by email or dashboard alert.
11. Seller uploads product. The seller fills out a form with product info and images. Django saves it, and the product appears on the site.

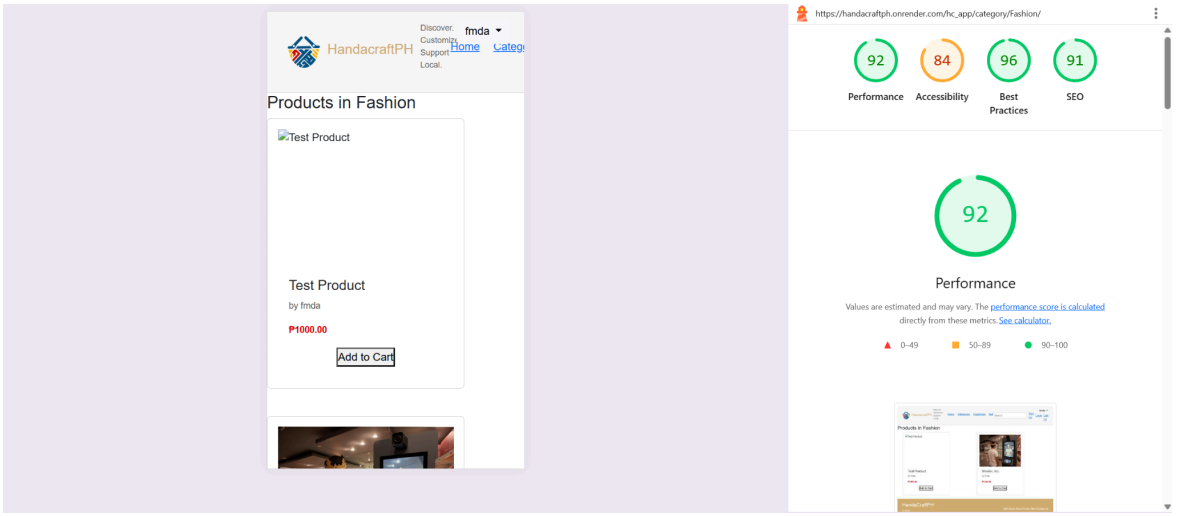
2. Core Transaction Flow

First, for **product listing**, listings are separated by their respective category. For this MVP, only the seller name and price is shown currently. Along with the product listing card is an add to cart button with each, and after it is clicked the item count beside the cart icon will update and the product will be added to the user's cart. Second, for the **add-to-cart and cart view**, when the cart icon is clicked, it shows a view of the cart of the user. They can update the quantity of the product or delete it from their cart, they can also see the sub-total of their orders (without the shipping fee). There is also a 'proceed to checkout' button which will lead to the checkout page. **Third**, for the checkout page, the user is required to type their address details and contact number, the products are shown (specifically, the product name, price of one unit, quantity, and subtotal of each item), the shipping option is also included (for now set to Php 100 given the current shipping API limitations), there are buttons that shows payment methods that user can select, grand total is also shown on the page (this is the subtotal for the whole order + shipping fee), a place order button at the lower right corner which leads to the order confirmation page that shows (Order ID, Date, Payment method selected, Total, Items in the order, Estimated delivery, and View my orders button and back to home button). In the my orders button (or through the 'my orders' dropdown), the user can see their orders along with its status, date order was placed, shipping cost, total, items (with quantity), and seller address, and the order can be deleted too.

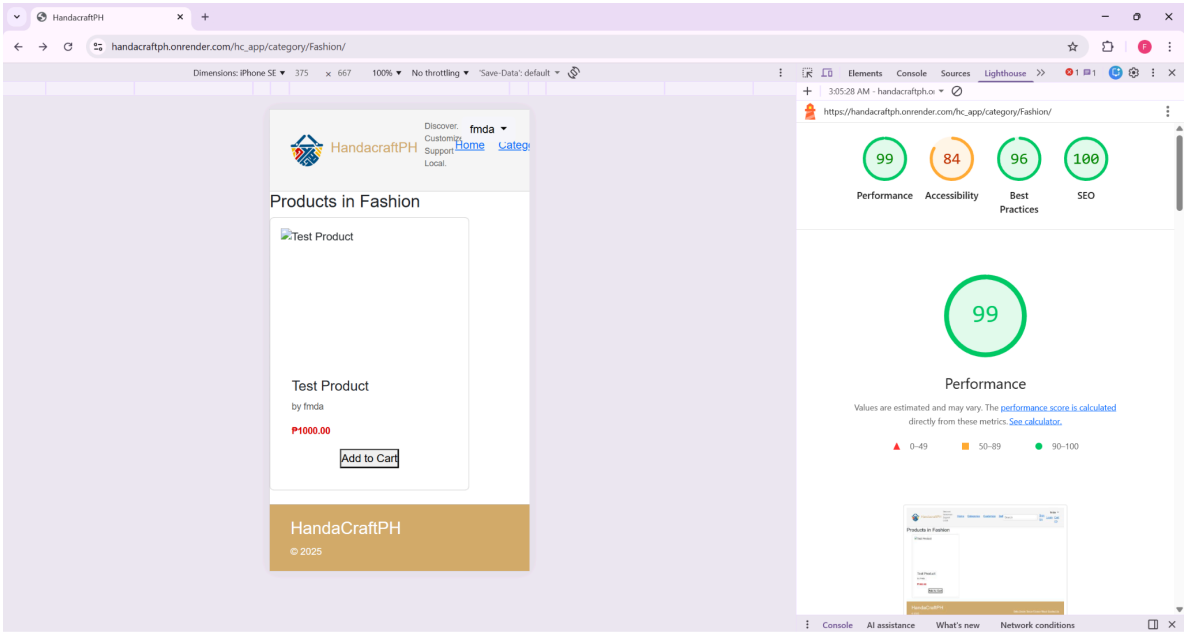
3. Performance, SEO, or Discoverability Optimization

The base.html and category_products.html has SEO and discoverability requirements as it includes meta description, meta keywords, and open graph tags. The logo image is lazy-loaded, and the Bootstrap JavaScript is loaded with defer for it to not block rendering.

Before Performance, SEO, or Discoverability Optimization for category_products.html



After Performance, SEO, or Discoverability Optimization for category_products.html



4. Live API Integration (IA4)
    - One of the main APIs used in the system is the **Twilio API, specifically its WhatsApp server**. This API plays a crucial role in the core functionality of user registration. During the registration process, users are required to provide a contact number, which is then retrieved and verified through the Twilio API to confirm their registration. Another API integrated into the system is **Quotable.io.** While it is not directly tied to the core features, it enhances the overall user experience and visual appeal of the login and registration pages. It provides random quotes that are fetched and displayed within the application, contributing to the website's aesthetic value and creating a more engaging user interface. The EasyPost API is used to create and manage shipping labels and track orders for products purchased through the platform.

References

Farar, J., & Farar, J. (2025, August 21). *Ecommerce shipping API and integration Solution*. EasyPost. https://www.easypost.com/blog/2024-11-20-shipping-api-for-ecommerce/