**SPECTRA STYLE CLOTHING MANAGEMENT SYSTEM**

A Collaboration Project

Presented to the Faculty of the College of Computer Studies

Filamer Christian University

In Partial Fulfillment of the Requirements for the Subjects:

ITE2 – IT Elective 2

ITE5 – Integrative Programming and Technologies

PT2 – Professional Track 2

Group #9

Gonzales, Marc Jason G. – BSIT 2C

Laher, Allen Jay B. – BSIT 2B

Vega, Erika Shigella L. – BSIT 2A

Villagracia, Manuel Gabriel R. – BSIT 2A

Yap, Fileteo II L. – BSIT 2B

June 16, 2025

**COMPILED RATING SHEETS**

This section contains the compiled group and individual rating sheets. It includes the final, completed rating sheets as provided by the instructors or the averaged scores from peer evaluations. This section serves as a direct input for the grading process.

**Compiled Rating Summary**

Group Performance Score: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Overall Output of Management System Score: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Project Paper Quality Score: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Individual Member Contribution Scores:**

Gonzales, Marc Jason Garbo: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Laher, Allen Jay Bibit: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Vega, Erika Shigella Lozada: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Villagracia, Manuel Gabriel Roxas: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Yap, Fileteo II Llorico: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**EXECUTIVE SUMMARY**

The Spectra Style Clothing Point of Sale (POS) System is a comprehensive web-based application developed to transform clothing retail operations by replacing disconnected manual processes with an integrated digital solution. The system addresses critical challenges faced by clothing retailers including inefficient checkout processes, inaccurate inventory tracking across product variants, disjointed order management, and limited business analytics capabilities.

The primary purpose of the project was to develop a secure, scalable, and user-friendly POS application that streamlines retail workflows while providing accurate real-time data for business decision-making. Key features include specialized clothing inventory management with size/color variant tracking, efficient sales transaction processing, comprehensive order status workflow, customer relationship management, real-time analytics dashboard, and role-based security controls.

This project successfully delivered a complete POS solution built using modern web technologies including Python Django framework for backend development, MySQL for database management, JavaScript for interactive elements, and Tailwind CSS for responsive design. The implementation follows best practices in software architecture including the Model-View-Template pattern, secure authentication mechanisms, and comprehensive data validation.

The system directly solves the problem of fragmented retail operations by integrating all core business functions in a single platform – from initial product creation through inventory management, point of sale transactions, order processing, to business analytics. This integration eliminates data silos, reduces errors, increases operational efficiency, and provides better visibility into business performance.

Main outcomes include a fully functional POS system with documented workflows, clearly defined KPIs for measuring operational efficiency, complete technical documentation, and a product roadmap for future enhancements. The project demonstrates the successful application of software engineering principles to create a practical business solution that addresses specific industry needs in the clothing retail sector.

**PROJECT DELIVERABLES CHECKLIST**

Process Mapping: [✓] Complete - Comprehensive mapping of POS workflows including sales transaction processing, inventory management, order fulfillment, and reporting (Refer to Chapter 4, Section 11.5)

Key Performance Indicators (KPIs): [✓] Complete - Defined and implemented metrics including Order Fulfillment Efficiency, Inventory Accuracy Rate, and System Performance Score to measure POS system effectiveness (Refer to Chapter 4, Section 11.6)

Service Level Agreement (SLA): [✓] Complete - Established performance standards and support parameters for the POS system's operation and maintenance (Refer to Chapter 4, Section 11.7)

Emerging Technology Integration: [✓] Complete - Analysis of AI, machine learning, and predictive analytics applications for enhancing POS functionality (Refer to Chapter 4, Section 11.8)

Project Management using Asana: [✓] Complete - Systematic tracking of development tasks, milestones, and team member responsibilities (Access granted to Ma'am Jen, Sir Villy, Sir Joven, Sir Mark)

Individual Member Contributions: [✓] Documented - Detailed accounting of each team member's role and contributions to the POS system development (Refer to Chapter 5)

**PROJECT REQUIREMENTS CHECKLIST**

Web-Based Point of Sale (POS) System: [✓] Designed and Developed - Fully functional web-based POS system with integrated inventory management, sales processing, and business analytics

Needs of Mock Clients (Ms. Escalada, Mr. Alovera, Mr. Delsocora): [✓] Addressed - System specifically designed to solve clothing retail challenges including product variant tracking, transaction processing, and sales reporting

Professionally Presented: [✓] Ensured - Clean, intuitive user interface with consistent design language and responsive layout

Well-Documented: [✓] Ensured - Comprehensive documentation including process maps, technical specifications, user guides, and API references

Backend Programming Language: Python: [✓] Used - Django 4.2.8 framework with Python 3.10+ for all backend functionality

Database: MySQL: [✓] Used - MySQL 8.0 for data storage with proper schema design and relational integrity

Core Functions: Create, Read, Update, Delete (CRUD): [✓] Implemented - Complete CRUD operations for all major entities including products, variants, customers, orders, and suppliers

Core Functions: User Login & Logout: [✓] Implemented - Secure authentication system with role-based access control

Frontend UI and Interactivity: JavaScript: [✓] Used - Dynamic features including real-time stock validation, order calculations, and interactive dashboards

Frontend UI Design: Tailwind CSS: [✓] Used - Responsive, modern UI built with Tailwind CSS 3.4.17 utility framework

**Table of Contents**

**Preliminary Pages**

* Title Page .............................................................................................................................i
* Grading Sheet .....................................................................................................................ii

**Main Content**

* Executive Summary .........................................................................................................1-2
* Project Deliverables Checklist ............................................................................................3
* Project Requirements Checklist .......................................................................................4-5
* Table of Contents ......................................................................................................6-7-8-9

**Chapter 1: Introduction .............................................................................................................10**

* 1.1 Project Context ............................................................................................................10
* 1.2 Problem Statement ......................................................................................................10
* 1.3 Importance of the Project ............................................................................................11
* 1.4 Project Objectives ..................................................................................................11-12
* 1.5 Significance of the Study .......................................................................................12-13
* 1.6 Scope and Limitations ......................................................................................13-14-15

**Chapter 2: Overview of the Study ...............................................................................................7**

* 2.1 System Description .......................................................................................................7
* 2.2 Core Functionalities ......................................................................................................7

**Chapter 3: System Requirements ................................................................................................8**

* 3.1 Functional Requirements ..............................................................................................8
* 3.2 Non-Functional Requirements ....................................................................................12
* 3.3 Hardware Requirements ..............................................................................................15
* 3.4 Software Requirements ...............................................................................................16

**Chapter 4: System Design & Development ................................................................................9**

* [4.1 Back-end Development (Sir Joven's Subject)](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_1) .............................................................17
  + [4.1.1 Programming Language: Python](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_1_1) .................................................................17
  + [4.1.2 Database: MySQL](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_1_2) ........................................................................................17
  + [4.1.3 Core Functions Implemented: CRUD](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_1_3) ..........................................................18
  + [4.1.4 Core Functions Implemented: User Login & Logout](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_1_4) ..................................18
  + [4.1.5 Backend Architecture and Design Choices](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_1_5) ..................................................18
* [4.2 Front-end Development (Sir Villy's Subject)](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_2) ..............................................................19
  + [4.2.1 UI and Interactivity (JavaScript)](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_2_1) ..................................................................19
  + [4.2.2 UI Design (Tailwind CSS)](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_2_2) ...........................................................................19
  + [4.2.3 Frontend Architecture and Design Choices](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_2_3) .................................................20
* [4.3 Development Tools and Technologies](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_3) ........................................................................20
  + [4.3.1 Languages](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_3_1) ....................................................................................................20
  + [4.3.2 Frameworks](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_3_2) ..................................................................................................21
  + [4.3.3 Databases](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_3_3) .....................................................................................................21
  + [4.3.4 Libraries](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_3_4) .......................................................................................................21
  + [4.3.5 Other Tools](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_3_5) ...................................................................................................21
* [4.4 Testing and Quality Assurance](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_4) ....................................................................................22
  + [4.4.1 Back-end Testing Strategies](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_4_1) .........................................................................22
  + [4.4.2 Front-end Testing Strategies](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_4_2) ........................................................................22
  + [4.4.3 Testing Results and Bug Fixes](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_4_3) .....................................................................23
* [4.5 Process Mapping](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_5) .........................................................................................................23
  + [4.5.1 System's Core Processes Visualization](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_5_1) ........................................................23
  + [4.5.2 Inputs, Outputs, Activities, and Decision Points](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_5_2) ..........................................24
* [4.6 Key Performance Indicators (KPIs)](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_6) ............................................................................24
* [4.7 Service Level Agreement (SLA)](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_7) .................................................................................26
  + [4.7.1 Proposed SLA Outline](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_7_1) .................................................................................26
  + [4.7.2 Expected Service Level Metrics](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_7_2) ..................................................................26
* [4.8 Emerging Technology Integration](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_8) ..............................................................................27
  + [4.8.1 Identified Emerging Technology](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_8_1) .................................................................27
  + [4.8.2 Function within the System](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_8_2) .........................................................................27
  + [4.8.3 Benefits and Justification](https://kinuris.github.io/spectra_documentation/project_paper.html#section11_8_3) .............................................................................27

**Chapter 5: Individual Member Contributions .........................................................................10**

* 5.1 Member 1: [Full Name] ..............................................................................................28
* 5.2 Member 2: [Full Name] ..............................................................................................28
* 5.3 Member 3: [Full Name] ..............................................................................................29
* 5.4 Member 4: [Full Name] ..............................................................................................29

**Chapter 6: Conclusion & Recommendations ...........................................................................11**

* 6.1 Project Outcomes ........................................................................................................30
* 6.2 Future Recommendations ...........................................................................................30

**References / Bibliography ..........................................................................................................12**

**Appendices ...................................................................................................................................13**

* Appendix A: List of Tables .............................................................................................A-1
* Appendix B: List of Figures ...........................................................................................B-1
* Appendix C: Diagrams ...................................................................................................C-1
* Appendix D: Technical Specifications ...........................................................................D-1
* Appendix E: Key Screenshots of the System ..................................................................E-1
* Appendix F: Individual Member Contribution Sheets ....................................................F-1

**CHAPTER 1: INTRODUCTION**

**1.1 Project Context**

The Spectra Clothing Management System operates within the fashion retail industry, specifically targeting small to medium-sized clothing boutiques. In today's competitive retail environment, clothing boutiques face significant challenges in managing their product catalogs, tracking inventory across multiple variants (sizes and colors), processing customer orders efficiently, and maintaining relationships with suppliers. Traditional manual processes, spreadsheet-based tracking, and disconnected systems lead to inventory discrepancies, delayed order processing, poor visibility into sales performance, and difficulty in making data-driven business decisions. Our web-based management system aims to address these operational inefficiencies by providing an integrated platform that centralizes all business operations.

**1.2 Problem Statement**

Clothing boutiques owned by Ms. Jeanny Escalada, Mr. Joven Alovera, and Mr. Villy Delsocora currently rely on manual processes such as spreadsheets and handwritten records to manage inventory, orders, and supplier information. This results in frequent stock discrepancies, delayed order processing, limited business visibility, and weak coordination with vendors. Additionally, the absence of role-based access control compromises data integrity and staff accountability. These inefficiencies hinder operational performance and limit the boutiques' ability to scale. A centralized, web-based management system is essential to streamline workflows, improve accuracy, and support data-driven decision-making.

**1.3 Importance of the Project**

This system is crucial for modern retail operations as it transforms manual, error-prone processes into automated, reliable workflows. The significance lies in its ability to reduce operational costs, minimize human errors, improve customer satisfaction through faster order processing, and provide real-time business insights. For boutique owners and staff, the system eliminates the complexity of managing multiple spreadsheets and disparate systems, allowing them to focus on customer service and business growth. The role-based access control ensures that different staff members (Admin, Stock Manager, Sales Staff) can efficiently perform their specific duties while maintaining data security and operational oversight.

**1.4 Project Objectives**

The project aims to achieve the following specific, measurable goals directly tied to addressing operational challenges in clothing boutique management:

* To develop a secure and intuitive web-based system that centralizes product catalog management, enabling efficient creation, updating, and organization of clothing items with their variants.
* To implement a comprehensive inventory tracking system that monitors stock levels across size and color combinations, automatically generates low-stock alerts, and maintains detailed audit trails of all inventory movements.
* To create an efficient order management module that streamlines customer order processing, validates stock availability, and tracks order status from creation to completion.
* To design a supplier management system that maintains vendor information and facilitates product sourcing coordination.
* To build analytical and reporting capabilities that provide actionable business insights through sales reports, performance dashboards, and key performance indicator (KPI) monitoring.
* To ensure role-based access control that allows appropriate system access for different user types while maintaining data security and operational efficiency.

**1.5 Significance of the Study**

**Mock Clients**

For Mock Clients (Ms. Escalada, Mr. Alovera, Mr. Delsocora): This system transforms their clothing retail operations by replacing disconnected, manual processes with an integrated digital solution. It enables them to process sales transactions efficiently, maintain accurate inventory records across product variants, build a valuable customer database, and make data-driven business decisions based on real-time analytics. The POS system directly addresses their operational pain points while providing a platform that can scale with their business growth, ultimately increasing profitability through improved operational efficiency and enhanced customer service capabilities.

**Students / Developers**

For the student developers, this project has served as a real-world application of academic learning. We gained hands-on experience in:

* Full-stack development using **Python (Django)** for backend and **Tailwind CSS**/**JavaScript** for frontend.
* Database management using **MySQL**.
* Implementing core programming concepts such as CRUD operations, user authentication, and data visualization.
* Project planning and task delegation through **Asana**, fostering collaboration and time management skills.
* Understanding real client needs and translating them into technical solutions.

**Community / Industry**

In a broader sense, the system provides a scalable model for other local clothing brands and small businesses looking to digitalize their operations. It also showcases how IT students can create meaningful, impactful systems that support entrepreneurship, especially in resource-limited communities. This aligns with the growing need for accessible digital tools in the fashion retail sector.

**1.6 Scope and Limitations**

**Scope**

The Spectra Style Clothing Management System encompasses a comprehensive Point of Sale solution with the following core modules and functionalities:

* **Product Management:** Creation and maintenance of product catalog with detailed product information, SKU management, category organization, price management, and product image handling.
* **Inventory Management:** Product variant tracking (size/color combinations), stock level monitoring, inventory adjustments (incoming, outgoing, manual), low stock alerts, and inventory audit trails.
* **Order Management:** Customer selection/creation during checkout, product selection with real-time stock validation, order creation, comprehensive order status workflow (PENDING → PROCESSING → AWAITING\_PAYMENT → SHIPPED → DELIVERED), and order history tracking.
* **Customer Management:** Customer records with contact details, purchase history, and customer lookup during sales transactions.
* **Supplier Management:** Supplier directory with contact information, association of products with suppliers, and supplier catalog management.
* **Analytics & Reporting:** Sales performance metrics, inventory status reports, product performance analysis, category analysis, and trend visualization.
* **User Management:** Role-based access control (Admin, Stock Manager, Sales Staff), user authentication, and security audit logging.
* **Dashboard:** Real-time KPI display, activity feed, quick action links, and system alerts.

**Limitations:** The current implementation of the Spectra Style Clothing POS system acknowledges the following constraints:

* The system does not include real-time payment gateway integration for processing credit cards or digital payments, which would be required in a production environment.
* While the system tracks orders through a complete workflow, it does not include integrated shipping label generation or third-party logistics integration.
* The analytics capabilities focus on descriptive analytics (what has happened) rather than predictive analytics (what will happen) or prescriptive analytics (what should be done).
* The system is designed as a web application but does not include a dedicated mobile application for on-floor sales processing or inventory management.
* There is no integrated employee time tracking or commission calculation functionality for sales staff performance management.
* The system does not include multi-store inventory management for businesses with multiple retail locations.
* Advanced loyalty program features and promotional campaign management are not implemented in the current version.
* The system is designed for clothing retail specifically and may require adaptation for use in other retail sectors with different inventory and sales models.

**CHAPTER 2: OVERVIEW OF THE STUDY**

**2.1 System Description**

The Spectra Style Clothing Management System is a comprehensive web-based Point of Sale (POS) application designed specifically for clothing retail operations. Built on the Django framework with a MySQL database backend, the system provides an integrated platform for managing the complete retail workflow from inventory management to sales processing and business analytics.

The primary purpose of the system is to streamline retail operations by providing a single platform that handles product management, inventory control with variant tracking, sales transactions processing, customer relationship management, and business analytics. This integration eliminates the need for multiple disconnected systems and manual processes that are prone to errors and inefficiencies.

The system architecture follows Django's Model-View-Template (MVT) pattern and consists of several interconnected modules:

* **Products Module**: Manages the product catalog including basic product information, categories, images, and pricing.
* **Inventory Module:** Handles product variants (size/color combinations), stock levels, inventory adjustments, and provides low stock alerts.
* **Orders Module:** Processes sales transactions, manages the order status workflow, and maintains order history.
* **Suppliers Module:** Maintains supplier information and associations between products and suppliers.
* **Accounts Module:** Handles user authentication, authorization, and role-based access control.
* **Dashboard Module:**Provides real-time analytics, KPIs, and system activity monitoring.

At a high level, the system allows staff to efficiently process sales transactions at the point of sale, maintain accurate inventory records across product variants, track customer information and purchase history, monitor business performance through analytics, and make data-driven decisions. The workflow integration ensures that sales transactions automatically update inventory levels, sales data flows into analytics, and inventory alerts prompt timely reordering decisions.

**2.2 Core Functionalities**

The Spectra Style Clothing Point of Sale system provides the following essential features and capabilities:

* **User Authentication and Authorization:** Secure login system with role-based access control that determines user permissions based on assigned roles (Admin, Stock Manager, Sales Staff). Django's authentication system provides password hashing, session management, and secure logout functionality.
* **Product Management:** Comprehensive product catalog management including creation, update, and deletion of products with detailed attributes (name, descriptions, cost/selling prices). Products can be organized by categories and associated with suppliers. The system supports product image uploads and management.
* **Variant Management:** Specialized functionality for clothing retail that enables tracking product variations by size and color combinations. Each variant maintains its own inventory levels while being associated with the parent product.
* **Inventory Control:** Real-time tracking of stock levels for each product variant with support for different types of inventory adjustments (incoming, outgoing, manual corrections). The system provides low stock alerts based on configurable reorder levels and maintains a complete audit trail of inventory changes.
* **Point of Sale Transaction Processing:** Efficient sales workflow allowing staff to select products, specify quantities, assign transactions to existing or new customers, and complete orders. The system validates stock availability in real-time during checkout to prevent overselling.
* **Customer Management:** Creation and maintenance of customer records with contact information and address details. The system tracks customer purchase history and allows quick customer lookup during checkout.
* **Order Management:** Comprehensive order processing with status workflow management (PENDING → PROCESSING → AWAITING\_PAYMENT → SHIPPED → DELIVERED). Each status change is tracked with timestamps and user attribution. Orders can be filtered and searched by various criteria.
* **Supplier Management:** Supplier directory with contact details and the ability to associate products with specific suppliers for streamlined procurement.
* **Analytics and Reporting:** Built-in analytics dashboard providing key performance metrics including sales totals, top-selling products, category performance, and inventory status. The system supports data visualization through charts and graphs to identify trends and patterns.
* **Dashboard Overview:** Real-time executive dashboard displaying current day's sales, pending orders, low stock alerts, and recent activities. The dashboard provides quick access to frequently used functions.
* **Key Performance Indicators (KPIs):** Automated tracking of critical business metrics including Order Fulfillment Efficiency, Inventory Accuracy Rate, and System Performance Score.

**CHAPTER 3: SYSTEM REQUIREMENTS**

**3.1 Functional Requirements**

The Spectra Style Clothing Point of Sale system fulfills the following functional requirements:

|  |  |  |
| --- | --- | --- |
| Requirement ID | Description | Module |
| FR-01 | The system shall require user authentication via username and password before accessing any functionality. | Authentication |
| FR-02 | The system shall enforce role-based access control based on Admin, Stock Manager, and Sales Staff. | Authentication |
| FR-03 | The system shall allow CRUD operations for products, customers, suppliers, and orders. | Product, Customer, Supplier, Order Management |
| FR-04 | The system shall track product variants based on size and color combinations. | Inventory Management |
| FR-05 | The system shall validate stock availability during order processing. | Point of Sale |
| FR-06 | The system shall provide low-stock alerts based on configurable thresholds. | Inventory Management |
| FR-07 | The system shall provide dashboards and reports with sales trends and KPIs. | Analytics & Reporting |
| FR-08 | The system shall allow association of orders with customer records. | Customer Management |

**3.2 Non-Functional Requirements**

The Spectra Style Clothing Point of Sale system meets the following non-functional requirements:

|  |  |
| --- | --- |
| Category | Requirement Description |
| Performance | The system shall load pages in under 3 seconds under normal conditions. |
| Performance | The system shall support at least 20 simultaneous users. |
| Security | Passwords must be stored using secure hashing algorithms. |
| Security | The system shall protect against CSRF, XSS, and SQL injection. |
| Usability | The system shall be usable with minimal training. |
| Usability | Responsive design for desktop, tablet, and mobile devices. |
| Scalability | The system shall support up to 1 million transaction records. |
| Maintainability | Code shall be documented and follow modular structure. |
| Portability | The system shall run on multiple browsers and OS platforms. |

**3.3 Hardware Requirements**

The Spectra Style Clothing Point of Sale system requires the following minimum hardware specifications:

|  |  |  |
| --- | --- | --- |
| Component | Specification | Notes |
| Server - Processor | Multi-core CPU (4+ cores) | 2.0 GHz or higher clock speed |
| Server - RAM | 8 GB minimum | 16 GB recommended |
| Server - Storage | 100 GB SSD | With backup solutions |
| Server - Network | Gigabit Ethernet |  |
| Server - Backup Power | UPS | To prevent data loss |
| Client - Processor | Dual-core CPU | 1.8 GHz or higher |
| Client - RAM | 4 GB minimum |  |
| Client - Storage | 20 GB available |  |
| Client - Display | 1366x768 resolution | Or higher |
| Client - Network | Wired/Wireless | Reliable connection |
| Client - Input Devices | Standard keyboard and mouse |  |
| POS - Barcode Scanner | USB compatible |  |
| POS - Receipt Printer | Thermal |  |
| POS - Cash Drawer | Standard with printer trigger |  |
| POS - Card Reader | Payment terminal | For future use |
| POS - Customer Display | Optional | Secondary display |

**3.4 Software Requirements**

The following software stack is required to run the Spectra Style Clothing Point of Sale system:

|  |  |  |
| --- | --- | --- |
| Category | Component | Version/Notes |
| Server Environment | Operating System | Linux Ubuntu 22.04 LTS / Windows Server 2019+ / macOS 12+ |
| Server Environment | Web Server | Nginx 1.18+ / Apache 2.4+ |
| Server Environment | WSGI Server | Gunicorn 20.1.0+ / uWSGI 2.0.19+ |
| Server Environment | Database | MySQL 8.0+ / MariaDB 10.6+ |
| Server Environment | Programming Language | Python 3.10+ |
| Server Environment | Framework | Django 5.2.1 |
| Server Environment | Python Libraries | mysqlclient 2.2.0, Pillow 10.1.0, python-dotenv 1.0.0 |
| Server Environment | Node.js | v20.0+ |
| Server Environment | NPM Packages | Tailwind CSS 3.4.17, Autoprefixer 10.4.21, PostCSS 8.5.3 |
| Client Environment | Web Browsers | Chrome 96+, Firefox 95+, Edge 96+, Safari 15+ |
| Client Environment | Operating System | Windows 10+, macOS 12+, modern Linux distro |
| Client Environment | PDF Viewer | Required for viewing reports/invoices |
| Development Environment | IDE | VS Code with Python/JS extensions |
| Development Environment | Version Control | Git 2.35+ |
| Development Environment | Database Tool | MySQL CLI / MySQL Workbench |
| Development Environment | Project Management Tool | Asana |
| Development Environment | Virtual Environment | Python venv or virtualenv |

**CHAPTER 4: SYSTEM DESIGN & DEVELOPMENT**

**4.1 Back-end Development (Sir Joven's Subject)**

**4.1.1 Programming Language: Python with Django Framework**

Python was selected as the primary backend programming language due to its readability, extensive libraries, and strong community support. The Django framework was chosen for its robust features, including an Object-Relational Mapper (ORM), built-in admin panel, and security features, which accelerate development and ensure a scalable and maintainable application. Key libraries used include Django for the core framework, mysqlclient for MySQL database connectivity, Pillow for image handling, and python-dotenv for managing environment variables.

**4.1.2 Database: MySQL**

MySQL is utilized as the relational database management system (RDBMS) for its reliability, performance, and widespread adoption. Django's ORM facilitates seamless interaction with the MySQL database, abstracting SQL queries and allowing developers to work with Python objects. The database schema is designed to support core functionalities such as user management, product catalog, inventory, orders, and supplier information. (Refer to ERD in Appendix 17.1 for detailed schema).

**4.1.3 Core Functions Implemented: Create, Read, Update, Delete (CRUD)**

The system implements comprehensive CRUD operations for all major entities. For example, administrators can create new product listings, users can read product details, inventory levels can be updated, and outdated supplier records can be deleted. These operations are handled through Django views, which process HTTP requests, interact with models (database tables), and return appropriate responses. Specific API endpoints are defined in Django's urls.py to manage these interactions.

**4.1.4 Core Functions Implemented: User Login & Logout**

User authentication is managed using Django's built-in authentication system. This includes secure password hashing, user registration, login, and logout functionalities. Session management is handled by Django to maintain user state across requests, ensuring a secure and personalized experience. Role-based access control can be implemented to differentiate user permissions.

**4.1.5 Backend Architecture and Design Choices**

The backend follows a Model-View-Template (MVT) architectural pattern, which is Django's variation of the Model-View-Controller (MVC) pattern. This promotes separation of concerns, making the application modular and easier to maintain. The system is designed as a monolithic application, suitable for the project's scope. RESTful principles are applied for API design where applicable, ensuring predictable and standard-based communication with the frontend or other services. Security considerations include protection against common web vulnerabilities (e.g., CSRF, XSS) provided by Django, and secure handling of sensitive data.

**4.2 Front-end Development (Sir Villy's Subject)**

**4.2.1 UI and Interactivity (JavaScript)**

JavaScript is employed to enhance the user interface with dynamic and interactive elements. This includes client-side form validation to provide immediate feedback to users, asynchronous updates for parts of a page without full reloads (e.g., filtering product lists, updating shopping cart contents), and managing UI events to create a responsive user experience. While a specific complex JavaScript framework (like React or Vue) is not indicated for this project, vanilla JavaScript is used to manipulate the DOM and interact with backend APIs for data retrieval and submission.

**4.2.2 UI Design (Tailwind CSS)**

Tailwind CSS, a utility-first CSS framework, is used for designing the user interface. This approach allows for rapid development and customization of styles directly within the HTML markup. Tailwind CSS facilitates the creation of a responsive design, ensuring the application is usable across various devices and screen sizes. Key design principles applied include a clean and intuitive layout, consistent styling for UI elements, and a focus on user experience. The tailwind.config.js and postcss.config.js files are configured to customize and process the Tailwind CSS build.

**4.2.3 Frontend Architecture and Design Choices**

The frontend architecture is primarily a traditional multi-page application (MPA) where Django templates render HTML pages. JavaScript is used to add interactivity on these pages rather than building a full single-page application (SPA). The frontend communicates with the backend through standard HTTP requests for page navigation and form submissions, and potentially through asynchronous JavaScript (AJAX) calls to Django views that return JSON data for dynamic updates. This approach simplifies development for the given project scope while leveraging Django's templating engine and Tailwind CSS for efficient UI development.

**4.3 Development Tools and Technologies**

**4.3.1 Languages**

* Python (Backend)
* JavaScript (Frontend Interactivity)
* HTML (Frontend Structure)
* SQL (Database Queries)
* CSS (Custom styles and Tailwind CSS framework)

**4.3.2 Frameworks**

* Django 4.2.8 (Python Backend Framework)
* Vanilla JavaScript (No complex frontend framework - DOM manipulation and AJAX)

**4.3.3 Databases**

* MySQL

**4.3.4 Libraries**

* Backend: Django 4.2.8, mysqlclient 2.2.0, Pillow 10.1.0, python-dotenv 1.0.0
* Frontend: Chart.js (for data visualizations), Font Awesome (for icons)
* CSS: Tailwind CSS 3.4.17, Autoprefixer 10.4.21, PostCSS 8.5.3

**4.3.5 Other Tools**

* Integrated Development Environment (IDE): Visual Studio Code
* Version Control System: Git
* Database Management Tool: MySQL Command Line Interface
* Project Management Tool: Asana
* Build Tools: Node.js and npm (for Tailwind CSS compilation)

**4.4 Testing and Quality Assurance**

**4.4.1 Back-End Testing Strategies**

The backend testing approach focused on ensuring reliable CRUD operations, data integrity, and proper authentication mechanisms. **Unit Testing**: Individual Django model methods and utility functions were tested using Django's built-in testing framework, including validation of model constraints, custom methods, and business logic. **Integration Testing**: Database connectivity and ORM operations were verified through test cases that create, read, update, and delete records across all models (Product, ProductVariant, Order, Customer, etc.). **API Endpoint Testing**: All Django views and URL patterns were tested to ensure proper HTTP responses, authentication requirements, and data handling. **Authentication Testing**: Role-based access control was validated by testing user permissions for different roles (Admin, Stock Manager, Sales Staff) across various system functions. **Data Validation Testing**: Form validations, model constraints, and business rules were tested to ensure data integrity, including SKU uniqueness, stock level validations, and order processing logic. (Mention testing frameworks/tools used, e.g., unittest, pytest).

**4.4.2 Front-End Testing Strategies**

Frontend testing ensured a consistent, responsive, and intuitive user experience across different devices and browsers**. UI/UX Testing**: Manual testing of all user interfaces to verify proper layout, navigation flow, and visual consistency using Tailwind CSS components. **Responsiveness Testing**: The system was tested across multiple screen sizes (desktop, tablet, mobile) to ensure responsive design functionality using Tailwind's responsive utilities**. Browser Compatibility Testing**: Cross-browser testing was conducted on Chrome, Firefox, Safari, and Edge to ensure consistent behavior and appearance**. Form Validation Testing**: Client-side and server-side form validations were tested for all input forms including product creation, order processing, and user authentication. **JavaScript Functionality** Testing: Interactive elements, AJAX requests, and DOM manipulations were tested to ensure proper functionality without page refreshes. **User Acceptance Testing (UAT) Simulation:** Role-based testing scenarios were conducted simulating the workflows of each user type (Admin, Stock Manager, Sales Staff) to validate system usability and feature completeness.

**4.4.3 Testing Results and Bug Fixes**

Throughout testing, several issues were identified and addressed to improve system robustness and user experience. Initial database integrity problems related to foreign key constraints during cascade deletions were resolved by refining deletion handling and adding confirmation prompts. Client-side form validation was enhanced to provide immediate feedback, reducing server-side errors. Improvements to authentication flows, such as session management and redirect logic, ensured smoother user transitions after login and logout. Inventory validation algorithms were refined to prevent overselling and maintain accurate stock during concurrent transactions. Visual inconsistencies were corrected by standardizing Tailwind CSS classes, enhancing error message displays, and improving responsiveness on mobile devices. Performance optimizations, including the use of Django’s select\_related and prefetch\_related, helped reduce database query counts and improve page load times. Overall, the testing and debugging process significantly strengthened the system’s reliability, accuracy, and user-friendliness, making it well-suited for managing clothing boutique operations.

**4.5 Process Mapping**

**4.5.1 System's Core Processes Visualization**

The Spectra Style Clothing Management System's core processes have been comprehensively mapped and visualized through detailed process flow diagrams. The key processes include: (1) **Product Management Process** - from supplier information input through product creation, category assignment, image upload, to final catalog availability; (2) **Inventory Management Process** - covering variant creation (size/color combinations), stock assignment, inventory tracking, and automated alert generation; (3) **Order Management Process** - spanning customer information capture, product selection, stock validation, order creation, and fulfillment tracking; (4) **Analytics & Reporting Process** - encompassing data collection, performance analysis, visualization generation, and business insight delivery; (5) **Dashboard Overview Process** - real-time data aggregation, KPI calculation, alert generation, and executive summary presentation. These process maps are implemented as technical documentation in the system's documentation folder, providing clear visual representations of data flows, decision points, and system interactions. (Detailed process flow visualizations are available in the system's documentation at documentation/process\_flow\_visualization.html and technical mapping at documentation/technical\_process\_mapping.md)

**4.5.2 Inputs, Outputs, Activities, and Decision Points**

Each core process is structured with clearly defined components: **Product Management:** Inputs include supplier information, product specifications, categories, and images; Activities involve product registration, category assignment, supplier association, and image management; **Inventory Management:** Inputs comprise existing products, size/color options, and initial quantities; Activities include variant creation, stock assignment, and inventory tracking; Decision points validate combination uniqueness, quantity thresholds, and reorder levels; Outputs generate stocked variants, adjustment records, and stock alerts. **Order Management:** Inputs consist of customer information, product selections, and quantities; Activities encompass customer management, product selection, stock validation, and order creation; Decision points verify customer existence, stock availability, and quantity validity; Outputs create order records, customer history, and inventory adjustments. **Analytics Process:** Inputs include historical order data and time periods; Activities involve data aggregation, trend analysis, and visualization generation; Decision points determine sufficient data availability and analysis scope; Outputs deliver performance reports, dashboards, and business insights. Each process maintains comprehensive audit trails and supports the overall business workflow through seamless integration points.

**4.6 Key Performance Indicators (KPIs)**

The Spectra Style Clothing Management System utilizes three core KPIs to evaluate operational efficiency, data accuracy, and technical performance. These KPIs are designed to provide measurable insights into system effectiveness and support continuous improvement initiatives that directly impact business operations and customer satisfaction.

**Defined KPIs**

**KPI 1: Order Fulfillment Efficiency**

* **Description:**Measures the percentage of customer orders processed completely and on time—tracking the full cycle from order placement to delivery confirmation.
* **Measurement Method:**Calculated as:(OrdersCompletedWithinTargetTime÷TotalOrders)×100(Orders Completed Within Target Time ÷ Total Orders) × 100%(OrdersCompletedWithinTargetTime÷TotalOrders)×100  
  Uses Django Order model timestamps across status transitions (PENDING → PROCESSING → SHIPPED → DELIVERED).
* **Target Value:** ≥ 95% for excellent performance, with acceptable performance at 85-94%. Orders should progress through all status phases within defined timeframes to meet customer expectations and operational efficiency goals.

**KPI 2: Inventory Accuracy Rate**

* **Description:**Tracks how accurately the system’s recorded stock matches the actual physical inventory across product variants (size and color).
* **Measurement Method:**Calculated as:(VariantswithAccurateStock÷TotalActiveVariants)×100(Variants with Accurate Stock ÷ Total Active Variants) × 100%(VariantswithAccurateStock÷TotalActiveVariants)×100  
  Uses automated logging from the InventoryAdjustment model and periodic stock reconciliations.
* **Target Value:** ≥ 98% for excellent performance, with acceptable performance at 95-97%. High accuracy is essential for maintaining customer trust and operational efficiency in the fashion retail environment where size and color combinations require precise tracking.

**KPI 3: System Performance Score**

* **Description:**A composite metric evaluating system speed, uptime, and ease of task completion by users.
* **Measurement Method:**Composite formula:  
  (PageLoadSpeed×0.4)+(SystemUptime×0.3)+(UserTaskCompletion×0.3)(Page Load Speed × 0.4) + (System Uptime × 0.3) + (User Task Completion × 0.3)(PageLoadSpeed×0.4)+(SystemUptime×0.3)+(UserTaskCompletion×0.3)
* Page Load: Target under 2 seconds (95th percentile)
* System Uptime: Monitored via server logs (≥ 99.5%)
* User Task Completion: Based on successful completion of key workflows (e.g., processing orders, adding inventory)
* **Target Value:** ≥ 90/100 overall score. Individual component targets include page load times under 2 seconds (95th percentile), system uptime ≥ 99.5% monthly, and task completion rate ≥ 95% for critical user workflows.

**4.7 Service Level Agreement**

**4.7.1 Proposed SLA Outline**

The Service Level Agreement (SLA) for the Spectra Style Clothing Management System outlines the expectations between the system provider and users. It ensures consistent performance, availability, and support across key system features like product browsing, order processing, inventory, and analytics. The SLA includes specific performance metrics, response times, and support procedures. It also clarifies responsibilities while excluding uncontrollable events such as third-party failures or force majeure.

**4.7.2 Expected Service Level Metrics**

**Availability**: The system promises 99.9% uptime for customer-facing pages and 99.5% for admin features, with maintenance scheduled only during off-peak hours and announced 48 hours in advance.  
**Response Time**: Critical issues (P1) will get a response within 1 hour and be resolved within 4 hours. High (P2), Medium (P3), and Low (P4) priority issues will follow longer response/resolution timelines depending on urgency. Pages are expected to load in under 2 seconds, with API responses under 500ms for reads and under 1 second for updates.  
**Support**: Users can contact support via a helpdesk, email, or phone (for urgent cases). Support is available 24/7 for tickets and during extended hours for critical issues.  
**Data Backup and Recovery**: The system performs daily backups (kept for 7 days), weekly full backups (30 days), and monthly archives (1 year). The goal is to lose no more than 24 hours of data (RPO) and to restore the system within 4 hours (RTO) during a failure.  
**Security Incidents**: Any security breach is handled immediately, with investigation and client notification within 24 hours. A full report is given within 72 hours, along with prevention strategies.

**4.8 Emerging Technology Integration**

**4.8.1 Identified Emerging Technology**

**Technology Name:** Artificial Intelligence (AI) for Personalization and Customer Experience Enhancement

**Description:** Artificial Intelligence, specifically machine learning (ML) algorithms, can be leveraged to create highly personalized shopping experiences for users. This involves analyzing vast amounts of user data—such as browsing history, purchase patterns, wish-listed items, demographic information, and real-time interactions—to understand individual preferences and predict future behavior.

**4.8.2 Function within the System**

AI personalization would be integrated into several key areas of the Spectra Style Clothing system to enhance user experience and business intelligence.

**Personalized Product**

**Recommendations:** Machine learning algorithms would analyze user behavior patterns to display "Recommended for You" sections on product pages, homepage, and shopping cart, using collaborative filtering and content-based filtering to suggest relevant items. **Dynamic Content Personalization:** The homepage layout, banners, and featured products would be customized based on individual user profiles, prioritizing categories, styles, and brands that align with demonstrated customer interests. **Intelligent Search Enhancement:** Search results would be re-ranked beyond simple keyword matching to prioritize items most relevant to specific users' preferences, brand affinity, size preferences, and color choices. **Targeted Promotions:** The system would generate personalized email campaigns and on-site offers based on abandoned carts, wish lists, and browsing patterns. **Enhanced Customer Support:** AI-powered chatbots would provide personalized product suggestions and assist with order-related questions by accessing individual customer history and preferences, improving support efficiency and customer satisfaction.

**Benefits and Justification**

**Benefits:** AI personalization delivers enhanced customer experience by making users feel understood and valued through relevant product presentations, leading to increased conversion rates as customers are more likely to purchase items that match their preferences. The technology drives higher average order values through intelligent cross-selling and upselling recommendations such as "complete the look" suggestions. It improves customer loyalty and retention by fostering stronger brand connections through personalized experiences, while providing better inventory management insights through demand prediction and trend analysis. Additionally, AI enables efficient scaling of personalized experiences as the product catalog and customer base grow, maintaining relevance without manual intervention.

**Justification:** For the Spectra Style Clothing system operating in the competitive fashion industry, AI personalization is strategically crucial for several reasons. It directly addresses modern consumer expectations for tailored shopping experiences that cater to individual tastes and needs, moving beyond generic one-size-fits-all approaches. The technology significantly improves key business metrics including sales, customer lifetime value, and engagement rates while enabling smarter utilization of the product catalog by highlighting relevant items for each user. This prevents popular products from overshadowing niche items that specific customers might prefer. As the business scales and product range expands, AI efficiently manages the complexity of matching customers with appropriate products—a task that would be manually unfeasible. Implementing AI personalization represents a strategic investment in building a more customer-centric, efficient, and profitable e-commerce business that aligns with maximizing customer satisfaction and driving sustainable growth for Spectra Clothing in an increasingly competitive marketplace.

**CHAPTER 5: INDIVIDUAL MEMBER CONTRIBUTIONS**

**Member 1: Gonzales, Marc Jason G.**

**Assigned Role(s): Quality Analyst & Tester (Also contributed as System Analyst & Researcher)**  
**Backend Contributions:**

* Validated business logic and tested CRUD operations on Django views
* Reported bugs and inconsistencies in model relationships and stock syncing

**Frontend Contributions:**

* Conducted manual tests on UI responsiveness and validation states
* Reported layout issues across browsers and screen sizes

**Other Contributions:**

* Led test case creation and bug tracking
* Reviewed system KPIs and proposed optimizations based on test results
* Assisted in process mapping for quality control points
* Participated in drafting the SLA and KPI measurement documentation

**Member 2: Laher, Allen Jay B.**

**Assigned Role(s): Backend Developer, System Analyst (Also contributed to Frontend)**  
**Backend Contributions:**

* Designed and developed Django models for Product, Inventory, Order, and Customer
* Implemented RESTful API endpoints for CRUD operations
* Built authentication and role-based access control using Django Auth
* Developed logic for order processing, stock deduction, and low-stock alerts
* Wrote inventory adjustment and order status transition logic

**Frontend Contributions:**

* Integrated API responses with frontend components using AJAX and JavaScript
* Helped build interactive forms for product creation and checkout
* Applied Tailwind CSS classes for layout consistency

**Other Contributions:**

* Created process flowcharts and system mapping diagrams
* Contributed to KPI definitions and SLA drafting
* Performed system analysis for performance optimization and logic validation

**Member 3: Vega, Erika Shigella L.**

**Assigned Role(s):** Project Manager, Frontend Developer  
**Backend Contributions:**

* Coordinated integration between frontend and backend teams
* Assisted in defining API requirements for product, inventory, and order management
* Collaborated in testing authentication and order workflows

**Frontend Contributions:**

* Designed and implemented key UI pages using Tailwind CSS
* Created responsive layouts for the dashboard, product catalog, and checkout pages
* Implemented JavaScript-based form validation and UI interactions
* Enhanced mobile responsiveness across modules

**Other Contributions:**

* Led project planning and task delegation using Asana
* Wrote major sections of the project documentation (Executive Summary, Project Objectives, Chapter 1)
* Oversaw compliance with KPIs and deliverable deadlines
* Participated in user acceptance testing

**Member 3: Villagracia, Manuel Gabriel R.**

**Assigned Role(s): UI Designer**  
**Backend Contributions:**

* N/A

**Frontend Contributions:**

* Designed all primary UI layouts using Tailwind CSS utility classes
* Created style guide and consistent UI elements across all modules
* Crafted visual themes and button states for user roles (Admin, Stock Manager, Sales Staff)
* Designed responsive product detail, cart, and analytics dashboard interfaces

**Other Contributions:**

* Collaborated in usability testing for UI/UX improvements
* Provided UI accessibility enhancements (color contrast, mobile navigation)
* Helped compile user guide screenshots and visual documentation

**Member 5: Yap, Fileteo II L.**

**Assigned Role(s): Quality Analyst & Tester**  
**Backend Contributions:**

* Executed backend validation testing for user authentication and order processing
* Logged test results from system load and speed scenarios

**Frontend Contributions:**

* Tested form validation behavior and navigation flow on key modules
* Reported styling issues and inconsistent button behavior

**Other Contributions:**

* Helped document testing results and screenshots
* Participated in role-based testing simulations (UAT)
* Verified error handling for incorrect inputs and failed requests

**CHAPTER 6: CONCLUSION & RECOMMENDATIONS**

**Project Outcomes**

The Spectra Style Clothing Management System met its main goals by delivering a functional, web-based platform that automates key operations for clothing boutiques. Using Django, Python, MySQL, Tailwind CSS, and JavaScript, the system includes full CRUD features, secure role-based access, real-time inventory tracking, order processing, supplier coordination, and analytics dashboards. Each mock client’s needs were addressed—Ms. Escalada gained full business oversight, Mr. Alovera improved inventory handling, and Mr. Delsocora processed orders smoothly. The system passed multiple testing phases, ensuring reliability, responsiveness, and data security, making it ready for real-world deployment.

**Future Recommendations**

To enhance the system further, future upgrades could include **AI-driven product recommendations**, **automated reordering**, and **third-party integrations** (e.g., accounting software, shipping APIs). Implementing **SKU-based inventory tracking** will provide more accurate stock monitoring and streamline inventory management. **Scalability improvements**, such as adopting **microservices architecture** and developing a **dedicated mobile app**, are also suggested to support business growth. Additionally, integrating **workflow automation**, **advanced reporting tools** with **interactive graphs**, and **customer feedback systems** will significantly optimize operations and decision-making.

**Lessons learned** throughout the project emphasize the importance of clear planning, continuous testing, and comprehensive documentation. The successful completion of this project lays a solid foundation for future innovations in retail and e-commerce system development.

**References / Bibliography**

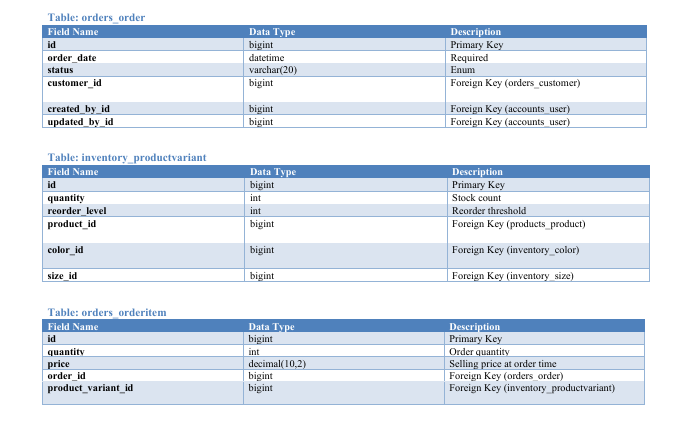
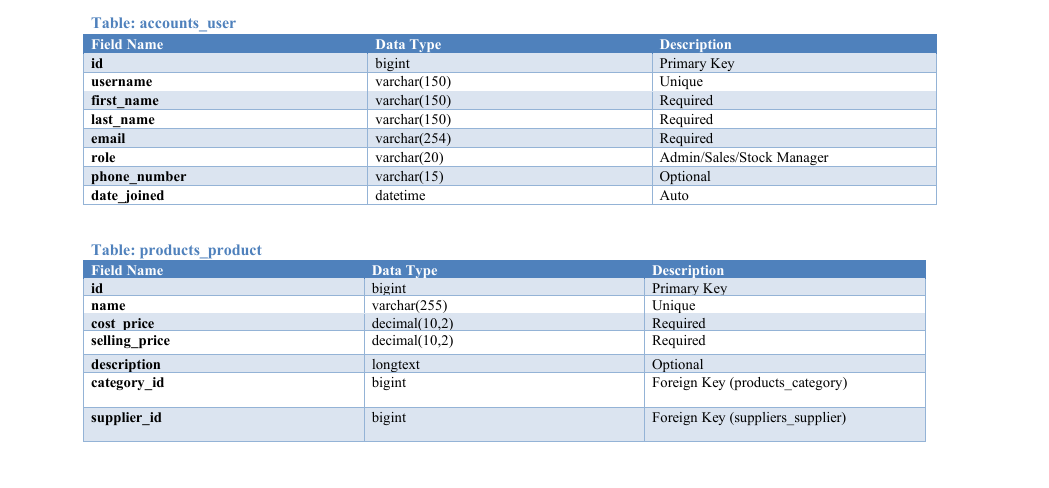
* Python Software Foundation. (n.d.). *Python Language Reference*. Retrieved from <https://docs.python.org/3/reference/index.html>
* Django Software Foundation. (n.d.). *Django Documentation*. Retrieved from <https://docs.djangoproject.com/en/stable/>
* Oracle Corporation. (n.d.). *MySQL Reference Manual*. Retrieved from <https://dev.mysql.com/doc/refman/8.0/en/>
* Mozilla Developer Network. (n.d.). *JavaScript*. Retrieved from <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
* Tailwind Labs Inc. (n.d.). *Tailwind CSS Documentation*. Retrieved from <https://tailwindcss.com/docs>
* W3Schools. (n.d.). *HTML Tutorial*. Retrieved from <https://www.w3schools.com/html/>
* W3Schools. (n.d.). *CSS Tutorial*. Retrieved from <https://www.w3schools.com/css/>
* Asana, Inc. (n.d.). *Asana Guide*. Retrieved from <https://asana.com/guide>

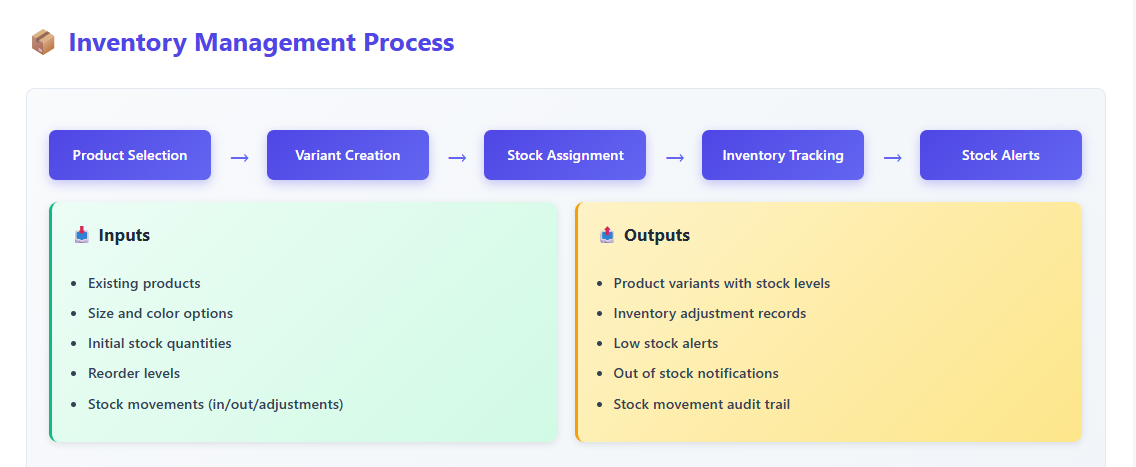
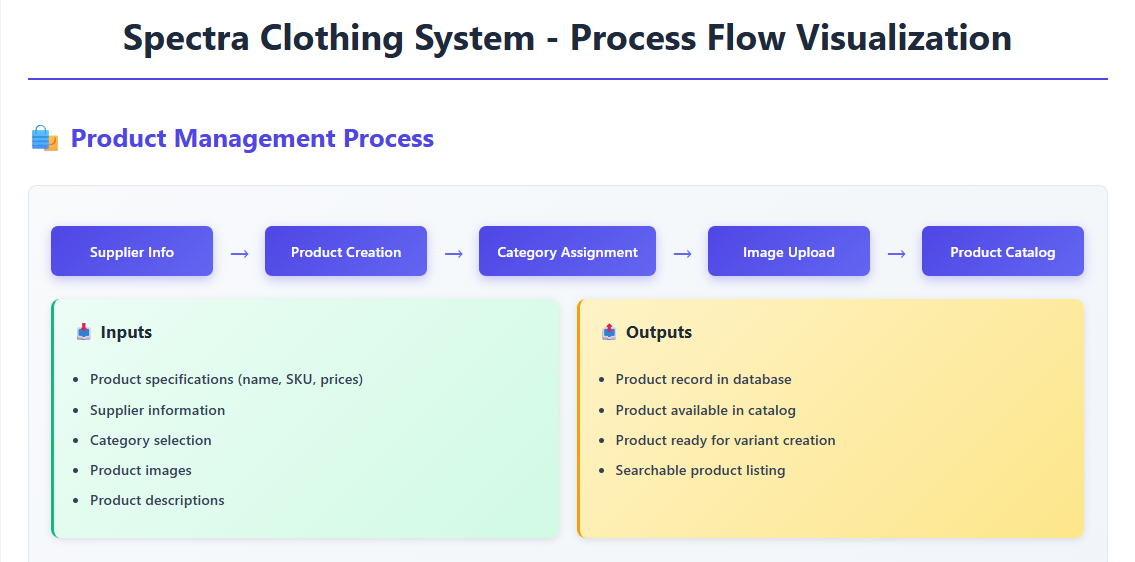
**Appendices**

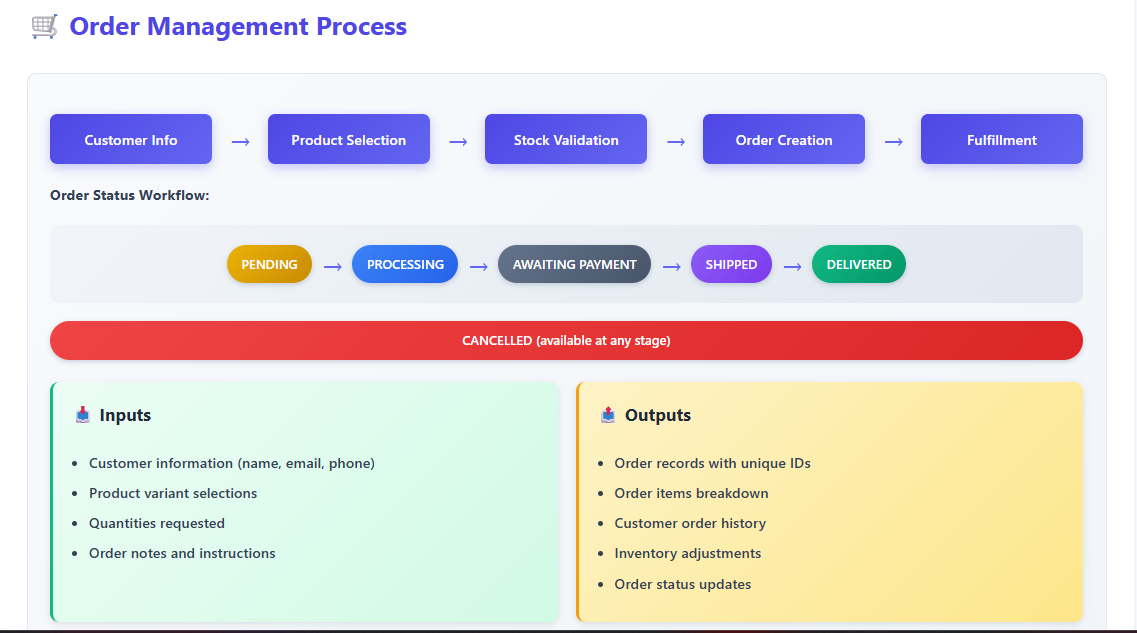
|  |  |  |
| --- | --- | --- |
| Table No. | Title | Page No. |
| Table 1 | Functional Requirements Checklist |  |
| Table 2 | Non-Functional Requirements Overview |  |
| Table 3 | Hardware Requirements |  |
| Table 4 | Software Requirements |  |

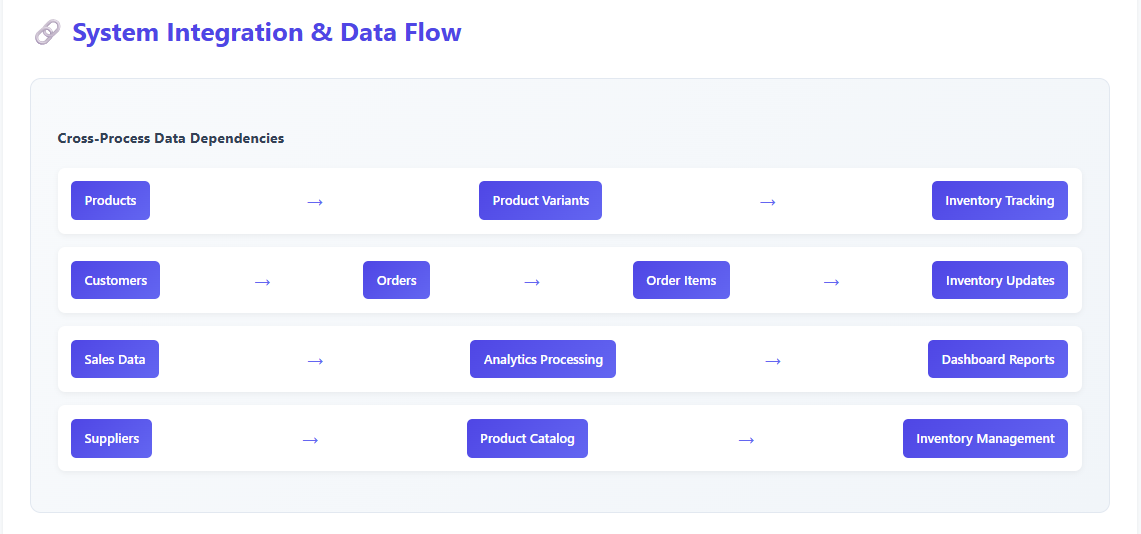
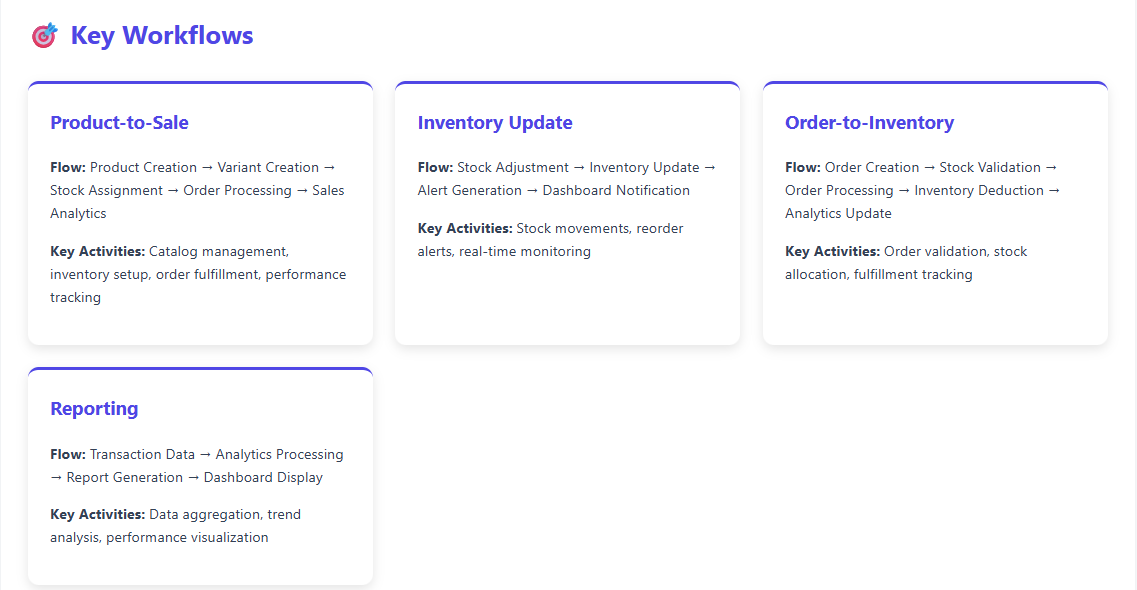
* **Appendix A: List of Tables:**
* **Appendix B: List of Figures:**
* **Appendix C: Diagrams:**

Entity-Relationship Diagram (ERD):

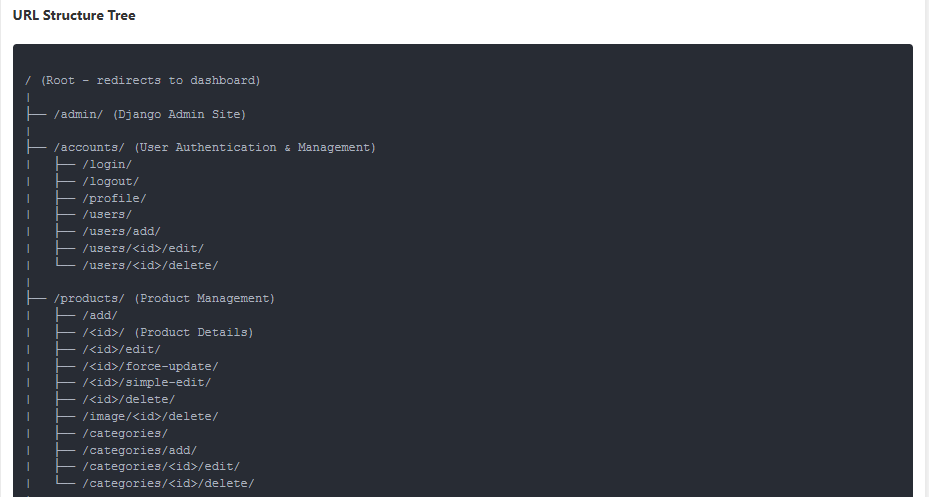


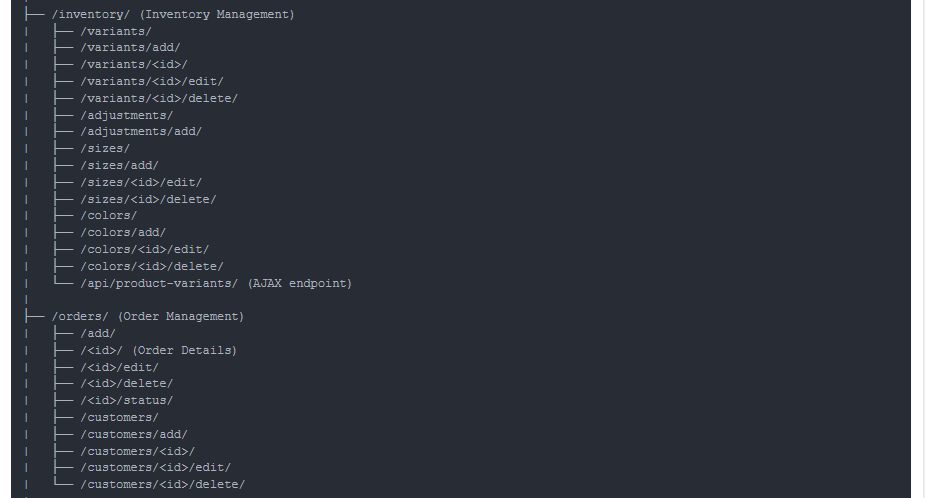
System Flowchart / Data Flow Diagram (DFD):****

****

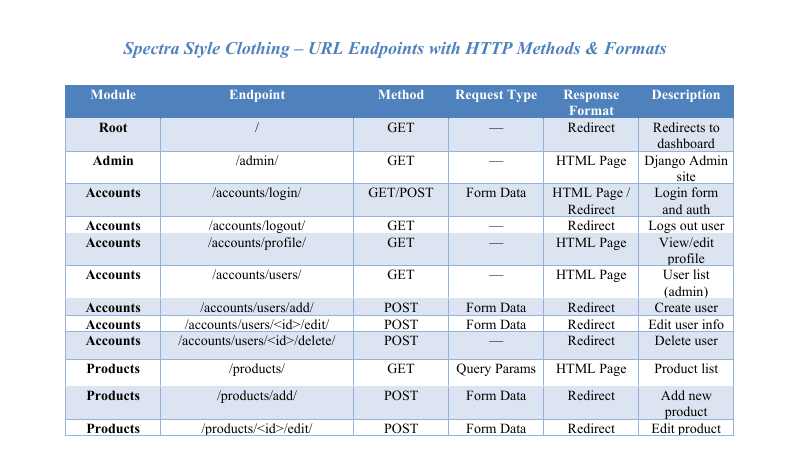
****

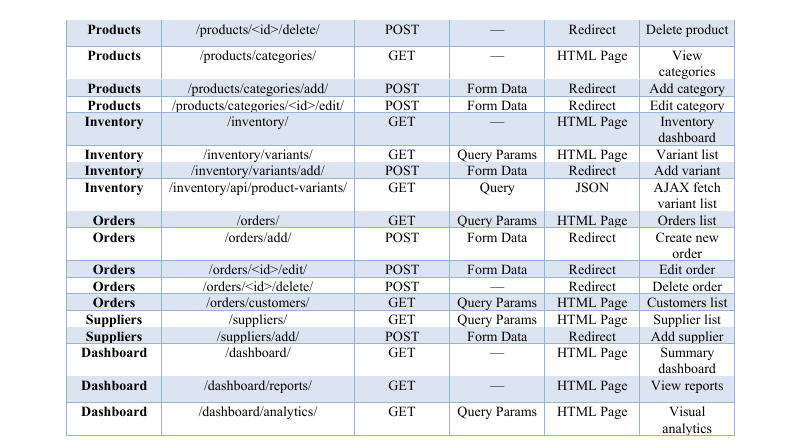
**Appendix D: Technical Specifications:**

* Detailed API endpoint documentation (e.g., endpoint URL, HTTP method, request/response format).
* Database table definitions (schema).

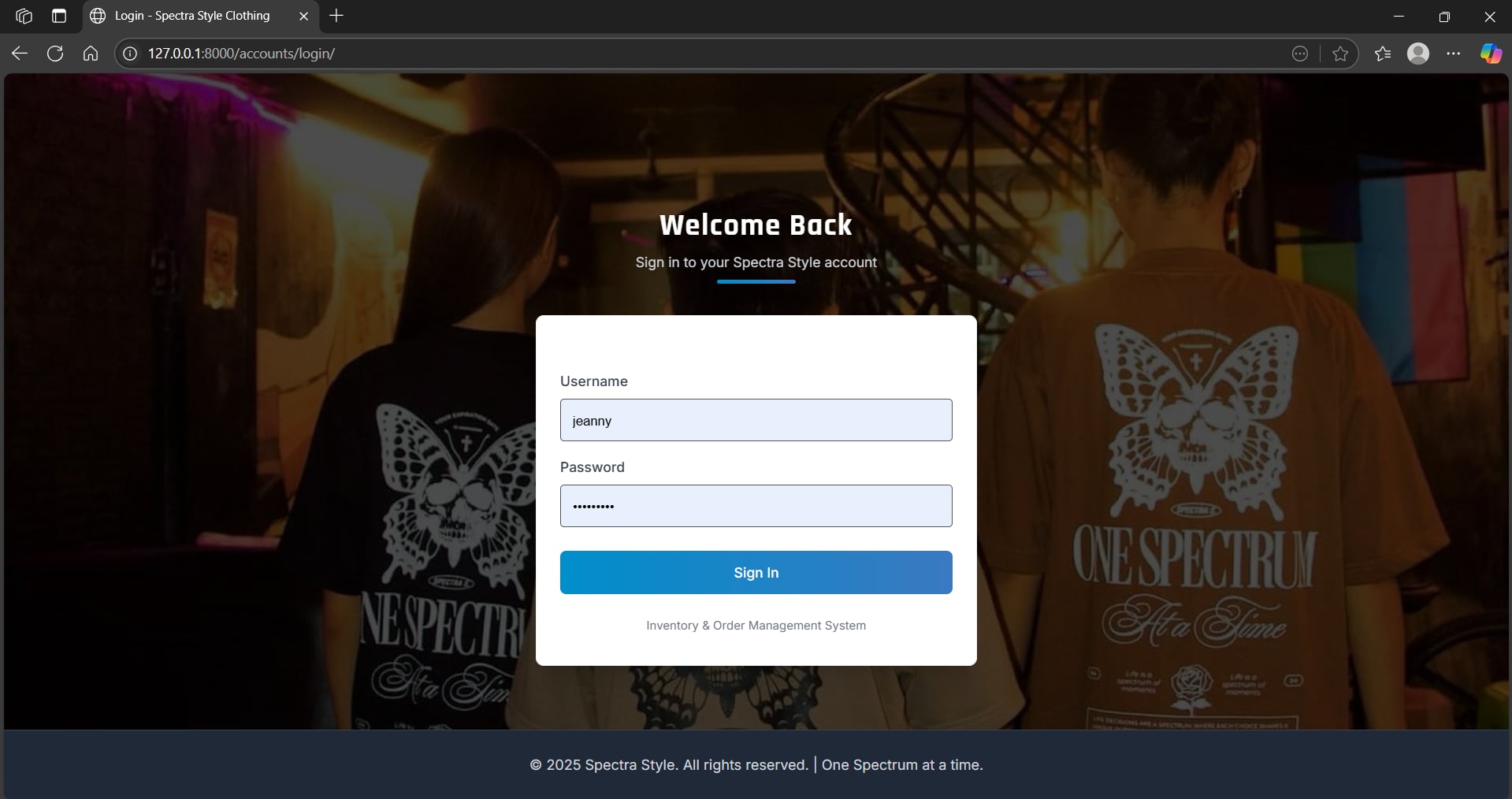


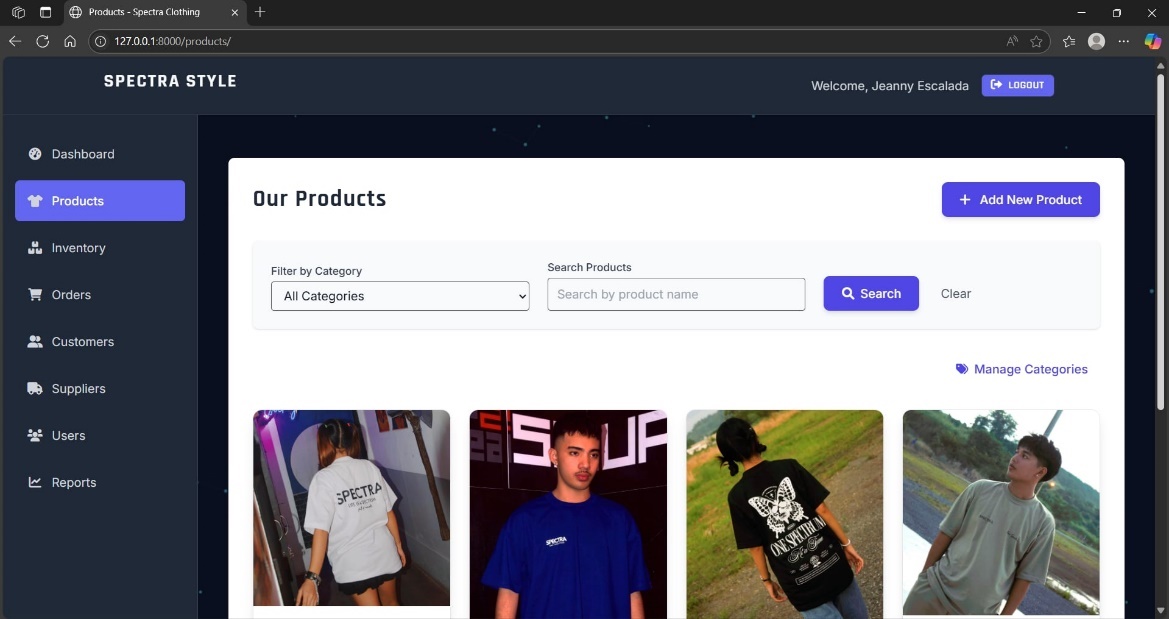
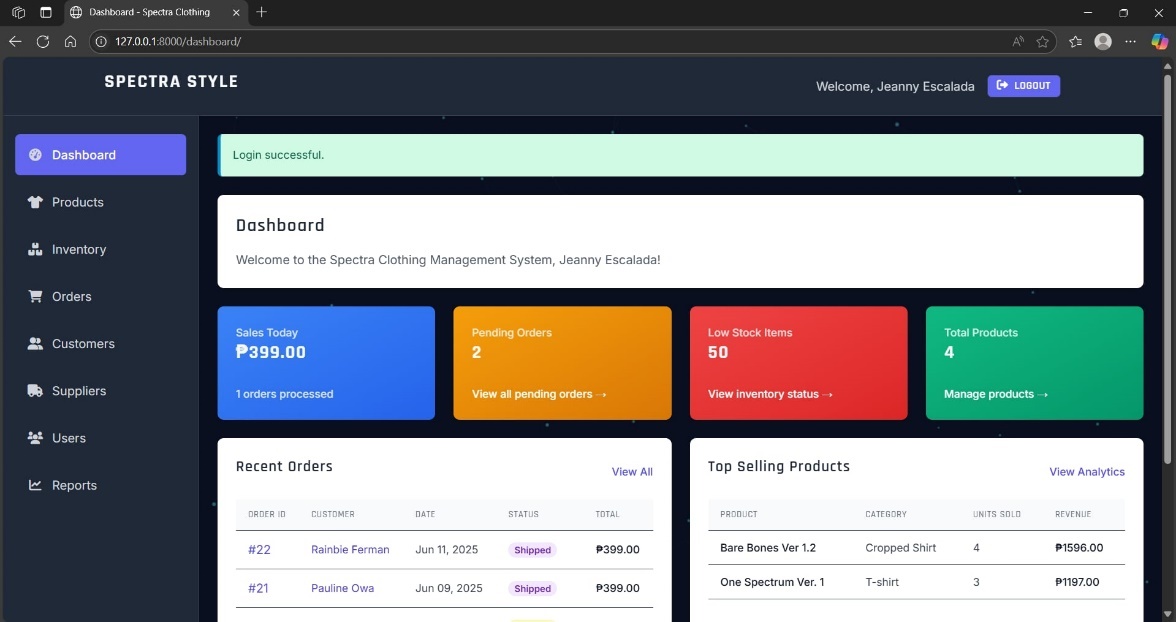


****

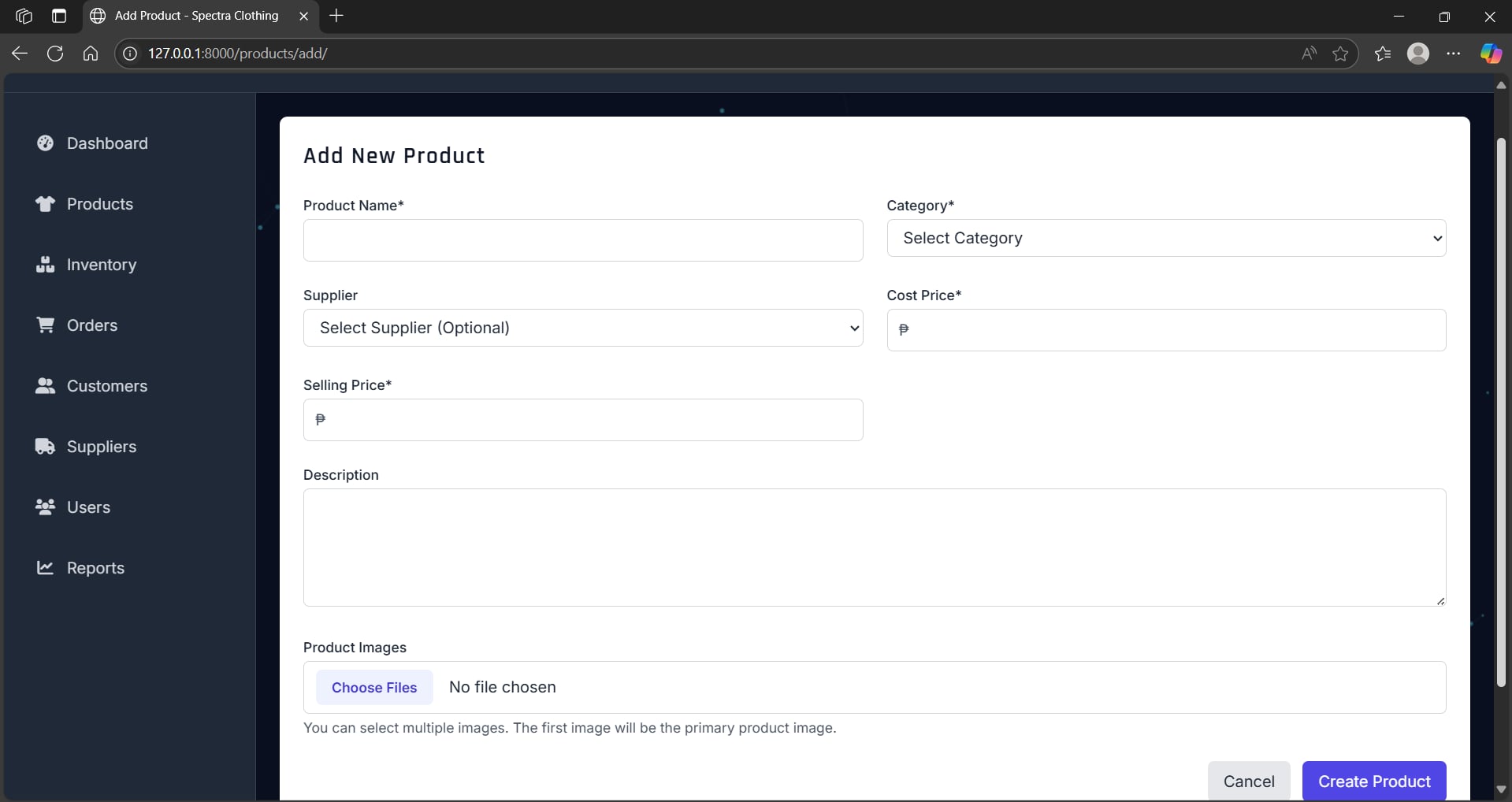
****

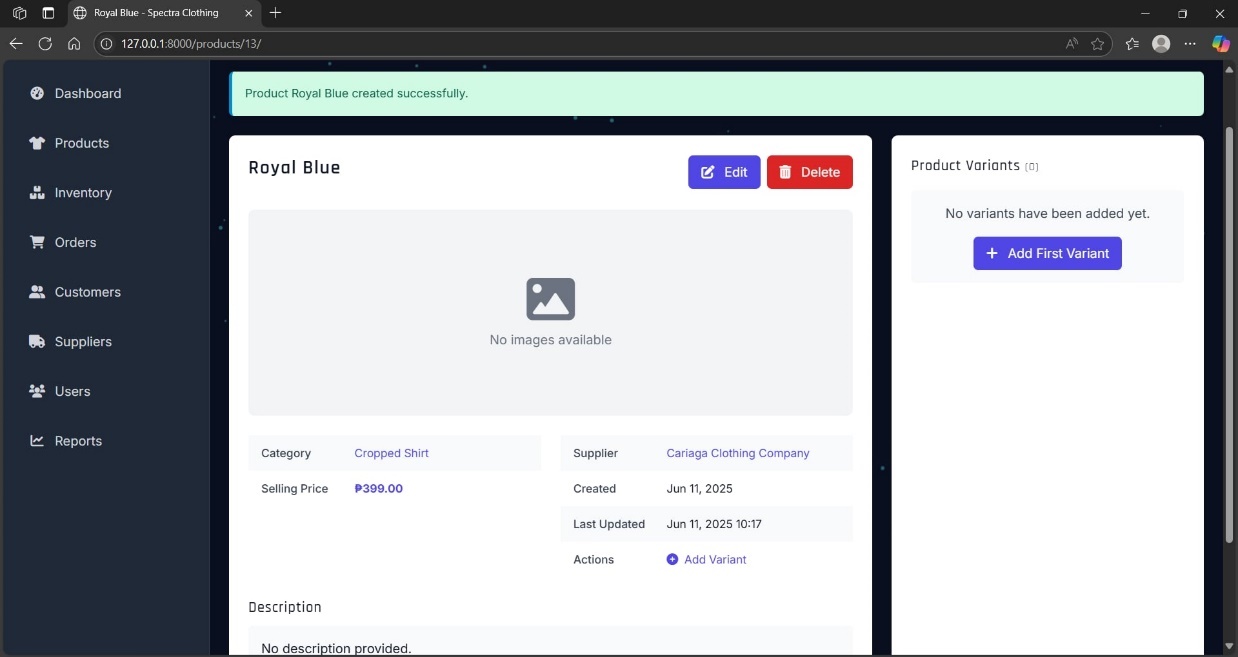
**Appendix E: Key Screenshots of the System:**

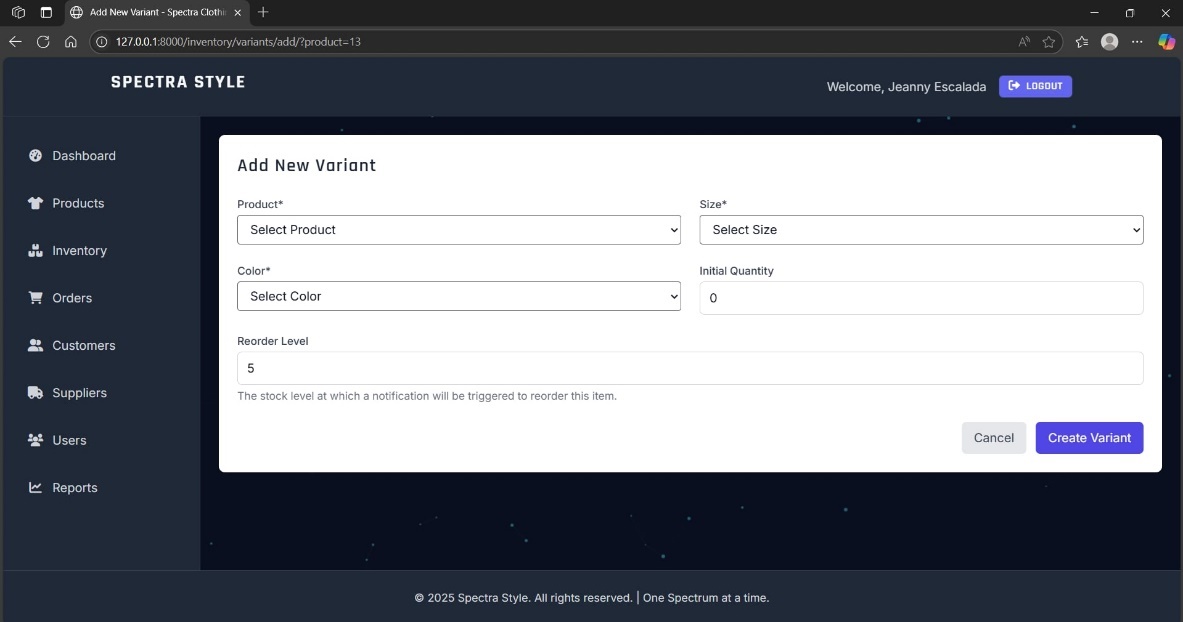


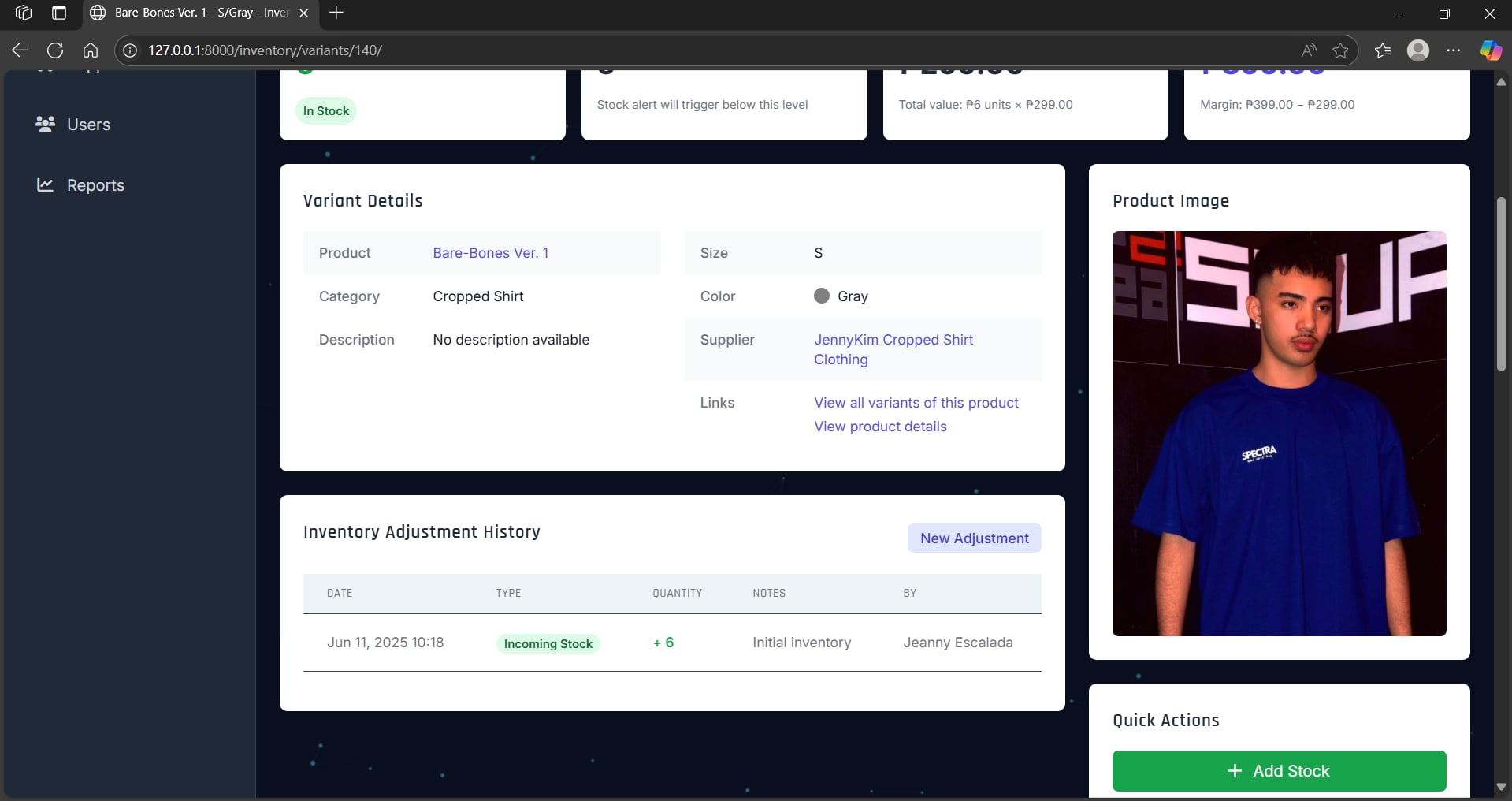
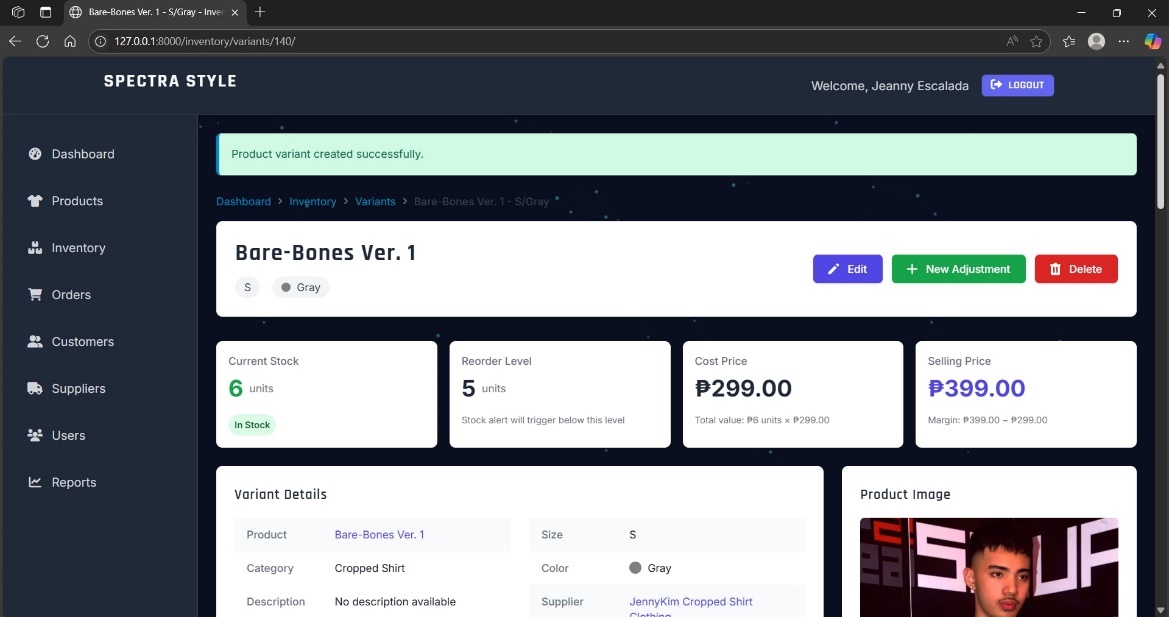


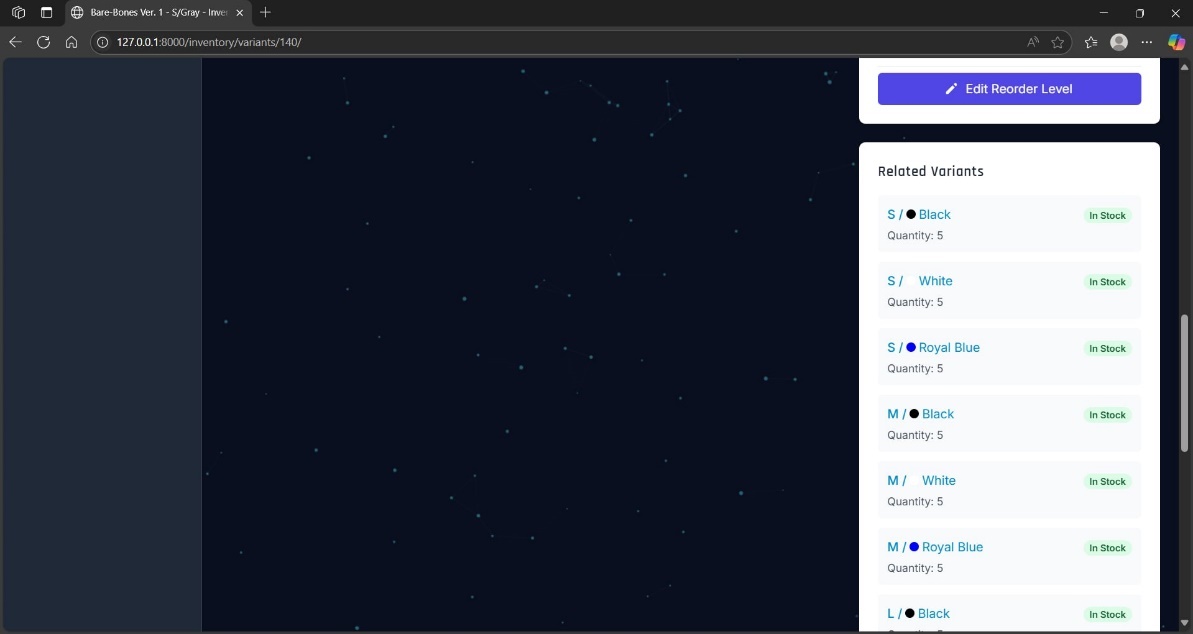
****

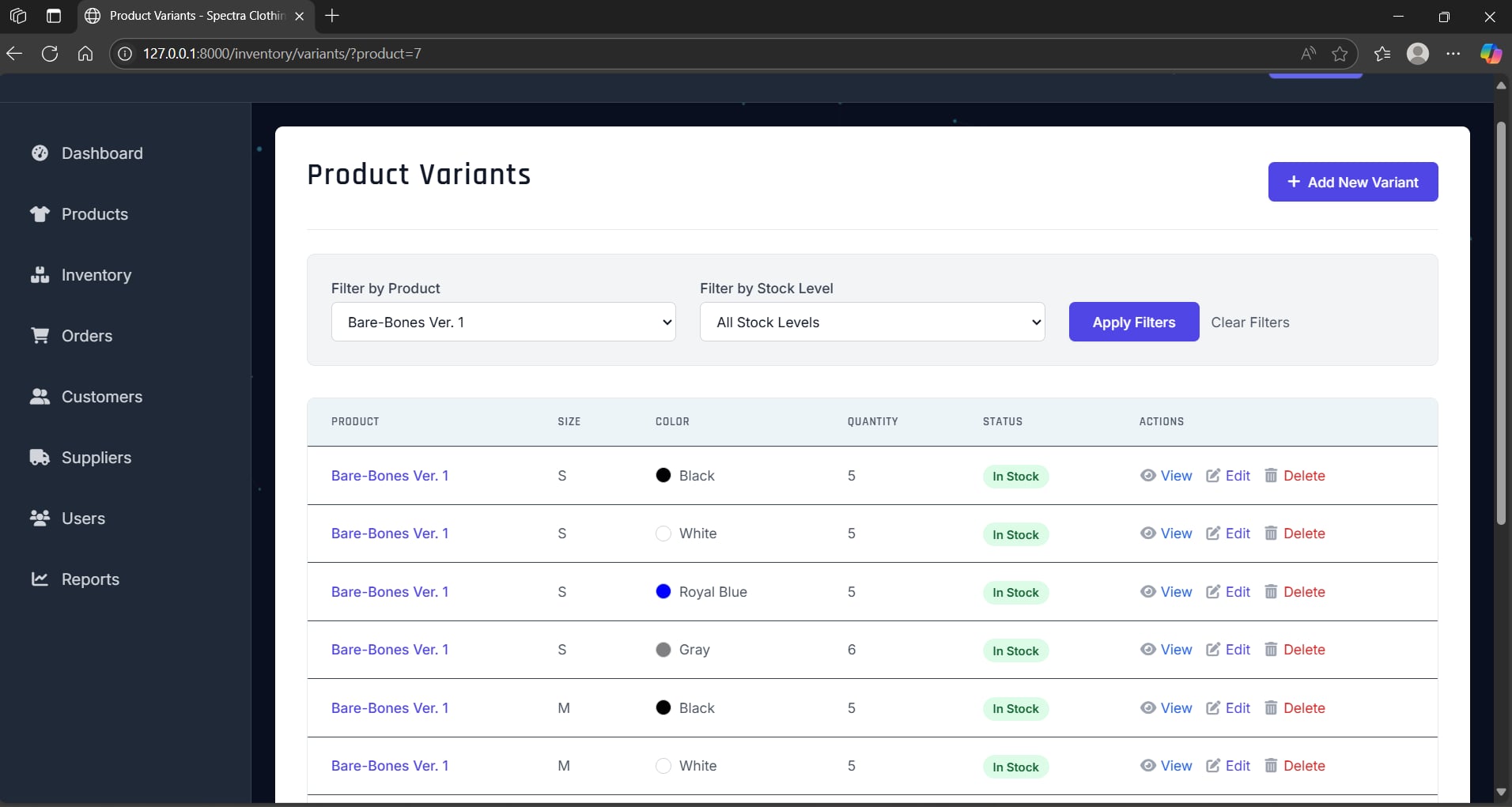
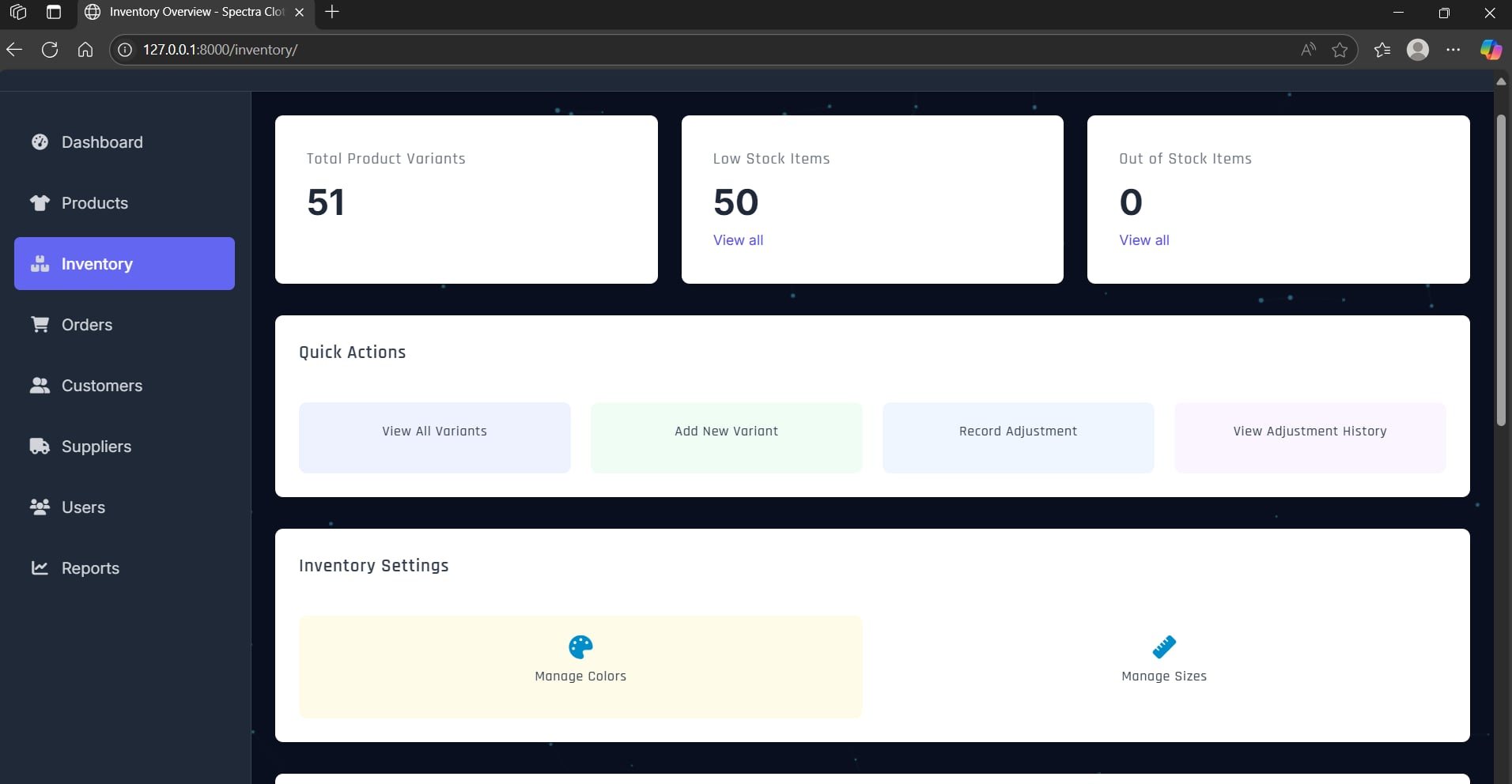
****

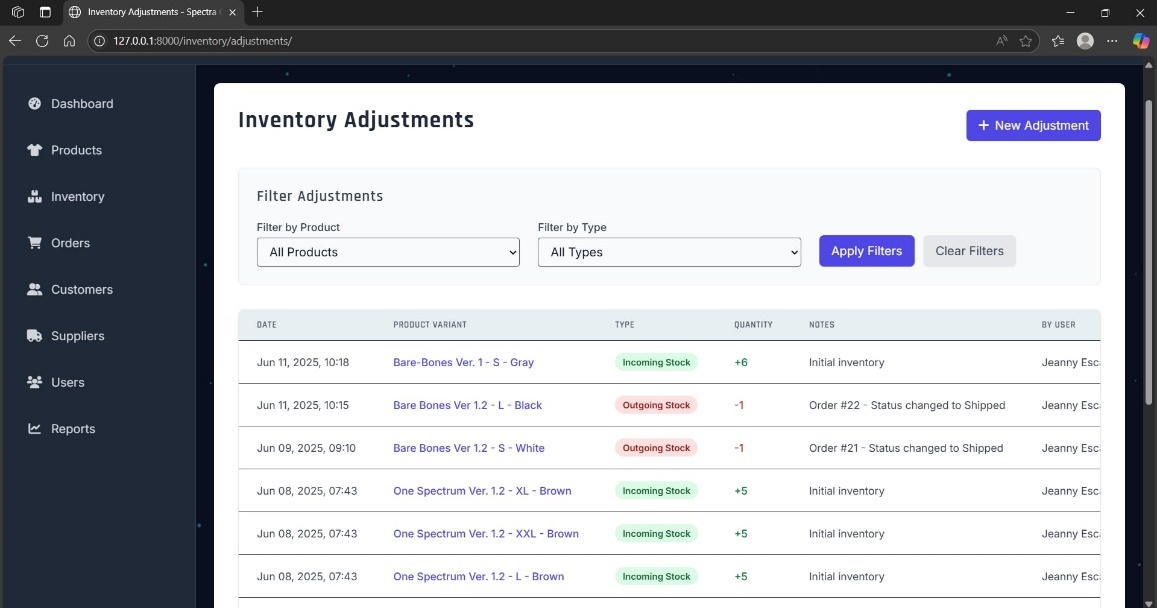
****

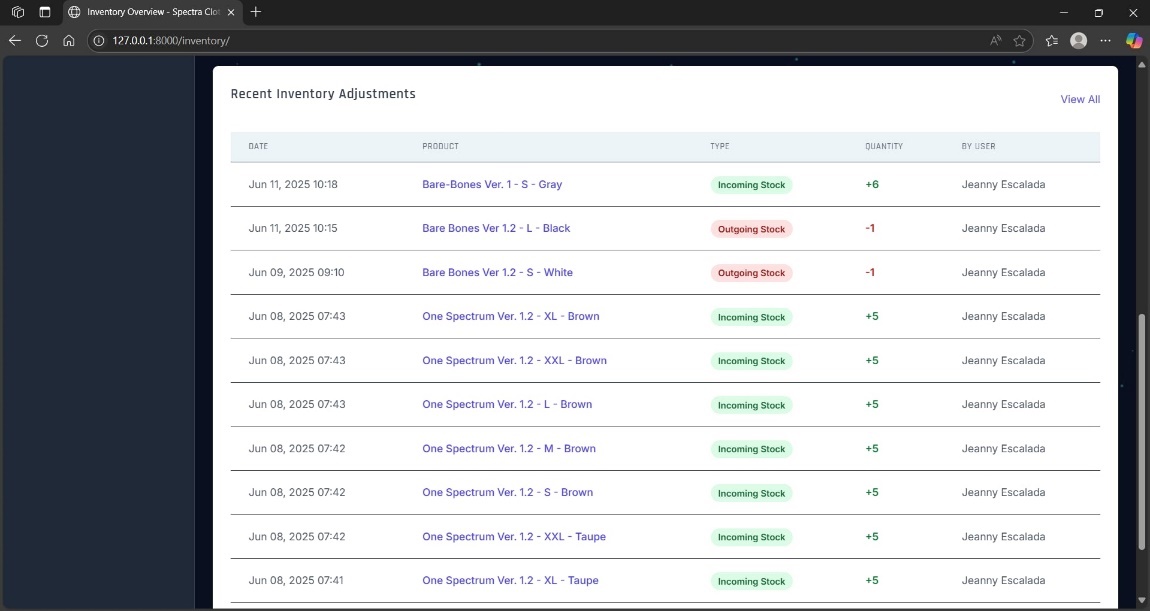
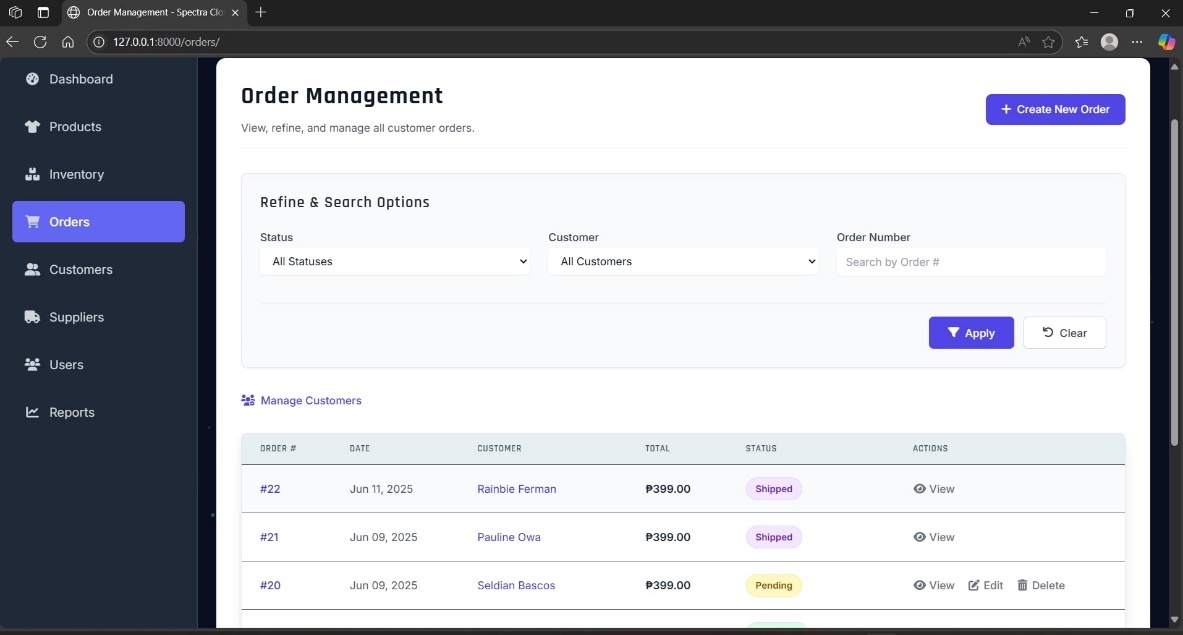


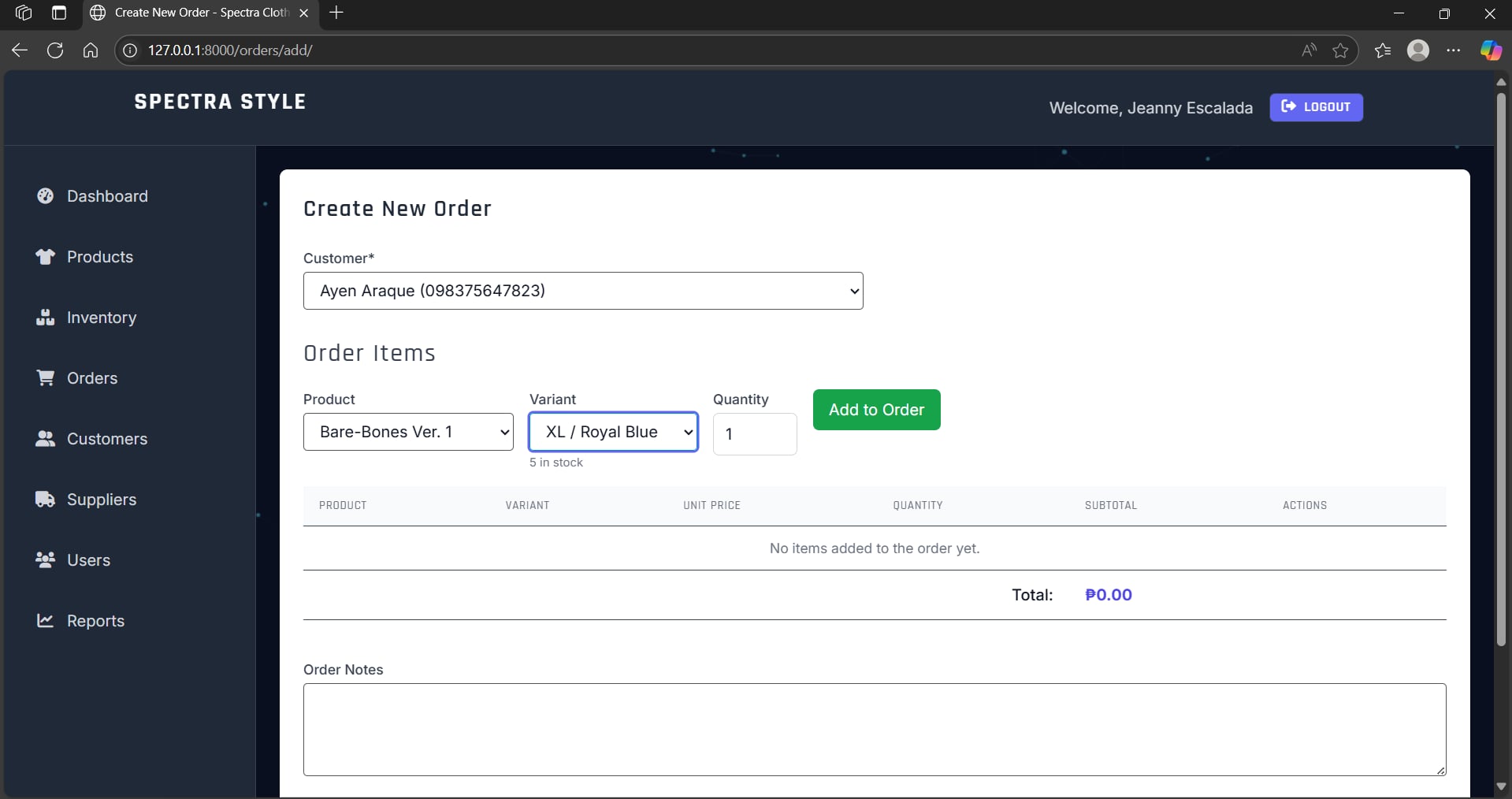


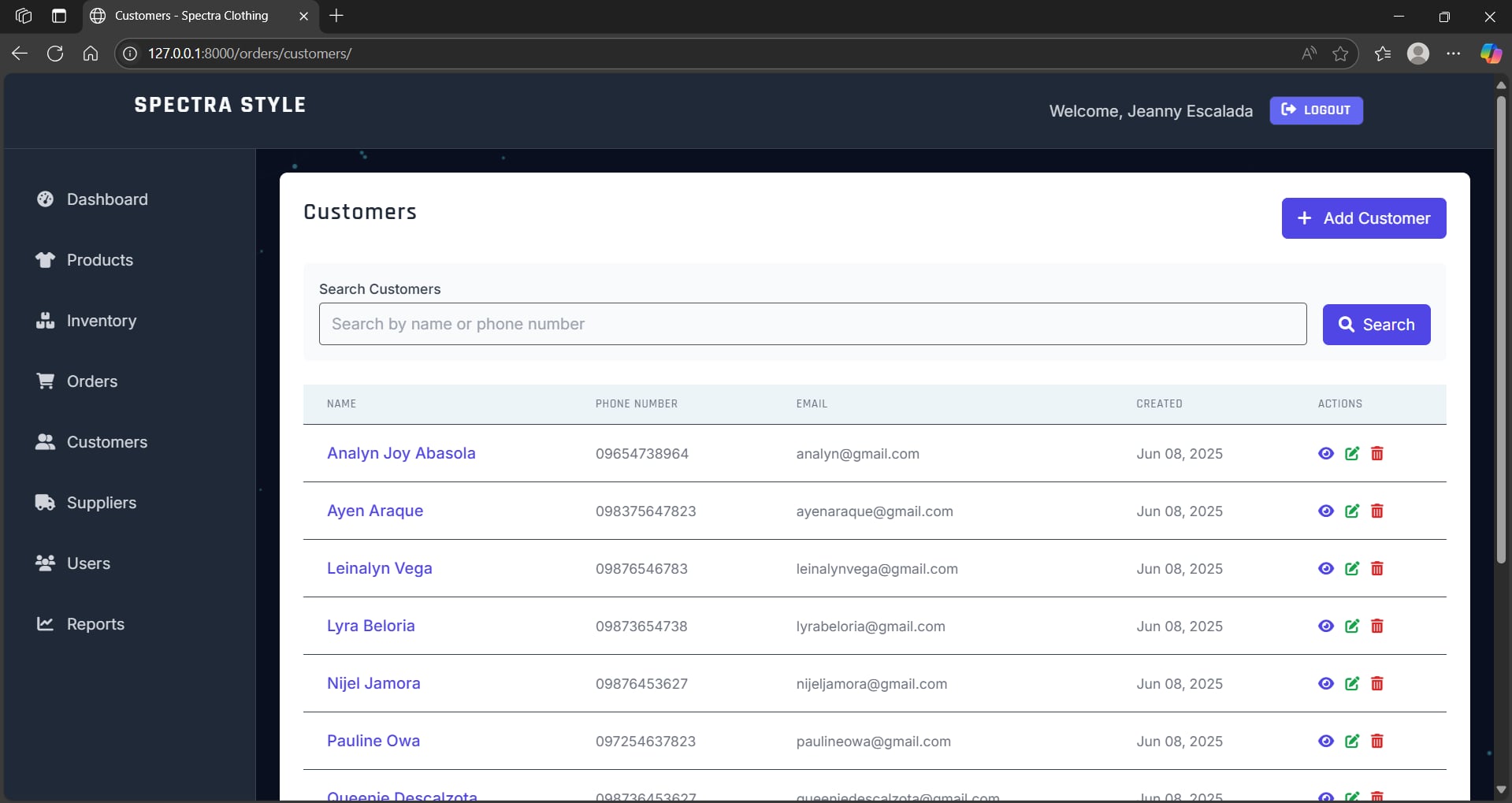
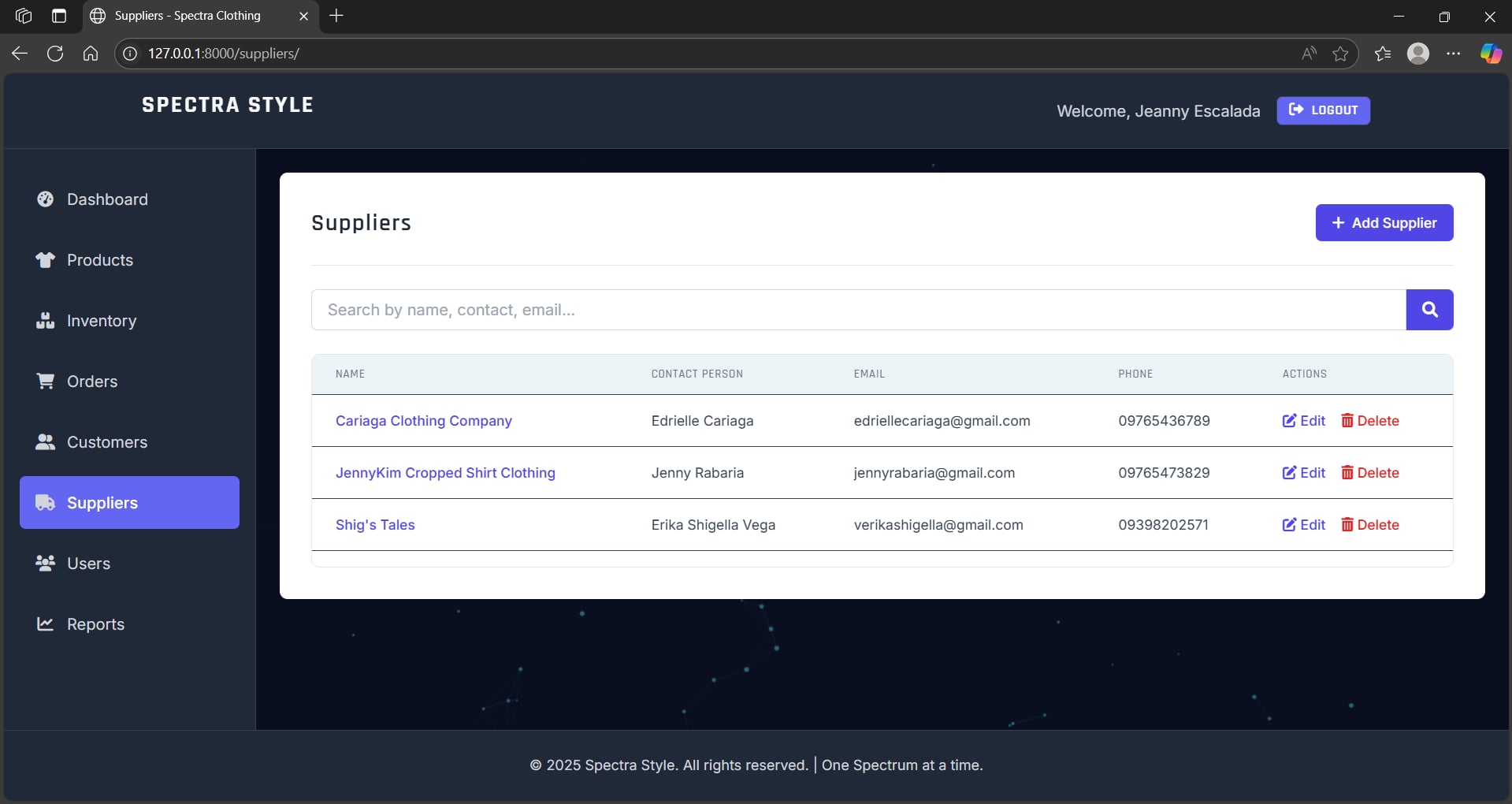


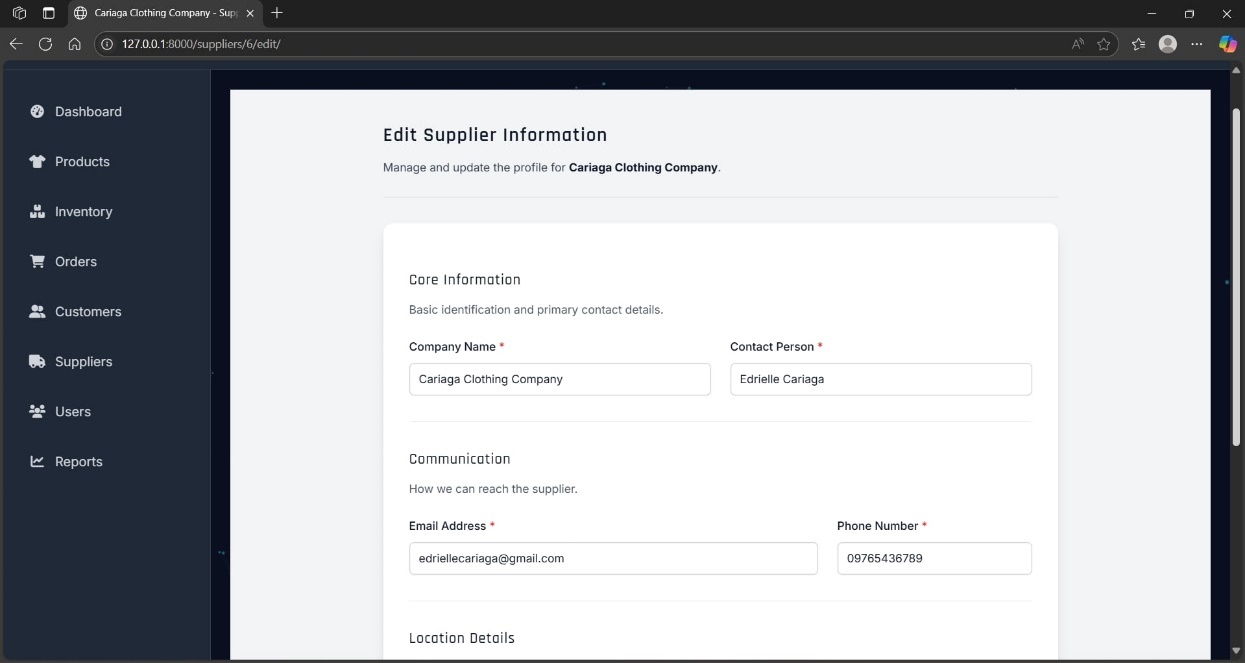


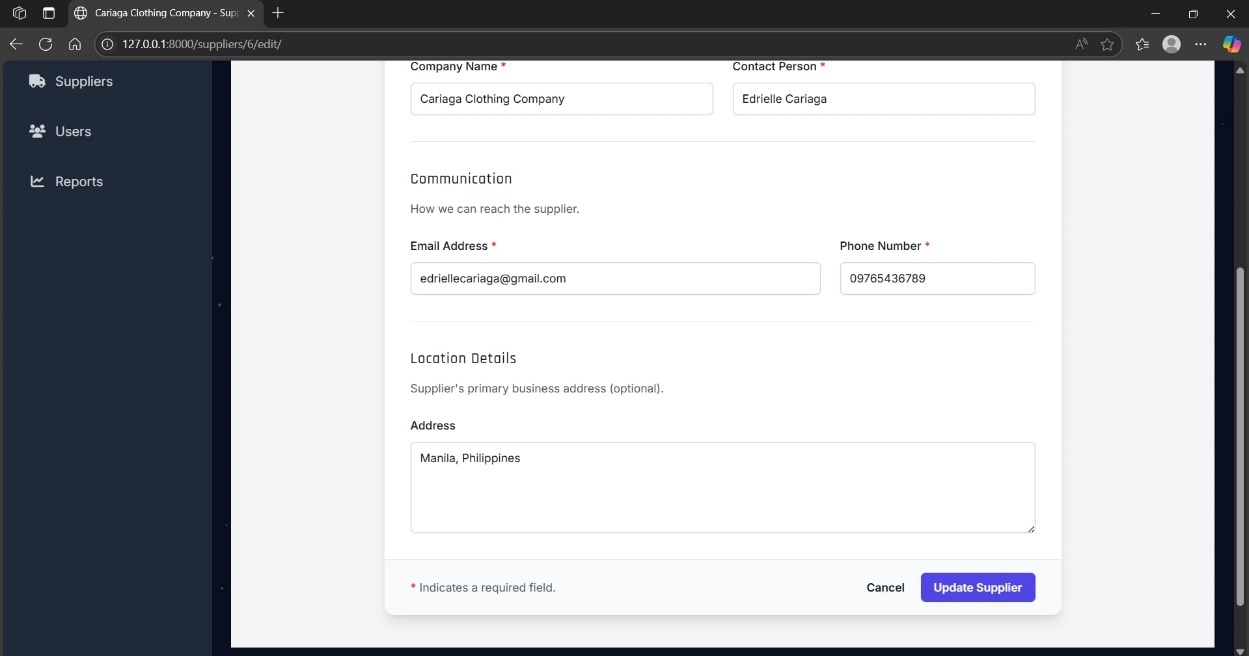
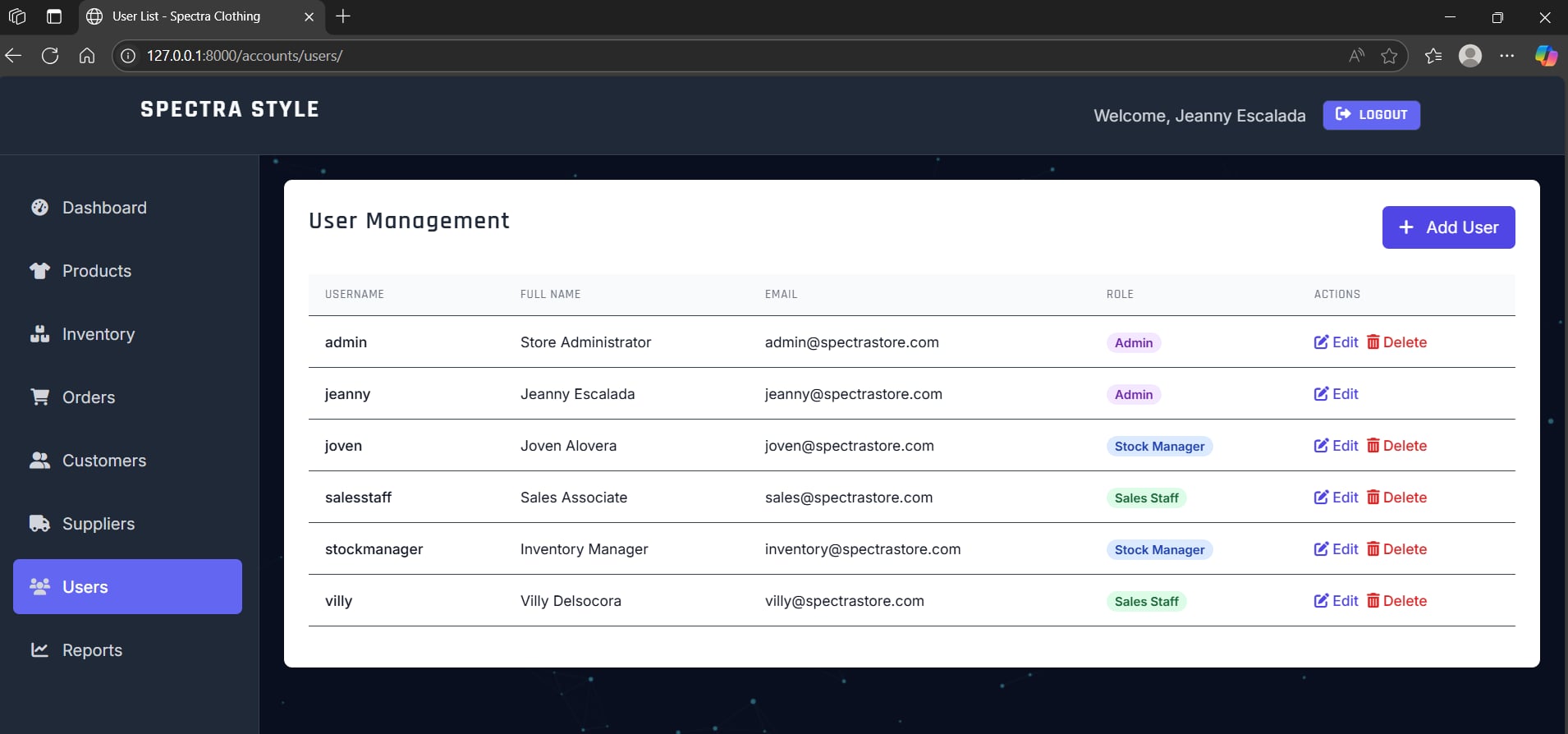


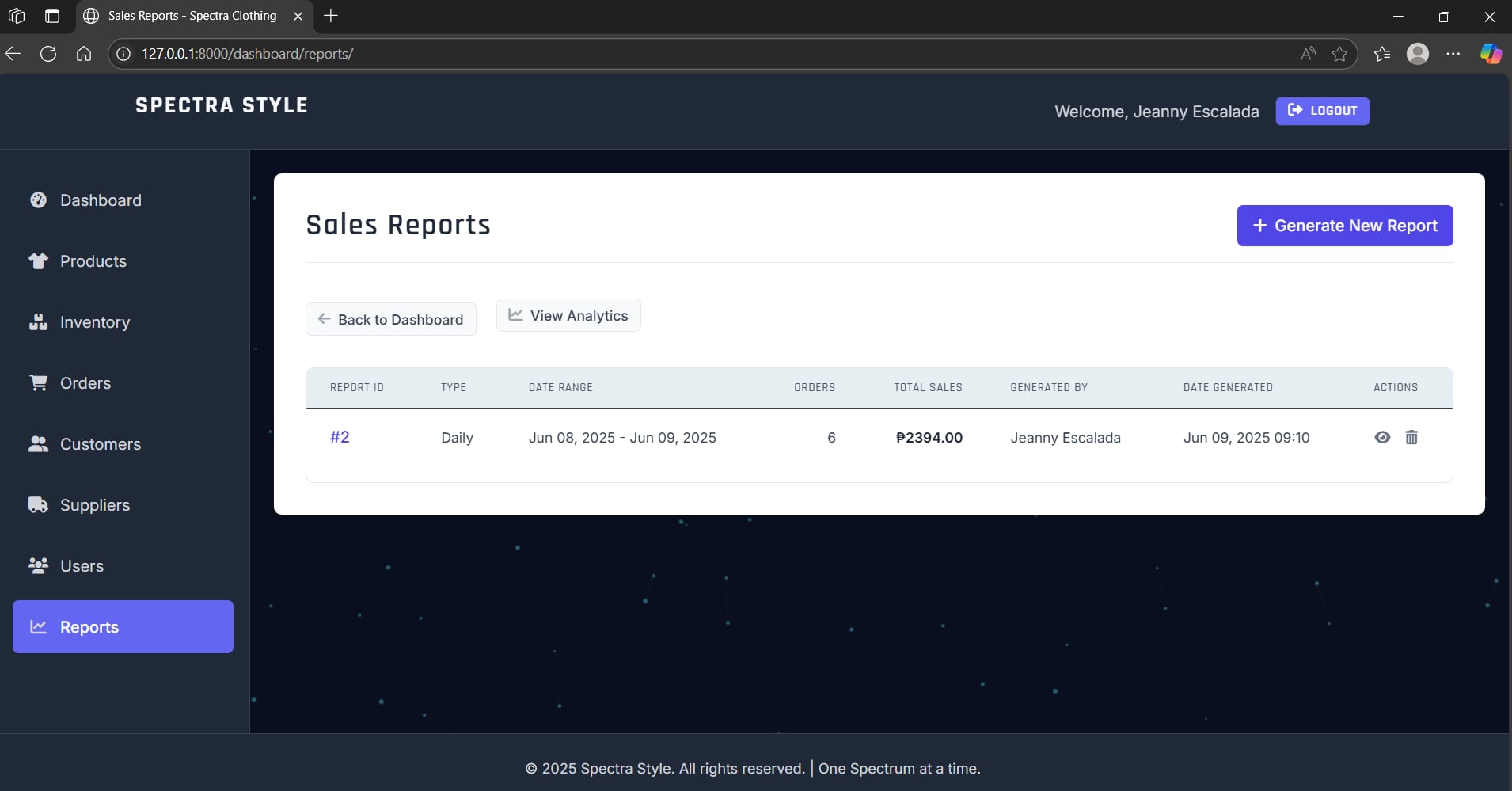


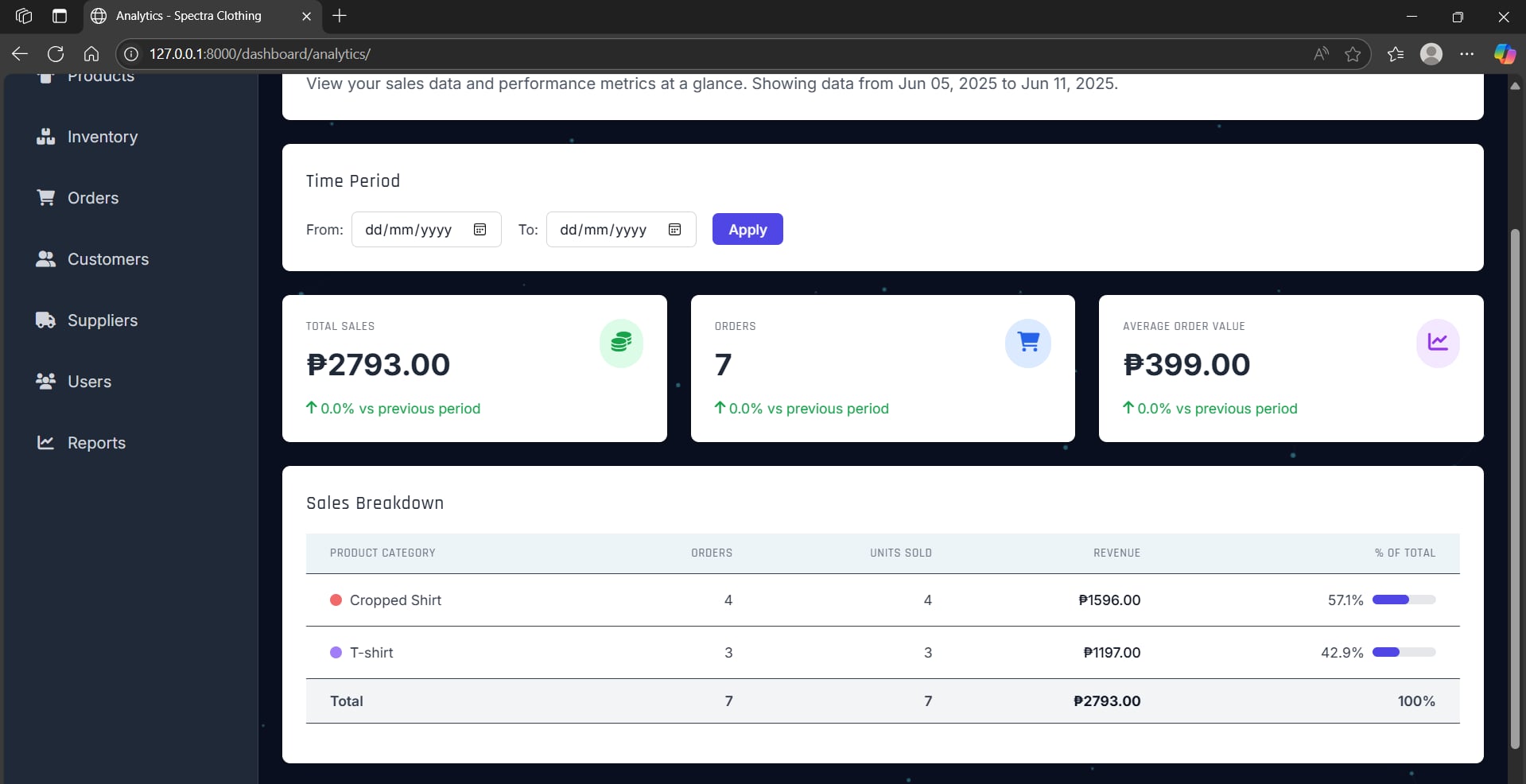
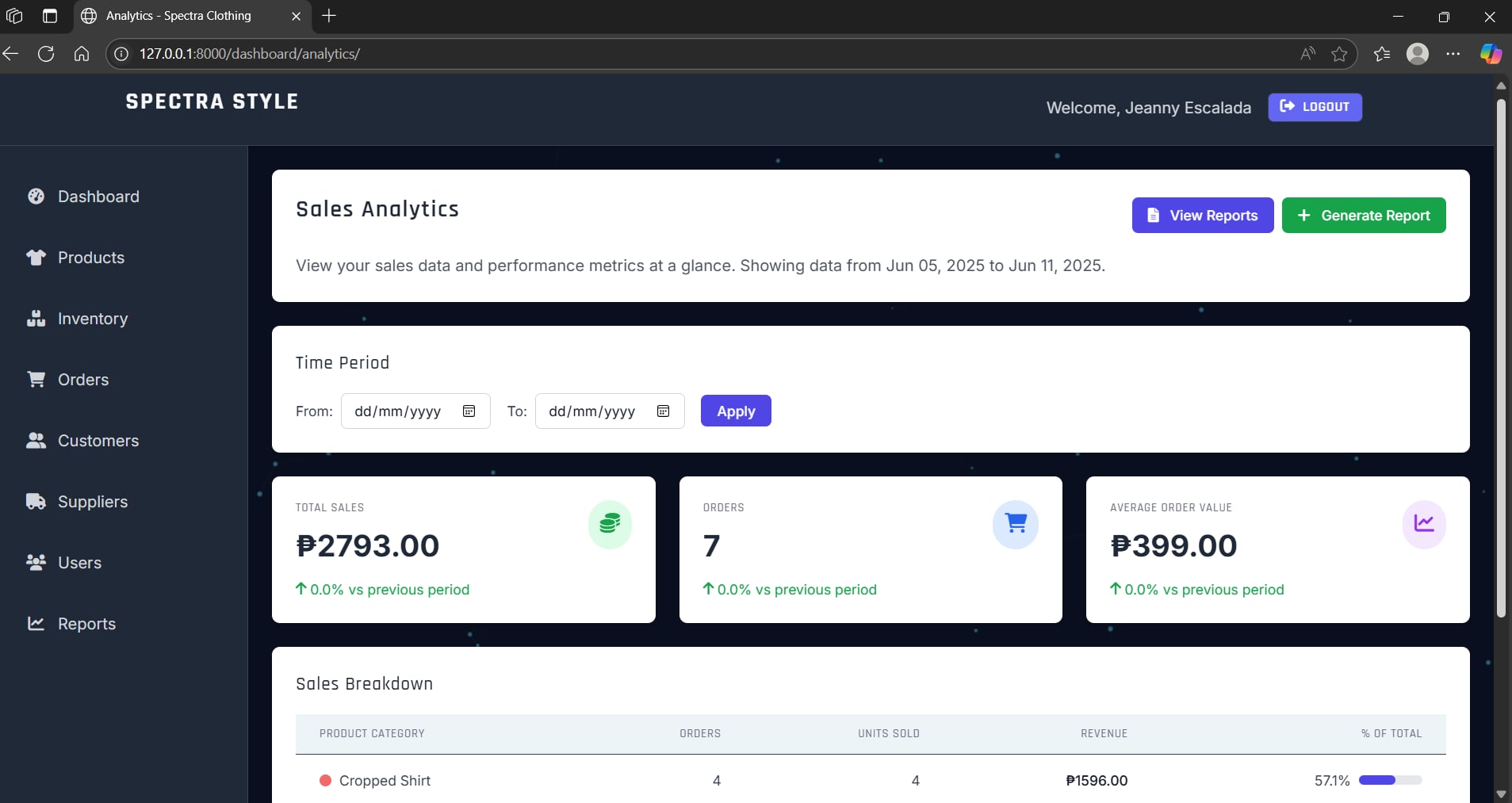


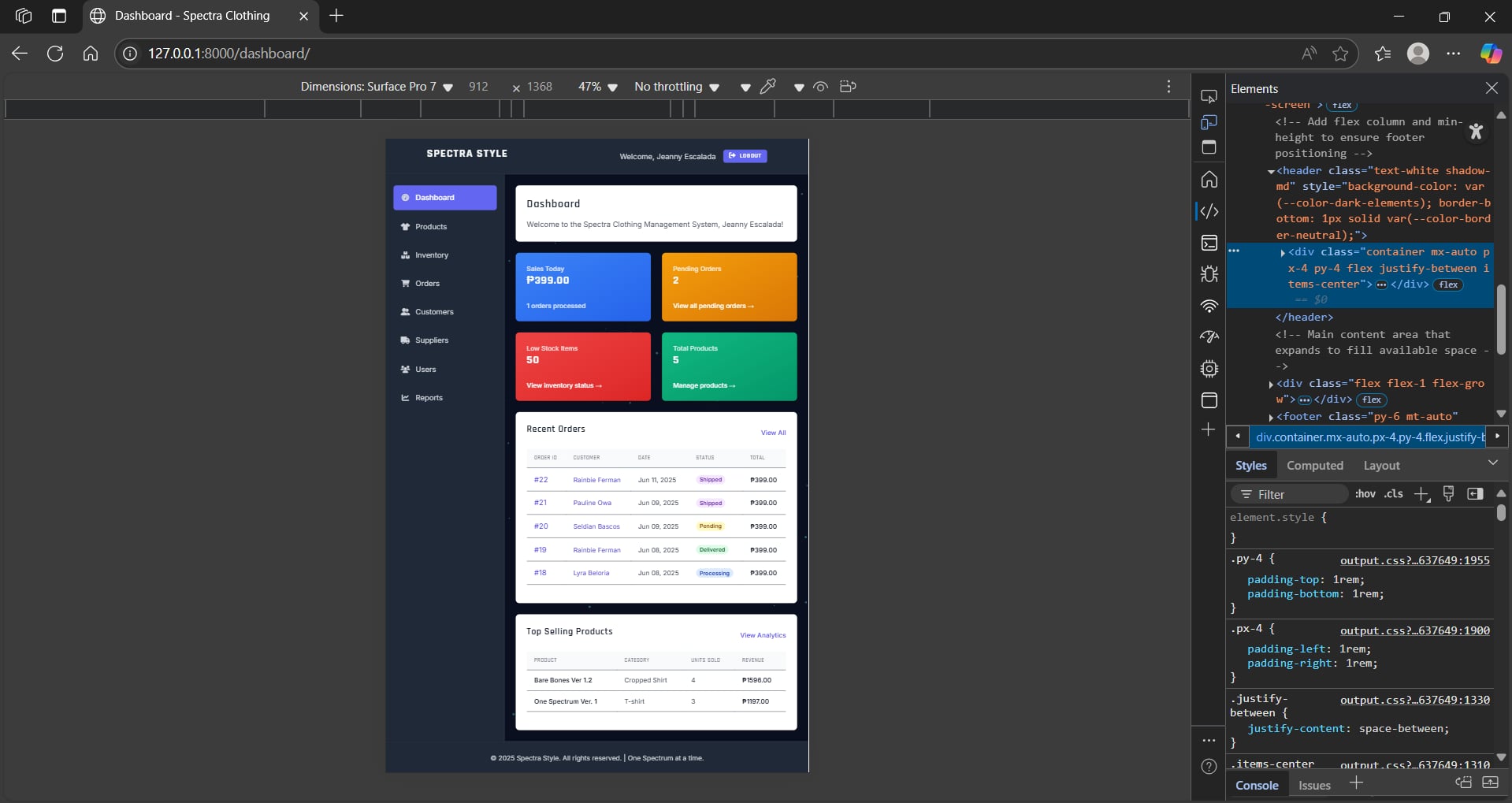












**Appendix F: Individual Member Contribution Sheets**

**Rating Checklist Template:**

**Team Member Name:** Gonzales, Marc Jason G. – BSIT 2C

**I. Task Completion & Quality (1 pt each item)**

* [ ] **Completed Assigned Tasks:** Consistently completed their assigned main tasks and subtasks as outlined in Asana.
* [ ] **Met Deadlines:** Delivered their work on time according to the project timeline.
* [ ] **Quality of Work:** Produced high-quality work that met the project's standards and requirements (e.g., well-written code, clear documentation, effective designs).
* [ ] **Attention to Detail:** Showed thoroughness and accuracy in their work, minimizing errors.
* [ ] **Problem Solving:** Actively identified and contributed to solving challenges or roadblocks encountered during the project.

**II. Collaboration & Communication  (1 pt each item)**

* [ ] **Active Participation:** Actively engaged in team meetings, discussions, and decision-making processes.
* [ ] **Effective Communication:** Communicated clearly and respectfully with team members, teachers, and mock clients (when applicable).
* [ ] **Responsiveness:** Responded promptly to team messages, inquiries, and requests for feedback.
* [ ] **Shared Information:** Proactively shared relevant information, progress updates, and challenges with the team.
* [ ] **Conflict Resolution:** Contributed positively to resolving any disagreements or conflicts within the group.

**III. Initiative & Proactiveness  (1 pt each item)**

* [ ] **Took Initiative:** Volunteered for tasks, identified areas for improvement, or started work without constant prompting.
* [ ] **Proactive Problem Identification:** Anticipated potential issues and brought them to the team's attention early.
* [ ] **Learned New Skills:** Demonstrated a willingness to learn new tools, technologies, or concepts required for their role.
* [ ] **Beyond Assigned Tasks:** Contributed to areas outside their primary role when needed to support the team.

**IV. Role-Specific Contributions (Based on Assigned Role)  (2 pts each item)**

* **If Quality Analyst and Tester:**
  + [ ] Developed comprehensive test plans and test cases.
  + [ ] Thoroughly executed various types of testing (unit, integration, system, UAT).
  + [ ] Provided clear and constructive bug reports.
* **If System Analyst:**
  + [ ] Accurately gathered and documented functional and non-functional requirements.
  + [ ] Translated business needs into clear system specifications.
  + [ ] Ensured system design aligned with client requirements.
* **If Researcher:**
  + [ ] Conducted thorough research on emerging technologies or specific project needs.
  + [ ] Provided valuable insights and data to support project decisions.
  + [ ] Effectively communicated research findings to the team.

**Additional Comments:**

**Rating Checklist Template:**

**Team Member Name:** Laher, Allen Jay B. – BSIT 2B

**I. Task Completion & Quality (1 pt each item)**

* [ ] **Completed Assigned Tasks:** Consistently completed their assigned main tasks and subtasks as outlined in Asana.
* [ ] **Met Deadlines:** Delivered their work on time according to the project timeline.
* [ ] **Quality of Work:** Produced high-quality work that met the project's standards and requirements (e.g., well-written code, clear documentation, effective designs).
* [ ] **Attention to Detail:** Showed thoroughness and accuracy in their work, minimizing errors.
* [ ] **Problem Solving:** Actively identified and contributed to solving challenges or roadblocks encountered during the project.

**II. Collaboration & Communication  (1 pt each item)**

* [ ] **Active Participation:** Actively engaged in team meetings, discussions, and decision-making processes.
* [ ] **Effective Communication:** Communicated clearly and respectfully with team members, teachers, and mock clients (when applicable).
* [ ] **Responsiveness:** Responded promptly to team messages, inquiries, and requests for feedback.
* [ ] **Shared Information:** Proactively shared relevant information, progress updates, and challenges with the team.
* [ ] **Conflict Resolution:** Contributed positively to resolving any disagreements or conflicts within the group.

**III. Initiative & Proactiveness  (1 pt each item)**

* [ ] **Took Initiative:** Volunteered for tasks, identified areas for improvement, or started work without constant prompting.
* [ ] **Proactive Problem Identification:** Anticipated potential issues and brought them to the team's attention early.
* [ ] **Learned New Skills:** Demonstrated a willingness to learn new tools, technologies, or concepts required for their role.
* [ ] **Beyond Assigned Tasks:** Contributed to areas outside their primary role when needed to support the team.

**IV. Role-Specific Contributions (Based on Assigned Role)  (2 pts each item)**

* **If Developer (Back-End/Front-End):**
  + [ ] Wrote clean, efficient, and well-commented code.
  + [ ] Successfully implemented required functionalities (CRUD, Login/Logout, UI interactivity).
  + [ ] Actively participated in debugging and troubleshooting.
* **If System Analyst:**
  + [ ] Accurately gathered and documented functional and non-functional requirements.
  + [ ] Translated business needs into clear system specifications.
  + [ ] Ensured system design aligned with client requirements.
* **If Researcher:**
  + [ ] Conducted thorough research on emerging technologies or specific project needs.
  + [ ] Provided valuable insights and data to support project decisions.
  + [ ] Effectively communicated research findings to the team.

**Additional Comments:**

**Rating Checklist Template:**

**Team Member Name:** Vega, Erika Shigella L. – BSIT 2A

**I. Task Completion & Quality (1 pt each item)**

* [ ] **Completed Assigned Tasks:** Consistently completed their assigned main tasks and subtasks as outlined in Asana.
* [ ] **Met Deadlines:** Delivered their work on time according to the project timeline.
* [ ] **Quality of Work:** Produced high-quality work that met the project's standards and requirements (e.g., well-written code, clear documentation, effective designs).
* [ ] **Attention to Detail:** Showed thoroughness and accuracy in their work, minimizing errors.
* [ ] **Problem Solving:** Actively identified and contributed to solving challenges or roadblocks encountered during the project.

**II. Collaboration & Communication  (1 pt each item)**

* [ ] **Active Participation:** Actively engaged in team meetings, discussions, and decision-making processes.
* [ ] **Effective Communication:** Communicated clearly and respectfully with team members, teachers, and mock clients (when applicable).
* [ ] **Responsiveness:** Responded promptly to team messages, inquiries, and requests for feedback.
* [ ] **Shared Information:** Proactively shared relevant information, progress updates, and challenges with the team.
* [ ] **Conflict Resolution:** Contributed positively to resolving any disagreements or conflicts within the group.

**III. Initiative & Proactiveness  (1 pt each item)**

* [ ] **Took Initiative:** Volunteered for tasks, identified areas for improvement, or started work without constant prompting.
* [ ] **Proactive Problem Identification:** Anticipated potential issues and brought them to the team's attention early.
* [ ] **Learned New Skills:** Demonstrated a willingness to learn new tools, technologies, or concepts required for their role.
* [ ] **Beyond Assigned Tasks:** Contributed to areas outside their primary role when needed to support the team.

**IV. Role-Specific Contributions (Based on Assigned Role)  (2 pts each item)**

* **If Project Manager:**
  + [ ] Effectively managed the Asana project (tasks, deadlines, assignments).
  + [ ] Maintained clear communication with all stakeholders.
  + [ ] Kept the project on track and within scope.
* **If Developer (Back-End/Front-End):**
  + [ ] Wrote clean, efficient, and well-commented code.
  + [ ] Successfully implemented required functionalities (CRUD, Login/Logout, UI interactivity).
  + [ ] Actively participated in debugging and troubleshooting.
* **If System Analyst:**
  + [ ] Accurately gathered and documented functional and non-functional requirements.
  + [ ] Translated business needs into clear system specifications.
  + [ ] Ensured system design aligned with client requirements.
* **If Researcher:**
  + [ ] Conducted thorough research on emerging technologies or specific project needs.
  + [ ] Provided valuable insights and data to support project decisions.
  + [ ] Effectively communicated research findings to the team.

**Additional Comments:**

**Rating Checklist Template:**

**Team Member Name:** Villagracia, Manuel Gabriel R. – BSIT 2A

**I. Task Completion & Quality (1 pt each item)**

* [ ] **Completed Assigned Tasks:** Consistently completed their assigned main tasks and subtasks as outlined in Asana.
* [ ] **Met Deadlines:** Delivered their work on time according to the project timeline.
* [ ] **Quality of Work:** Produced high-quality work that met the project's standards and requirements (e.g., well-written code, clear documentation, effective designs).
* [ ] **Attention to Detail:** Showed thoroughness and accuracy in their work, minimizing errors.
* [ ] **Problem Solving:** Actively identified and contributed to solving challenges or roadblocks encountered during the project.

**II. Collaboration & Communication  (1 pt each item)**

* [ ] **Active Participation:** Actively engaged in team meetings, discussions, and decision-making processes.
* [ ] **Effective Communication:** Communicated clearly and respectfully with team members, teachers, and mock clients (when applicable).
* [ ] **Responsiveness:** Responded promptly to team messages, inquiries, and requests for feedback.
* [ ] **Shared Information:** Proactively shared relevant information, progress updates, and challenges with the team.
* [ ] **Conflict Resolution:** Contributed positively to resolving any disagreements or conflicts within the group.

**III. Initiative & Proactiveness  (1 pt each item)**

* [ ] **Took Initiative:** Volunteered for tasks, identified areas for improvement, or started work without constant prompting.
* [ ] **Proactive Problem Identification:** Anticipated potential issues and brought them to the team's attention early.
* [ ] **Learned New Skills:** Demonstrated a willingness to learn new tools, technologies, or concepts required for their role.
* [ ] **Beyond Assigned Tasks:** Contributed to areas outside their primary role when needed to support the team.

**IV. Role-Specific Contributions (Based on Assigned Role)  (2 pts each item)**

* **If UI Designer:**
  + [ ] Created intuitive and aesthetically pleasing designs (wireframes, mockups).
  + [ ] Ensured designs aligned with user experience principles and Tailwind CSS best practices.
  + [ ] Collaborated effectively with front-end developers for implementation.
* **If Researcher:**
  + [ ] Conducted thorough research on emerging technologies or specific project needs.
  + [ ] Provided valuable insights and data to support project decisions.
  + [ ] Effectively communicated research findings to the team.

**Additional Comments:**

**Rating Checklist Template:**

**Team Member Name:** Yap, Fileteo II L. – BSIT 2B

**I. Task Completion & Quality (1 pt each item)**

* [ ] **Completed Assigned Tasks:** Consistently completed their assigned main tasks and subtasks as outlined in Asana.
* [ ] **Met Deadlines:** Delivered their work on time according to the project timeline.
* [ ] **Quality of Work:** Produced high-quality work that met the project's standards and requirements (e.g., well-written code, clear documentation, effective designs).
* [ ] **Attention to Detail:** Showed thoroughness and accuracy in their work, minimizing errors.
* [ ] **Problem Solving:** Actively identified and contributed to solving challenges or roadblocks encountered during the project.

**II. Collaboration & Communication  (1 pt each item)**

* [ ] **Active Participation:** Actively engaged in team meetings, discussions, and decision-making processes.
* [ ] **Effective Communication:** Communicated clearly and respectfully with team members, teachers, and mock clients (when applicable).
* [ ] **Responsiveness:** Responded promptly to team messages, inquiries, and requests for feedback.
* [ ] **Shared Information:** Proactively shared relevant information, progress updates, and challenges with the team.
* [ ] **Conflict Resolution:** Contributed positively to resolving any disagreements or conflicts within the group.

**III. Initiative & Proactiveness  (1 pt each item)**

* [ ] **Took Initiative:** Volunteered for tasks, identified areas for improvement, or started work without constant prompting.
* [ ] **Proactive Problem Identification:** Anticipated potential issues and brought them to the team's attention early.
* [ ] **Learned New Skills:** Demonstrated a willingness to learn new tools, technologies, or concepts required for their role.
* [ ] **Beyond Assigned Tasks:** Contributed to areas outside their primary role when needed to support the team.

**IV. Role-Specific Contributions (Based on Assigned Role)  (2 pts each item)**

* **If Quality Analyst and Tester:**
  + [ ] Developed comprehensive test plans and test cases.
  + [ ] Thoroughly executed various types of testing (unit, integration, system, UAT).
  + [ ] Provided clear and constructive bug reports.
* **If System Analyst:**
  + [ ] Accurately gathered and documented functional and non-functional requirements.
  + [ ] Translated business needs into clear system specifications.
  + [ ] Ensured system design aligned with client requirements.

**Additional Comments:**