Development of the Temmu-Based application

Nelson Navarro, Jefferson Rico
Dept. of Computer Engineering
Universidad Distrital Francisco José de Caldas
Email: {ndjesusn, jdricor}@udistrital.edu.co

***Abstract*—Online marketplaces have transformed how people buy and sell goods, emphasizing accessibility and user experience. This paper presents the simulation of a marketplace system inspired by Temu, designed as an academic project using Java. The system enables role-based user access, product listing, browsing, filtering, and simulated purchases. Results show that the system successfully implements the core logic of a marketplace, offering a solid foundation for future expansions such as real payment integration or logistics.**

***Index Terms*—Marketplace Simulation, Java Application, Software Engineering, System Architecture, Temu-inspired App**

I. Introduction

The rise of e-commerce platforms has transformed how consumers interact with products and sellers, giving rise to large-scale digital marketplaces such as Amazon, Shopee, and Temu [1], [2]. These platforms allow vendors to publish products and users to explore, compare, and purchase items in an intuitive way. Beyond the commercial scope, the architecture and logic of these systems offer valuable insights for computer engineering students [3], especially in terms of user experience, modular design, and role-based interfaces.

Temu, in particular, stands out due to its aggressive pricing, gamified experience, and mobile-first interface [4]. Its success illustrates how digital marketplaces can combine traditional e-commerce workflows with dynamic user engagement mechanisms. Understanding such systems requires analyzing both technical and business-related decisions.

This paper focuses on the simulation of a marketplace system inspired by Temu, developed as an academic project. The simulation was implemented using Java and structured to reflect essential marketplace features including user login, product listing, role-specific navigation, filtering options, and simulated purchases. Users can choose between acting as a buyer or a vendor. Vendors can publish and view their listed and sold products, while buyers can explore available products and simulate purchases.

The goal is not only to recreate functional components of a marketplace, but also to understand the software architecture behind it. Through this simulation, we aim to explore the design of modular systems, file-based data persistence, and separation of user roles within a single application [5].

Additionally, this work allows students to apply theoretical knowledge in a real development environment. The process of implementing a system from scratch fosters deeper understanding of UI/UX principles, file handling, and object-oriented design—all fundamental skills in software engineering [6].

Funded by the Software Engineering course project.

II. Methods and Materials

The system was developed using Java in Eclipse IDE, applying OOP principles. The main classes include:

- **User**: base class for all users, including name, email, and password.
- **Seller** and **Buyer**: subclasses with specific methods for posting or purchasing products.
- **Product**: represents items with attributes such as name, price, and category.
- **Controller**: manages business logic and bridges the model and the view.

.

A. Architecture

The MVC architecture separates responsibilities to promote maintainability:

- **Model**: handles user, product, and transaction data.
- **View**: console-based interface for interaction.
- **Controller**: interprets user input and coordinates actions.

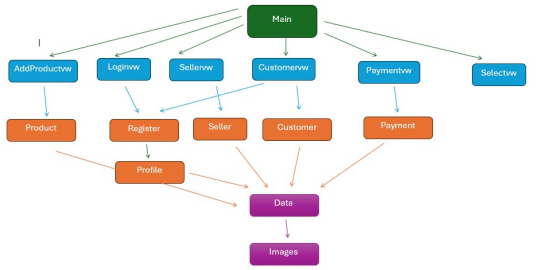

Fig. 1. System architecture following the MVC pattern.

B. Data Persistence

Data is stored in '.txt' files:

- `users.txt`
- `products.txt`
- `sales.txt`

This design avoids the complexity of relational databases while reinforcing file-handling in Java.

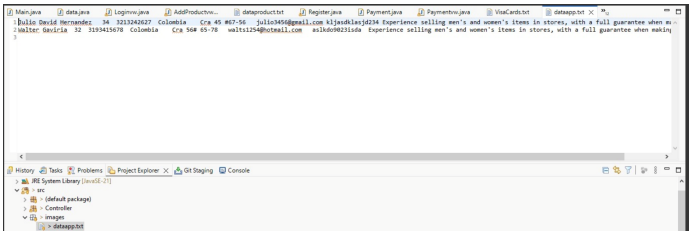

Fig. 2. Database

C. Design

The application was built using Java with Swing for the graphical interface.

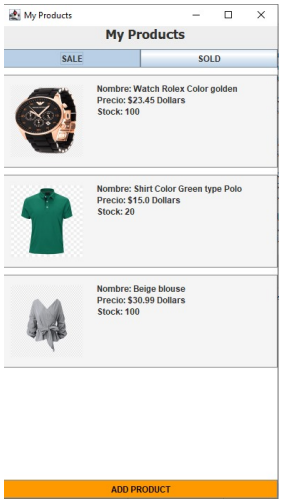

Fig. 3. Main Seller Interface

III. Results

Each feature was manually tested. Table I summarizes functional tests performed:

TABLE I
FUNCTIONAL TESTS OVERVIEW

Function	Input	Expected	Status
Login	Role: Seller	Loads Seller UI	Pass
Publish product	Item info	Display on panel	Pass
Simulate Purchase	Click "Pay Now"	Show confirmation message	Pass
Persistence	Exit app	Data in .txt	Pass

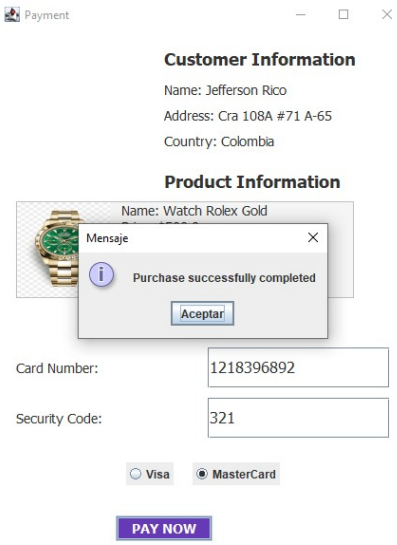

Fig. 4. Simulate Purchase

IV. Conclusions

The development of a marketplace simulation using Java provided valuable experience in system architecture, user interaction modeling, and data persistence strategies. The role-based access control and separation of concerns ensured maintainability and clarity in design.

This academic project demonstrates that essential marketplace logic can be replicated without external dependencies, offering a strong educational foundation. Future work could integrate real-time databases, admin roles, and secure authentication layers.

Acknowledgment

Special thanks to Professor Carlos Sierra for guidance and feedback throughout the project, and to the team members who contributed to design and testing phases.

References

[1] V. Lau, "Shopee's rise in southeast asia: Strategies behind the growth," *E-commerce Journal*, 2021, accessed: 2025-07-09. [Online]. Available: https://www.techinasia.com/shopees-strategy

[2] M. . Company, "Temu and the rise of chinese cross-border e-commerce," 2023, accessed: 2025-07-09. [Online]. Available: https://www.mckinsey.com/industries/retail/our-insights/temu-ecommerce-expansion

[3] A. M. Shuqair, "Modular microservices architecture for scalable e-commerce platforms," *International Journal of Computer Applications*, vol. 178, no. 12, pp. 12–18, 2021.

[4] Statista, "Temu monthly active users worldwide 2024," 2024, accessed: 2025-07-09. [Online]. Available: https://www.statista.com/statistics/temu-active-users/

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[6] R. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 9th ed. McGraw-Hill, 2019.