International Journal of Information Technology and Computer Science Applications (IJITCSA)
p-ISSN 2964-3139 e-ISSN 2985-5330
Vol. 2, No. 1, page 65 - 74
Submitted 30/1/2024; Accepted 05/02/2024; Published 06/02/2024

Design and Build API-Based Multistore Marketplace Using Agile Methodology.

[1]Donny Fernando and [2]Dentik Karyaningsih
[1]Information Systems, Serang Raya University, Banten, INDONESIA
[2]Informatics Department, Serang Raya University, Banten, INDONESIA
e-mail : [1]mr.donny2008@gmail.com, [2]karya_tiek@gmail.com

Corresponding Autor: Donny Fernando

Abstract
The business activities of a company require technology that can drive operational efficiency. Marketplaces have become one of the effective marketing channels today for promoting the company's products and services. To reach a broader market, having more than one store on a marketplace is a choice that many companies currently opt for. However, managing multiple stores on marketplaces is not an efficient management approach. This research is an applied study conducted in one startup company (*profitto.co.id*) that sells developing furniture components, aiming to provide the implementation of an API-based system integration solution to manage multiple stores on *Tokopedia* and utilizes the Scrum methodology for software development.

Keywords—System Development, Application Interface, Marketplace, Scrum

1 Introduction

As per the data presented in the "E-commerce in Southeast Asia 2023" report, Indonesia witnessed a substantial accumulation of consumer expenditure amounting to 51.9 billion US dollars or approximately IDR 773.7 trillion in the year 2022[1]. Tokopedia, identified as a pivotal marketplace, exhibits a highly effective marketing channel for the sale of furniture component products. The strategic adoption of multiple stores within a single marketplace stands out as a chosen approach by companies seeking to extend their product reach across the entirety of the Indonesian market.

However, in managing day-to-day operations, the company faces inefficient processes when handling multiple stores on a marketplace, such as managing prices, inventory, and order management, including order acceptance, receipt printing, and pickup requests. All these processes need to be executed separately for each store, and the transition from one store to another is not efficient, requiring significant effort.

Managing multiple stores on a marketplace can pose various challenges for the company, as highlighted above. Here are specific challenges that businesses may face:

1. **Inefficient Processes**: Handling day-to-day operations like managing prices, inventory, and order management becomes inefficient when each process needs to be executed separately for each store. This can lead to duplication of efforts, increased chances of errors, and overall inefficiency in operations.
2. **Lack of Centralized Control**: The absence of a centralized system for managing multiple stores makes it challenging for the company to have a unified and streamlined approach. Without a central control mechanism, coordinating and implementing changes across different stores becomes a complex task.
3. **Transition Difficulties**: Based on high level management interview, the management mentions that transitioning from one store to another is not efficient and requires significant effort. This could be due to the absence of standardized processes, interfaces, or tools, making it cumbersome for the staff to switch between stores seamlessly.

4. **Price and Inventory Discrepancies**: Managing prices and inventory for multiple stores can result in discrepancies. Without a centralized system, it becomes challenging to ensure consistency in pricing strategies and to maintain accurate and up-to-date inventory levels across all stores.
5. **Order Management Complexity**: Order management involves various tasks such as order acceptance, receipt printing, and pickup requests. When each store operates independently, these processes can become complicated and time-consuming, leading to delays, errors, and customer dissatisfaction.
6. **Scalability Issues**: As the number of stores increases in future, scalability becomes a significant concern. Existing processes and systems are having difficulty handling the ever-increasing volume of operations, potentially causing a decrease in efficiency.

To address these challenges above, this research aims to implement a solution for developing a multi-store management system for Tokopedia based on API, enabling the integration and management of sales channels (multi-store) on Tokopedia through a single interface[2].

This study is conducted in one of the newly established startups engaged in selling a variety of industrial and furniture component needs, ranging from Plywood, MDF, Hardware Fitting, Glue, to Sandpaper.

2 Research methods

This research is an applied study, and the software development methodology utilized in this research is the Scrum Methodology. Scrum is an Agile software development framework that is incremental and iterative, designed to manage software development[3],[4]. Currently recognized as the most popular agile development methodology, Scrum has been applied in various contexts and for diverse purposes, both within and outside the traditional software development context[5],[6].

With Scrum, the product is built in a series of fixed-time iterations known as sprints. A sprint is a fixed time cycle during which the product is developed and delivered for feedback. One sprint is one iteration lasting a month or less, maintaining consistent length throughout the development effort. Only the Product Owner has the authority to cancel a Sprint. Passing milestones – namely, the end of a sprint – occurs regularly and at regular intervals, bringing a tangible sense of progress with each cycle, energizing everyone, providing inspiration to the team, and aiding in the early detection of misunderstandings or incorrect requirements. Short iterations also underscore the importance of accurate estimation, a recurring challenge in software development using waterfall methodologies.

The implementation of the Scrum methodology in developing a multi-store management system with an API-based approach involves several key steps and roles. Scrum is an agile framework that emphasizes iterative development, collaboration, and adaptability. As the result, here's a detailed explanation of how Scrum was applied in this research:

1. Product Backlog Creation:
   - The first step is to create a product backlog, which is a prioritized list of features, enhancements, and tasks. In the context of a multi-store management system, this could include features such as inventory management, order processing, reporting, and API integration.
2. Sprint Planning
   - Before each sprint started, the development team, product owner, and Scrum Master meet for sprint planning.
   - During sprint planning meeting, the team selects a set of items from the product backlog to work on during the upcoming sprint. These items are then broken down into tasks.
   - Sprint planning involves estimating the effort required for each task and committing to completing a set amount of work within the sprint duration.
3. Sprint:
   - The development work is organized into fixed-length iterations called sprints, typically lasting 2-4 weeks. And in this research sprint length is 4 weeks.
4. Daily Stand-ups:
   - The team holds daily stand-up meetings to discuss progress, challenges, and plans for the day. Stand-up meetings are held for a maximum of 15 minutes every day.
5. Sprint Retrospective:
   - At the end of each sprint, a sprint review is conducted to demonstrate the completed work to stakeholders.

- A sprint retrospective is held to reflect on the sprint, discuss what went well, what could be improved, and make adjustments for the next sprint.

The stages of Scrum Methodology are as follows, as shown below figure 1;

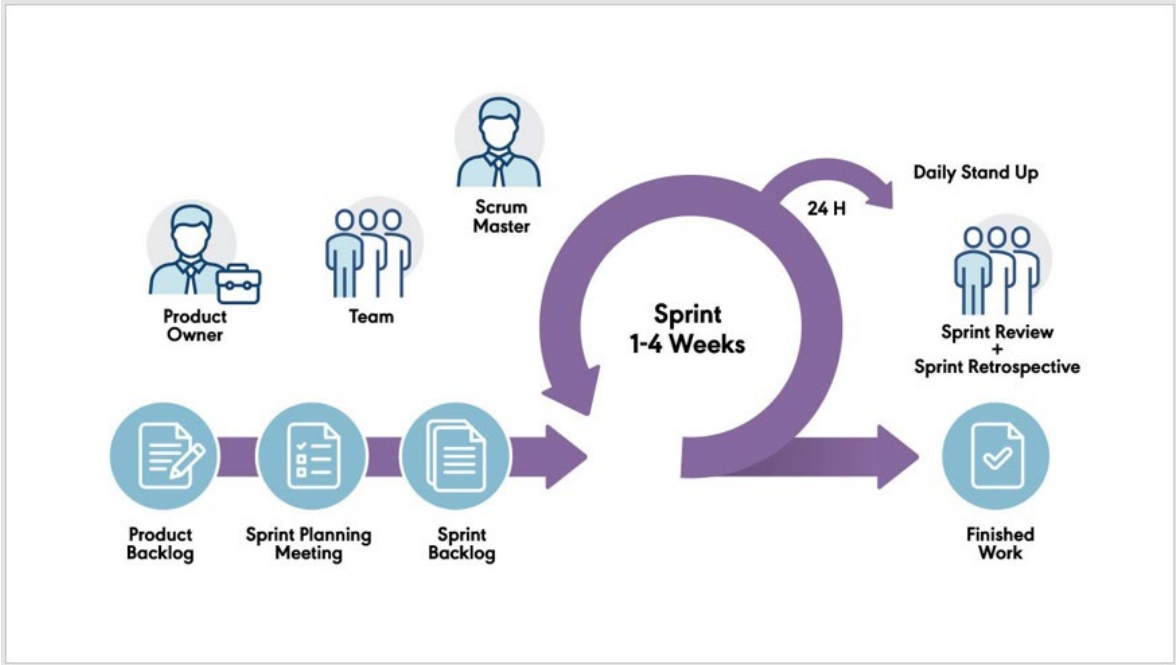


Figure 1. Stage of Scrum Methodology

The Scrum team has 3 specific roles namely Product Owner, Scrum Master and Development Team[3]. In Table 1 below are the responsibilities of each role.

Table 1. Responsibilities of each role

Product Owner	Scrum Master	Development Team
▪ Take on the responsibility of comprehending stakeholder requirements. ▪ Design and manage the Product Backlog, which constitutes a comprehensive list of product requirements to be addressed ▪ Establish task priorities ▪ Actively participate in iterations to provide feedback and ensure the product meets customer expectations	▪ Act as a facilitator for the Scrum team. ▪ Remove obstacles and ensure the team can work effectively. ▪ Ensure the implementation of Scrum principles and maintain focus on sprint goals. ▪ Assist the team in continuously improving their performance through retrospectives	▪ The team member is directly responsible for producing the product. ▪ It is an autonomous and multifunctional team. ▪ Receives and implements tasks from the Product Backlog during the Sprint. ▪ Is accountable for achieving sprint goals and delivering a quality product.

Below are the research steps implemented which can be seen in Figure 2:

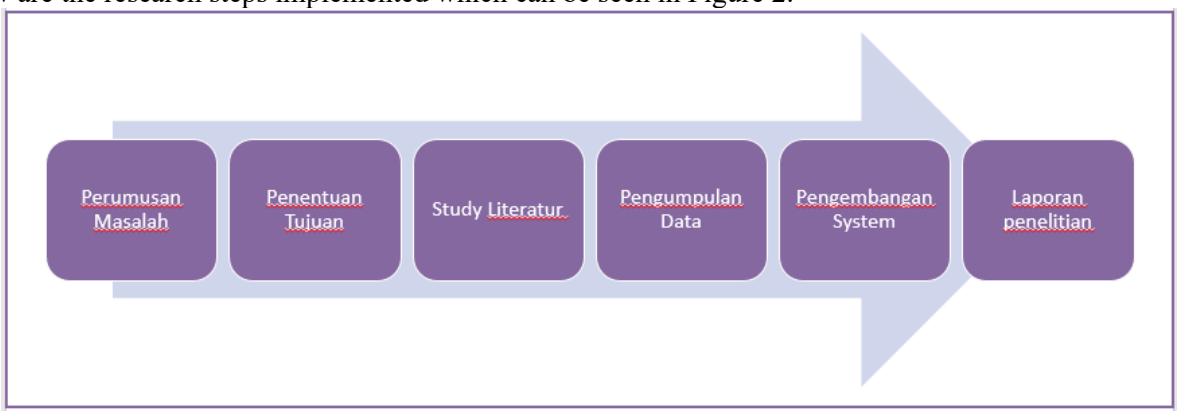


Figure 2. Research Flowchart

This research commenced with collaborative discussions with the company to formulate the problem statement and establish objectives addressing real business problems encountered in day-to-day operations. Subsequently, a literature review was conducted, drawing from various sources such as articles and journals related to the research, to support and augment the study.

Data collection involved conducting interviews to acquire the necessary business requirements associated with the identified issues. Following the data collection phase, system development was undertaken using the Scrum methodology.

The system development process initiated with the design phase, employing Unified Modeling Language (UML) [7]. UML serves as the tool for designing in this research context.

3 Results and Discussion

When determining the Product Backlog, users describe the desired business flow for the application to be developed. Figure 2 below is the High Level Business Flow application that will be developed.

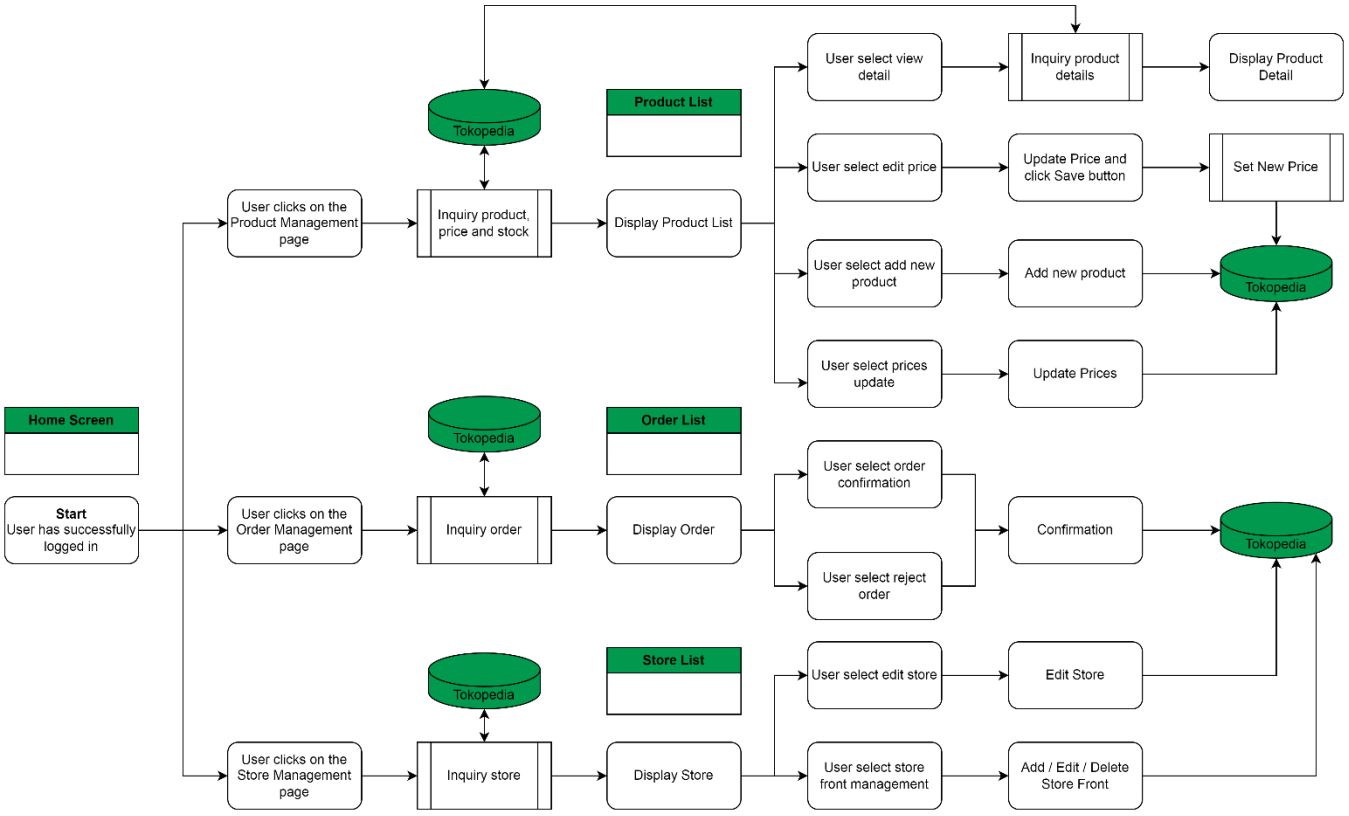


Figure 3. Business Flow

The use case diagram illustrates the primary functionalities and interactions within the Multistore Marketplace. The main actors identified include administrators, and users/employees. Each actor is associated with specific use cases that represent distinct functionalities or tasks they perform within the system.

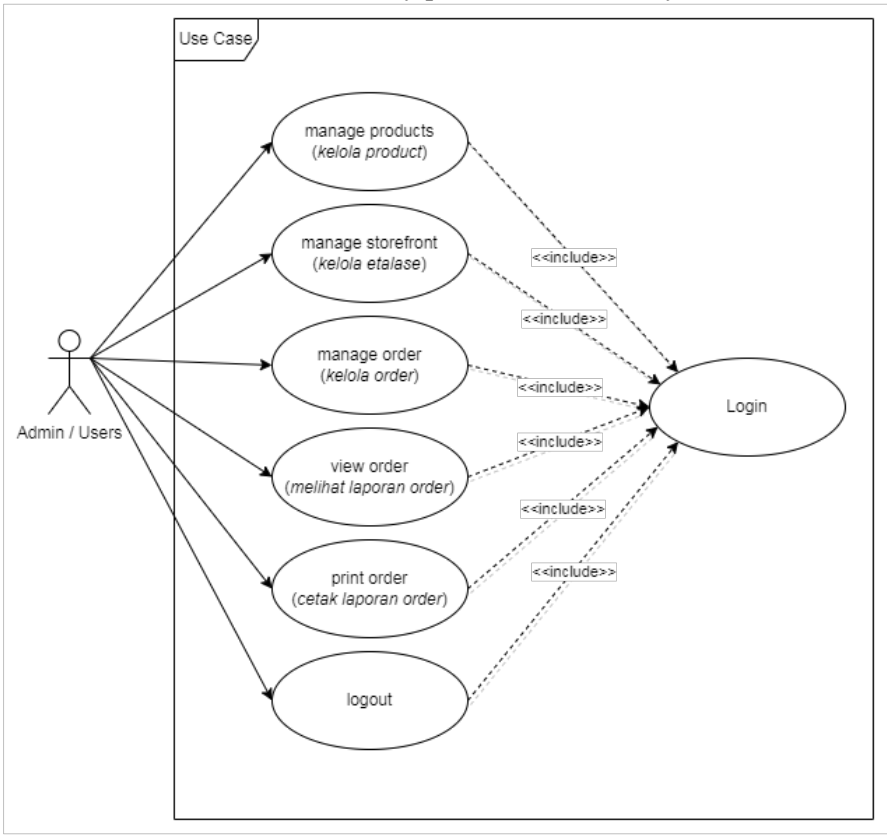


Figure 4. Use Case Diagram

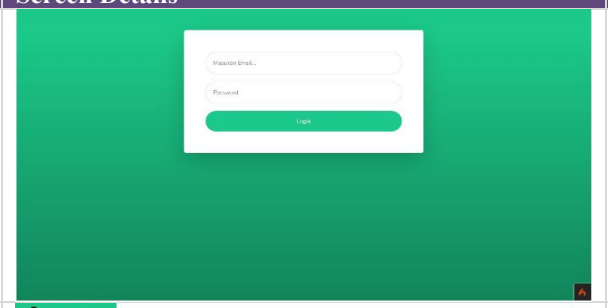
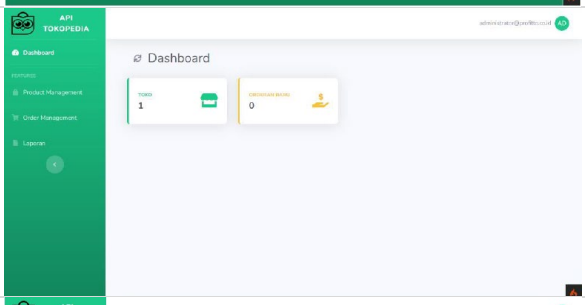
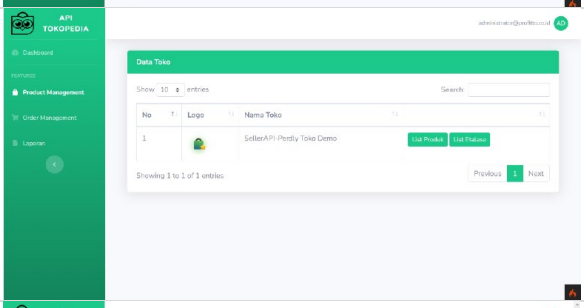
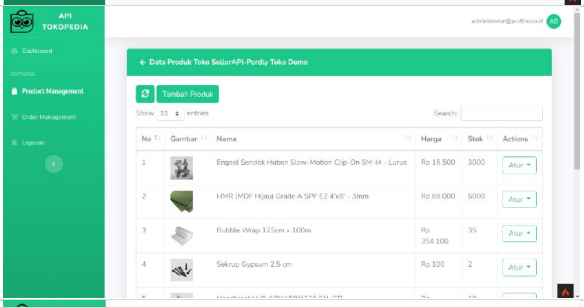
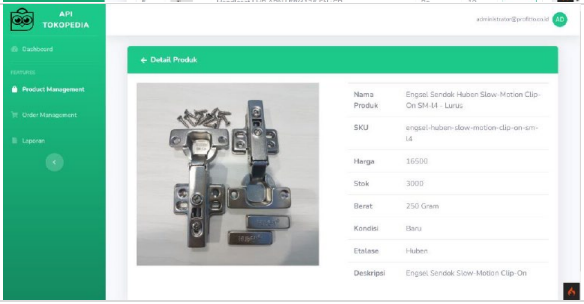
Several roles of actors can be explained below:

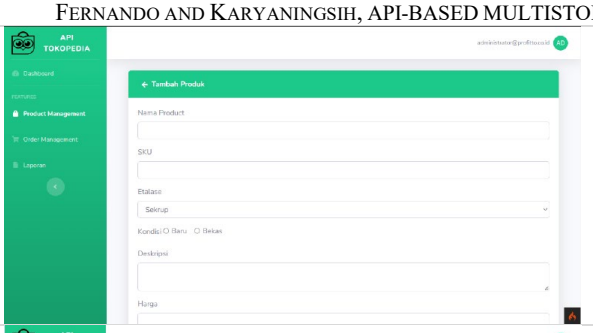
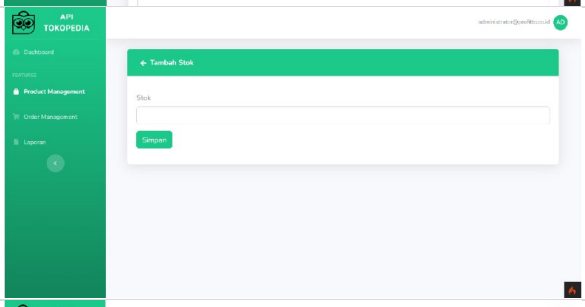
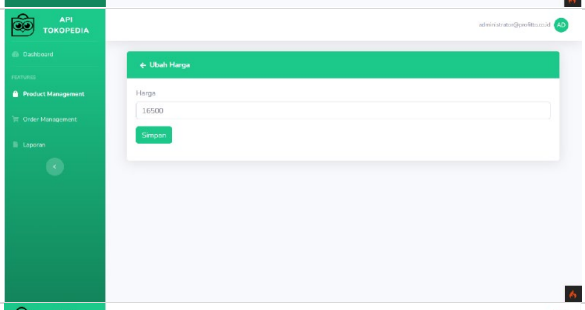
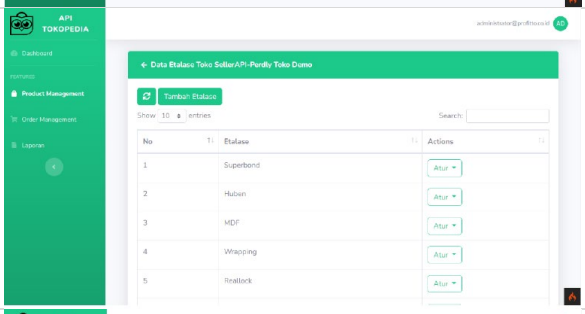
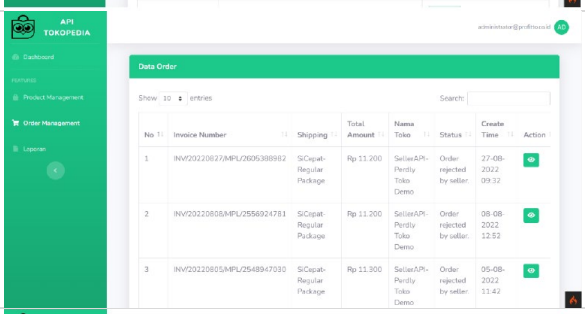
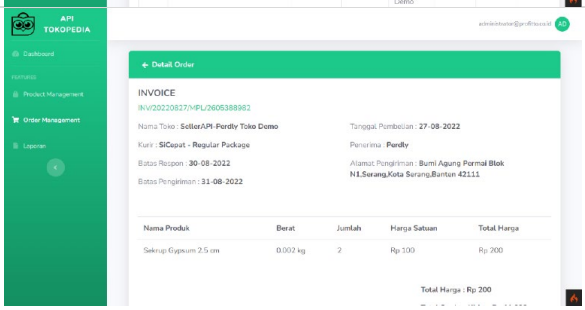
a. **Administrator**
   - **Login**: Allows administrator to access the system securely.
   - **Manage Product**: Allow administrator to add, update, delete products, add stock or change price.
   - **Manage StoreFront**: Allow administrator to add, update or delete storefront.
   - **Manage Order**: Allow administrator to accept orders or reject orders, view order, confirm delivery, request pickup, and print delivery receipts.

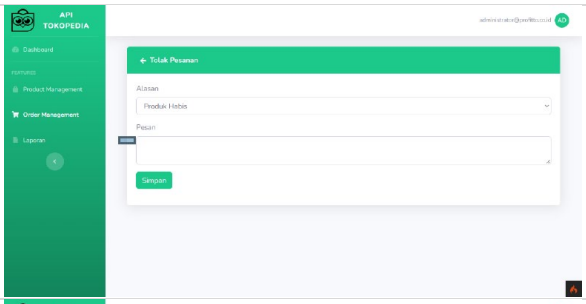
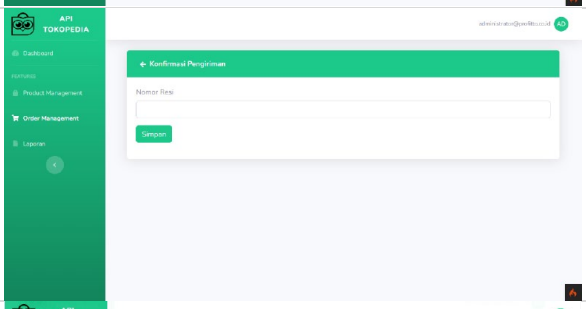
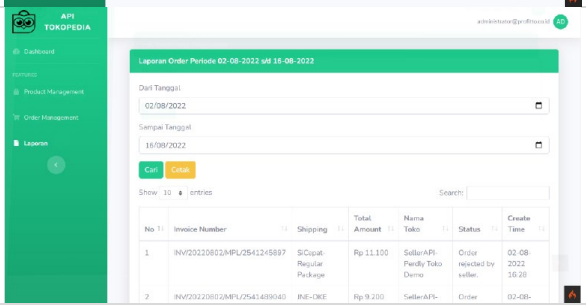
b. **Users** or **Employees**
   - **Login**: Allows users to access the system securely.
   - **Manage Product**: Allow users to view product and price.
   - **Manage StoreFront**: Allow users to view storefront.
   - **Manage Order**: Allow users to accept orders, view order, confirm delivery, request pickup, and print delivery receipts.

The Use Case diagram functions as a blueprint for the application interface, and Table 2 below is a detailed breakdown of the screens that will materialize the user interactions outlined in the Use Case diagram.

Table 2. Screen Details

No	Screen Name	Screen Details	Screen ID	Screen Description
1	Login		*loginScreenId*	The Login screens are used to allow users to access a secure area of this application. It's required a username and password.
2	Home Screen		*homeScreenId*	The home screen is the main content area of this application. On it, users can also view the dashboard.
3	Store List		*storeList ScreenId*	This store list is used to view your store list.
4	Product List		*productList ScreenId*	This screen is used to display a list of products
5	Product Detail		*productList DetailScreenId*	This screen is used to display detailed product information

6	Add Product		*addProduct ScreenId*	This screen is used to manage products. Users can add, edit and delete products on this page
7	Add Stock		*addStock ScreenId*	This screen is used to manage product stock.
8	Price Update		*priceUpdate ScreenId*	This screen is used to manage product prices.
9	Store Front Management		*storefront ScreenId*	Screen is used to manage the showcase. Here users can edit and delete storefronts.
10	Order List		*orderList ScreenId*	This screen contains data on incoming orders from managed shops.
11	Order List Detail		*orderListDetail ScreenId*	This screen contains detailed data on incoming orders. Here the user can accept the order or reject the order, confirm delivery, request pickup, and print a delivery receipt according to the authorization they have.

12	Order Reject		*orderReject ScreenId*	This screen is used to reject incoming orders.
13	Order Confirmation		*orderConfirmation ScreenId*	This screen is used to confirm orders
14	ViewOrder		*orderView ScreenId*	This page contains reports of incoming orders per certain period, here admins or users can print order reports for a certain period.

To facilitate the functionalities of the aforementioned screens, Table 3 outlines the corresponding back-end APIs that serve as the backbone of data exchange." (Emphasizes the enabling role of APIs).

Table 3. List of API

No	API Description	Method	Query Parameter	Parameter Value	Details
1	mp/v1/dashboard	GET			API used to get the contents of dashboard
2	mp /v1/store	GET			API used to get the contents of the list of stores
3	mp/v1/storefront	GET			API used to get the contents of the list of storefronts
4	mp/v1/storefront	POST	subType	add, edit, delete	API used to manage the storefront
5	mp/v1/product	GET			
6	mp/v1/product/add	POST	subType	add, edit, delete	API used to manage product
7	mp/v1/product/detail-list	GET			API used to get detail list of product information
8	mp/v1/stock/edit	POST			API used to update stock of product
9	mp/v1/price/edit	POST			API used to update price of product

Table 2 outlined the various screens users will interact with. Now, let's explore how these screens communicate with the back-end using APIs (Application Programming Interfaces). Table 3 provides a list of these APIs, but understanding their interaction requires a peek into the communication flow:

1. User Initiates Action:
    - User interacts with a specific element on a screen (e.g., clicks a button, submits a form).
    - Front-end code captures this action and translates it into an API request.
2. Crafting the API Request
    - The front-end code determines the appropriate API endpoint (URL) based on the user's action and intended functionality.
    - It gathers relevant data from the user input or application state (e.g., form data, authentication tokens).
    - This data is formatted according to the API's specifications (we used JSON).
3. Sending the Request
    - The front-end code send the request to the back-end server hosting the targeted API endpoint.
4. Back-End Receives and Processes
    - The back-end server receives the request, parses the data, and authenticates the user.
    - Based on the API endpoint and the data received, the back-end executes the relevant logic, potentially interacting with the database, performing calculations, or accessing other services. (Detailed logic is depicted in the Sequence Diagram)
5. Sending the Response
    - The back-end prepares a response containing the requested data, error messages, or updated application state.
    - This response is formatted according to the API's specifications.
6. Front-End Receives and Updates
    - The front-end code receives the response and parses the data.

4 Conclusion

In summary, the conducted research affirms that a multi-store management system, skillfully designed and developed with API integration for multiple stores on Tokopedia, has been successfully implemented. The seamless utilization of this system has improved productivity and efficiency of the company's business operations.

5 Suggestion

The current research has provided valuable insight into markets with many stores; however, further investigation is needed to explore specific aspects, explore potential correlations, and address knowledge gaps. Future research could also focus on the security of data exchange using APIs so as to improve our understanding of microservices. This research aims not only to expand the theoretical framework but also offers practical implications for companies post-implementation such as the ease felt by users when handling day to day operations.

6 Acknowledgments

The author would like to thank,

1. Serang Raya University for providing financial and non-financial support for this research.
2. PT. Profitto Invonasi Kreatif for collaboration and support for this research.

BIBLIOGRAPHY

[1] W. K. P. Galuh Putri Riyanto, "6 Marketplace Terbesar di Indonesia Tahun 2022, Shopee Terata," *16 Jun 2023*, 2023. https://tekno.kompas.com/read/2023/06/16/19300027/6-marketplace-terbesar-di-indonesia-tahun-2022-shopee-teratas.

[2] I. S. dan T. T. S. Winda Angelina Utama (Sistem Informasi, Institut Sains dan Teknologi Terpadu Surabaya), Hartarto Junaedi (Sistem Informasi, "Manajemen Produk dan Pesanan untuk Multichannel E-Commerce Menggunakan Framework Laravel, API Tokopedia dan API Lazada," *J. Inf. Syst. Graph. Hosp. Technol.*, vol. 05, pp. 32–38, 2023.

[3] S. Sachdeva, "Scrum Methodology," *Int. J. Eng. Comput. Sci.*, vol. 5, no. 16792, pp. 16792–16800, 2016, doi: 10.18535/ijecs/v5i6.11.

[4] J. Betta, T. Chlebus, and D. Kuchta, *Applying Scrum in New Product Development Process*, vol. 1. Springer International Publishing, 2019.

[5] S. Hron, Michal (Aarhus University, Fuglesangs Allé 4, 8210 Aarhus V, Denmark), Nikolaous Obwegeser (Bern University of Applied Sciences, Brückenstr. 73, 3005, Bern, "Why and how is Scrum being adapted in practice: A systematic review," *ScienceDirect*, vol. 183, 2022, doi: https://doi.org/10.1016/j.jss.2021.111110.

[6] V. Hema, S. Thota, S. N. Kumar, C. Padmaja, and C. Bala, "Scrum : An Effective Software Development Agile Tool Scrum : An Effective Software Development Agile Tool," *Springer*, 2020, doi: 10.1088/1757-899X/981/2/022060.

[7] R. Fauzan, "Use Case Diagram Similarity Measurement : A New," *2019 12th Int. Conf. Inf. Commun. Technol. Syst.*, pp. 3–7, 2019.