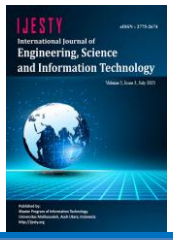
International Journal of Engineering, Science and InformationTechnology
Volume 5, No. 3 (2025) pp. 327-336
eISSN: 2775-2674
Website: http://ijesty.org/index.php/ijesty
DOI: https://doi.org/10.52088/ijesty.v5i3.994
Research Paper, Short Communication, Review, Technical Paper



Optimization of LPG Distribution for a Multiplatform-Based LPG Marketplace

Herman Budianto*, Farhan Faisal Zainul Mustaqin, Esther Irawati Setiawan, Joan Santoso

Department of Informatics, Institut Sains dan Teknologi Terpadu Surabaya, Surabaya, Indonesia

**Corresponding author Email: herman.budianto@istts.ac.id*

The manuscript was received on 26 January 2025, revised on 28 February 2025, and accepted on 10 June 2025, date of publication 23 June 2025

Abstract

Marketplace applications have become an essential digital solution supporting online transactions, including LPG distribution. The development of this application adopts a multiplatform approach, enabling the application to run on various devices, particularly Android platforms and websites. Using the React Native framework, developers can build applications with a single, efficient codebase for multiple platforms. This study aims to provide users with convenience in purchasing LPG without leaving their homes while offering a more practical and effective user experience. This research includes features for selling, buying, payment, and delivery via courier. The transaction feature facilitates sellers' recording of sales within the application. The results of alpha testing indicate that the Elpijiku marketplace app works well despite some significant errors or bugs. However, acceptance testing results were very positive, with 91% of respondents rating the application and user experience as good. These findings indicate that the Elpijiku application meets user needs in terms of convenience and efficiency and is suitable for use as a digital solution for LPG distribution.

Keywords: *Marketplace, Multiplatform Application, React Native.*

1. Introduction

The primary needs of every human being are clothing, food, and shelter. Clothing refers to basic human garments, food pertains to essential needs related to eating and drinking, and shelter relates to housing requirements. One key resource that supports food preparation is LPG, a standard cooking fuel utilized across a broad range of users—from households to street vendors. Although LPG is relatively easy to find, individuals with busy schedules or students in new surroundings may face difficulties purchasing it efficiently.

The ELPIJIKU platform was developed as an innovative, Android-based solution to address this challenge. The application is built by leveraging React Native technology, ensuring seamless performance and enabling future expansion to web [1], [2] and iOS platforms from a unified codebase with minimal effort.

ELPIJIKU's speciality lies in its intelligent, distributed logistics system, which is architected to handle complex, real-time operations. This system moves beyond simple order-taking by implementing a dynamic dispatch algorithm that automatically assigns incoming orders to the most optimal seller based on proximity, availability, and delivery load. For sellers, the system provides an automated queue management and inventory tracking system, reducing manual workload and preventing stockouts.

Furthermore, its core distribution feature utilizes geospatial data to instantly and accurately calculate delivery fees based on the buyer's precise location[3]. Buyers are empowered with the convenience of ordering from their mobile devices and gain complete visibility through real-time delivery tracking. By focusing on a robust distributed architecture, cross-platform compatibility, and real-time logistics, ELPIJIKU is not just another ordering app—it's a modern solution designed to simplify daily living and create a more resilient and efficient LPG supply chain for both consumers and sellers.

This article consists of five parts. The first section discusses the introduction. The second part explains the literature review and the third part describes the methods used. The fourth section explains the results and discussion, and the final section discusses the conclusions.

2. Literature Review

Web-based systems are frequently developed to address specific domain challenges. Hariyanto et al. [1] designed a prototyping system to monitor student final projects, improving administrative efficiency and transparency. In the healthcare sector, Salat et al. [2] built a rabies diagnosis expert system that uniquely combines Forward Chaining and Dempster-Shafer methods to handle data uncertainty. Both

studies demonstrate the merit of using web technology to create accessible and specialized digital solutions for management and diagnostics.

Choosing a suitable framework is critical in mobile development, and several studies offer valuable guidance. Zou and Darus [4] provided a comparative analysis of cross-platform frameworks, creating a matrix to aid developers in decision-making. Additionally, using empirical data, Kishore et al. [5] delivered a quantitative performance comparison between React and Flutter. At the same time, the case study by De Almeida et al. [6] offers real-world insights into the practical maintenance and performance challenges encountered during development.

The versatility of cross-platform technology is evident in its application across different fields. For Example, Setiawan et al. [3] developed a user-friendly waste management application using a prototyping methodology to support the circular economy. In another practical implementation, Hardjanto et al. [7] created a goods storage rental application with the React Native framework, highlighting the efficiency of building a functional app for multiple operating systems from a single codebase.

Recent research shows a trend toward creating specialized e-marketplaces tailored to community needs. Coltey et al. [8] proposed an AI-driven marketplace to enhance small business competitiveness, while Prasitsupparote et al. [9] designed a local food platform with integrated marketing strategies. Focusing on agriculture, Nucos et al. [10] conceived a marketplace with fair pricing regulation, and the survey by Salih et al. [11] explores the broader application of this concept to industrial Internet of Things (IoT) ecosystems.

The architecture and application of Node.js remain a key area of study for building scalable backends. Chaplin and Klym [12] designed a layered, modular project architecture for RESTful microservices, enhancing maintainability compared to monolithic structures. Demonstrating a practical application, Prasad et al. [13] utilized Node.js and the Web Speech API to develop an interactive AI chatbot, showcasing the framework's capability to power both robust backend systems and dynamic user-facing features.

Modern cloud and serverless architectures offer flexible and scalable backend solutions. Athreya et al. [14] demonstrated the viability of this model by implementing a serverless e-commerce mobile application. To guide technology choices, Varshitha et al. [15] built a real-time blood bank communication system using Supabase for the backend, while Azkiya et al. [16] showed a practical integration of Google App Engine and Cloud Storage to build a REST API for an innovative farming application.

Advancements in black-box testing aim to improve software reliability across different systems. Gustinov et al. [17] highlighted the benefits of a holistic approach by combining black-box and white-box methods to test an e-commerce platform thoroughly. More specialized research from Corradini et al. [18] focuses on automated black-box testing to find mass assignment vulnerabilities in RESTful APIs. For artificial intelligence, Aghababaeyan et al. [19] proposed a novel method using test case diversity to more effectively uncover flaws in deep neural networks.

Mahardika et al. [20] focused on modernizing business operations by implementing a payment gateway into a mobile restaurant ordering system. Through the integration of a third-party API, they successfully created a secure, automated, and versatile payment process. The primary merit was replacing a manual system, significantly improving operational efficiency and customer convenience. This work shows how such integrations enhance the scalability and professionalism of small businesses.

Furthermore, The development of this research incorporates several technologies, as follows

2.1. React Native

React Native is a framework for creating mobile applications using JavaScript[6], [8]. It provides a set of components for both iOS and Android platforms to build mobile applications with a genuinely native appearance. Using the React Native framework, developers can render user interfaces for iOS and Android platforms[4]. React Native is an open-source framework that is expected to be compatible with other platforms, such as Windows or tvOS.

2.2. Node.js

Node.js is a platform for running JavaScript asynchronously in an event-driven manner, designed to develop network applications and APIs[12], [13]. Node.js is built using the Chrome V8 Engine, an open-source JavaScript engine developed in C++ that implements ECMAScript. Node.js utilizes an asynchronous, event-driven, and non-blocking I/O model, making it lightweight, efficient, and highly performant[16].

2.3. Supabase

Supabase[15] is an open-source alternative to Firebase, built on Postgres. It offers nearly all backend services needed to build a product. Supabase is an open-source database that serves as an alternative to Firebase, providing scalability and functionality that are competitive with Firebase, though not a complete replacement. It offers many similar features but approaches them differently. Supabase leverages existing tools within the ecosystem rather than developing everything from scratch. Additionally, Supabase uses Postgres as its database, intentionally opting for a relational database over NoSQL.

2.4. Marketplace

A marketplace platform acts as an intermediary between sellers and buyers to facilitate the online transaction process for products. It provides various features and facilities, such as payment methods, delivery times, and product selection based on user needs[9], [10]. Furthermore, sellers and buyers conduct transactions online using the application provided by the developer. Once the transaction is completed and both parties agree, the buyer pays, the seller prepares the goods for shipment, and delivery is carried out [11][14].

3. Methods

This section outlines the methodological approach, explains the system architecture, and moves on to the application design stages. The system architecture is explained in detail, defining the roles and interactions among the four leading actors—sellers, buyers, couriers, and administrators—and the technologies used, such as React Native and Supabase. Furthermore, the design stages are illustrated through detailed interface designs for each actor, which depict the application's functionality from a user's perspective.

3.1. System Architecture

The system architecture is designed to help developers understand how the system will be implemented in the application to meet its intended functionality. The application involves four actors: sellers, buyers, couriers, and administrators. This application facilitates the buying and selling of LPG. Sellers can create a store and add products to sell. Logged-in buyers can purchase products by selecting the

nearest store. Meanwhile, administrators can add discounts to the application. To conduct buying and selling activities, several steps need to be performed:

1. Login (as a seller, buyer, or admin)
2. Register (as a seller or buyer)
3. Complete address details (as a seller or buyer)
4. Add products (as a seller)
5. Add couriers (as a seller)
6. Purchase products and make payments (as a buyer)
7. Add ratings (as a buyer)
8. Add discounts (as an admin)

The product ordering process in the ELPIJIKU marketplace application begins with the user selecting the menu to view available products. The system then displays a list of products from which the user can choose. After the user selects a product, the system will show its details, complete with an input field to enter the desired quantity. The user then selects the buy option. Subsequently, the system will display a shopping summary where the buyer can adjust the product quantity, select a payment method, and enter a voucher code (optional). The system then saves the order data and displays an order details page, allowing users to view their order.

The order monitoring process in the ELPIJIKU marketplace application starts when the user selects the "My Orders" menu. After this menu is chosen, the system will display a page listing orders the user places. The user then selects an order to access the details they wish to check further. The system displays the specifics of the selected order, providing detailed information about its status and contents. Finally, the user clicks the "Order Received" option to signify receiving the order. This process ensures that users can easily monitor and confirm the status of their orders within the application.

The shipment monitoring process begins when the user selects the "Shipment" menu. Afterwards, the system displays a list of ongoing product shipments. The user then selects a specific shipment to check its status. The system displays the detailed shipment status for the product chosen, providing up-to-date information on its location and progress. The user can then view the displayed status. If the responsible party has confirmed the delivery, the shipment status will be updated to "Completed." However, if the shipment has not yet been confirmed, the status will remain "In Transit."

The product management process for an admin in the ELPIJIKU marketplace application starts when the admin selects the "Product" menu. The system then displays a list of existing products. The admin can add a new product by creating a form to fill in the new product's details. The admin then completes the form and selects the option to upload the product. The system will save the newly uploaded product information into the database.

After a product is added, the admin is given the option to edit it if necessary. If the admin chooses to edit, the system allows them to update the product details as desired. If not, the admin can opt to delete the product. If choosing to delete, the system will display a confirmation prompt to ensure the action is intended. If the admin decides not to delete, the system will redirect to the product list. This process allows the admin to easily manage existing products by adding, editing, or deleting them as needed.

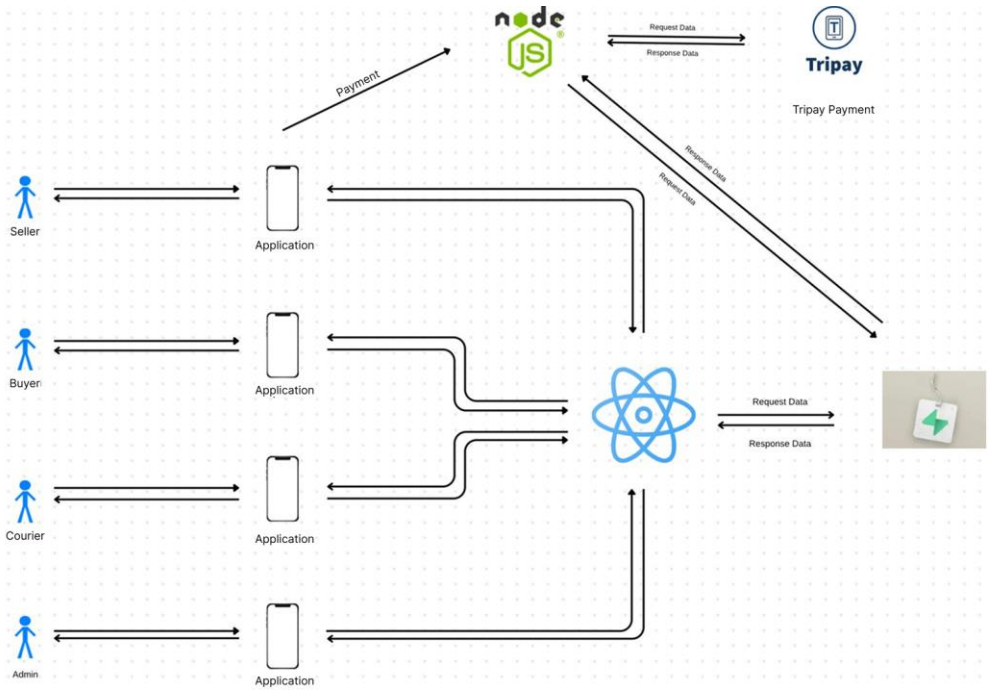


Fig 1. System Architecture Design

In Figure 1, the architecture illustrates the system accounts that will be developed. This research utilizes the React Native framework[5], [7], directly interacting with users. Four actors are involved: sellers, buyers, couriers, and administrators. Sellers log in and post products. The application returns product data, chat, orders, and settings to the seller. Next, the application interacts with React Native, which returns product data, chat, orders, and settings. React Native requests data from Supabase, and Supabase responds with the required data. The workflow for other actors follows a similar flow to that of the seller. The system sends payment data to Node.js to handle payments. Node.js forwards this data to Tripay as the Payment Gateway[20]. Tripay then returns the payment data to Node.js, which updates the payment information in Supabase.

3.2. Database Design

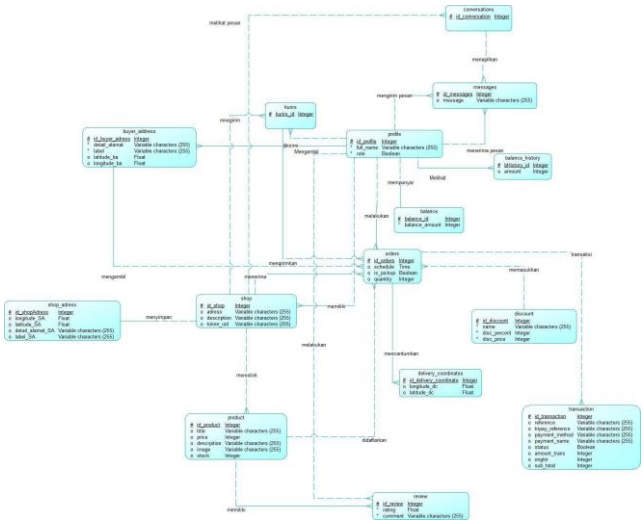


Fig 2. ERD CDM

This diagram illustrates the relationships between these entities, showing how they are interconnected and interact within the system. The admin oversees and manages the entire system, while Sellers and Buyers interact directly in the buying and selling process, supported by Products, Orders, and Deliveries managed within the system.

3.3. Seller Interface Design

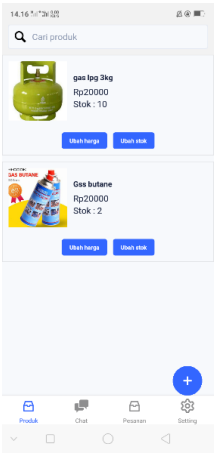


Fig 3. Seller Interface Design

This page displays several products sold by the seller, and the prices can be changed at any time, along with the ability to adjust the stock available. To add a product to the store, the seller can tap the plus sign at the bottom of the page, upload a photo, enter the product name and price, and write a description of the product being sold. Once all the fields are filled, the seller can press the post button, and the new product will appear on the product page.

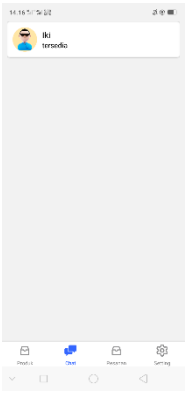


Fig 4. Seller Chat Interface Design

The chat page displays messages from buyers who ask the seller questions before or during the transaction process. The chat page in the ELPIJIKU application is designed to facilitate communication between buyers and sellers during the transaction. This interface allows users to engage in text conversations with the store they transact with. Users will see a display on the chat interface that includes a message area and a conversation history list.

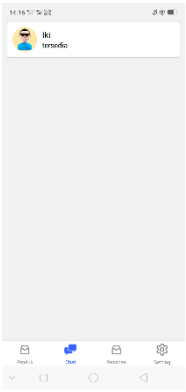


Fig 5. Seller Order Interface Design

This page shows that the seller can view incoming orders on the application. Buyers who have made orders will appear in the seller's orders, and the payment status can be checked to see whether the buyer has completed the payment. When the payment status is "unpaid," the seller will wait for the buyer to pay. Once the buyer has paid, the seller can change the status to "confirmed" and select a courier for delivery.

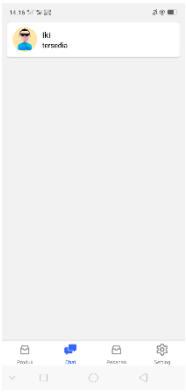


Fig 6. Seller Settings Interface Design

The seller Settings page contains several features that the seller can access. These features help complete the store's identity, such as setting the store name, address, and the couriers used for shipping. The settings page in the ELPIJIKU application is designed to allow users to update their information efficiently.

3.4. Buyer Interface Design

Fig 7. Buyer Home Interface Design

On the buyer's home page, buyers can view several featured products and search for nearby stores with recommendations. The search feature for nearby stores speeds up the delivery process and allows buyers to select stores with the most affordable shipping costs.

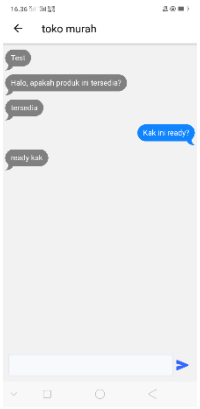


Fig 8. Buyer Chat Interface Design

The chat page displays messages from buyers who ask the seller questions before or during the transaction process. The chat page in the ELPIJIKU application is designed to facilitate communication between buyers and sellers during the transaction. This interface allows users to engage in text conversations with the store they transact with. Users will see a display on the chat interface that includes a message area and a conversation history list.

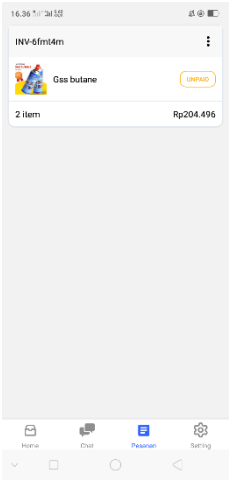


Fig 9. Buyer Order Interface Design

The order page displays the purchase invoice and the payment status, which can be viewed. There are several payment statuses: Unpaid, meaning the payment has not been made; Paid, meaning the payment has been completed; Confirmed, indicating that the seller has approved the payment; Delivered, meaning the seller is in the process of shipping; and Success, meaning the product has been successfully delivered and received by the buyer. When the purchase invoice is opened, it will show details such as the order date, order quantity, item price, shipping fee, discount, status, and the total amount the buyer must pay. A payment code can also be used according to the buyer's agreed-upon payment system. Once all steps are completed, the seller will arrange for the delivery to be assigned to the courier used by the store.

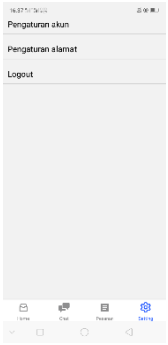


Fig 10. Buyer Settings Interface Design

Buyers can change their accounts on the settings page, such as updating their email and password. The address settings feature also allows buyers to modify their location, address details, and labels for address changes.

3.5. Admin Interface Design

Fig 11. Admin Home Interface Design

Several sections on the admin's home page are available for monthly sales reports, top-selling stores, top-selling products, and most active users.

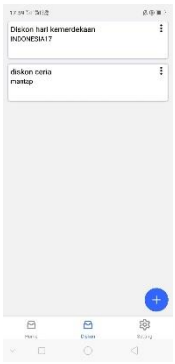


Fig 12. Admin Discount Interface Design

The admin interface page has a section for managing discounts when needed. Discounts can be added by pressing the "+" button at the bottom of the page, which leads to the discount creation section. Here, the admin enters the discount name, unique code, and discount amount and presses the save button. Once the discount is created, buyers can use it by entering the unique code provided.

3.6. Courier Interface Design

Fig 11. Courier Product Interface Design

This page displays a list of orders and the buyer's address details that are ready for delivery. For further information, please refer to Example Formula 1.

4. Results and Discussion

User Acceptance Testing (UAT) verifies that a solution created within a system suits the user. This process differs from system testing (which ensures the software doesn't crash and meets the requirements document). Instead, UAT confirms that the solution within the system will work for the user (i.e., it tests that the user accepts the solution). The client or end-user generally performs UAT and typically does not focus on identifying simple issues like spelling mistakes or showstopper defects like software crashes. Testers and developers identify and fix these issues during the earlier stages of functional, integration, and system testing.

The type of UAT used in this research is Black Box Testing. Black Box testing is one of many standard methods used to test an application. This method only requires the upper and lower bounds of the expected data, as well as the entry rules that must be met, to estimate the amount of test data needed. By using this method, it can be determined whether the functionality can still accept unexpected data inputs, which would result in invalid data. Testing is a series of planned and systematic activities to test or evaluate the desired correctness. Software testing checks whether the software's functions, inputs, and outputs correspond to the required specifications without examining the program's design and code. Black Box testing is used to find weaknesses in the system so that the resulting data is valid with what was input and to minimize shortcomings in the application before the user uses it.

Alpha Testing is conducted to detect critical issues or bugs before the software is released to the public. Internal employees usually perform this testing in a lab or stage environment.

4.1. Alpha Black Box Testing Results

Based on the testing [18] and [19] conducted, the conclusion can be drawn that the application functions as expected [17]. Each feature in the application works well and meets its intended purpose. The conclusion can be seen in the table below:

Table 1. Alpha Testing Results

No	Component Tested	Feature Availability		Conclusion
		Completed	Not Completed	
1	Login Register	V		Successful
2	Search	V		Successful
3	Price Filter	V		Successful
4	Online Purchase	V		Successful
5	Discount	V		Successful
6	Nearby Store Location	V		Successful
7	Rating	V		Successful
8	Chat	V		Successful
9	Product Master	V		Successful
10	Product Stock Master	V		Successful
11	Order Status	V		Successful
12	Payment	V		Successful
13	Add Courier	V		Successful
14	Tracking	V		Successful
15	Self-Pickup	V		Successful
16	Scheduling	V		Successful

4.2. Beta Testing Results

Out of the 52 respondents involved, the majority expressed satisfaction with the application, which can be detailed in the chart below:

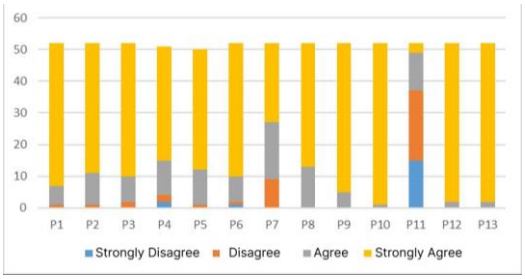


Fig 13. Questionnaire Results Chart

This page displays a list of orders and the buyer's address details that are ready for delivery. For further information, please refer to Example Formula 1.

Table 2. Beta Testing Results

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12

Strongly Disagree	0	0	0	2	0	1	0	0	0	0	15	0
Disagree	1	1	2	2	1	1	9	0	0	0	22	0
Agree	6	10	8	11	11	8	18	13	5	1	12	2
Strongly Agree	45	41	42	36	38	42	25	39	47	51	3	50

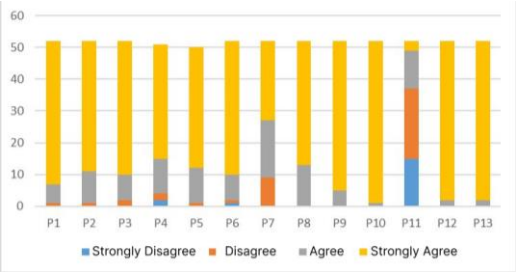


Fig 14. Questionnaire Results Percentage

As shown in the percentage above, the respondents' satisfaction indicates that 75% expressed "strongly agree," while 16% stated "agree." On the other hand, 6% disagreed, and 3% strongly disagreed. Based on this percentage, it can be concluded that out of the 52 respondents, the majority considered the application good quality, with a combined "agree" and "strongly agree" rating of 91%.
However, 9% of the respondents mentioned that a small portion of the application's quality could be improved. As seen in Table 6.23, question 7 received relatively high negative feedback. Question 7 relates to the usefulness of the filter feature within the application.

5. Conclusion

Based on the research results, the Elpijiku marketplace application performed well in the alpha testing without showing any errors. In the beta testing, most users gave positive feedback, with 91% rating the application and user experience positively, indicating high satisfaction among most users.
Although most respondents gave positive feedback about the application's quality, 9% expressed dissatisfaction with some features. Therefore, the developer needs to improve and enhance the application's quality, focusing on refining the filter feature. With this feedback, the user experience is expected to be further enhanced.

Acknowledgement

This work was partially funded by Institut Sains dan Teknologi Terpadu Surabaya (ISTTS) under the Institute for Research and Community Services or Lembaga Penelitian dan Pengabdian kepada Masyarakat (LPPM).

References

[1] F. Hariyanto, T. Budiman, A. B. Yulianto, and V. Yasin, "Designing a Web-Based Information System for Monitoring Final Projects," *International Journal of Engineering, Science and Information Technology*, vol. 5, no. 2, pp. 142–153, Feb. 2025, doi: 10.52088/ijesty.v5i2.799.

[2] J. Salat, R. Rasna, M. Ichsan, and D. Abdullah, "Web-based Rabies Disease Diagnosis Expert System with Forward Chaining and Dempster Shafer Methods," *International Journal of Engineering, Science and Information Technology*, vol. 5, no. 2, pp. 160–167, Apr. 2025, doi: 10.52088/ijesty.v5i2.801.

[3] E. I. Setiawan, P. Hartono, T. N. T. Vu, K. J. Halim, F. X. Ferdinandus, and J. Santoso, "Cross Platform Waste Reuse, Reduce And Recycle Management Application With Prototyping Methodology," *Applied Information System and Management (AISM)*, vol. 7, no. 1, pp. 43–52, Apr. 2024, doi: 10.15408/aism.v7i1.37230.

[4] D. Zou and M. Y. Darus, "A Comparative Analysis of Cross-Platform Mobile Development Frameworks," in *2024 IEEE 6th Symposium on Computers & Informatics (ISCI)*, IEEE, Aug. 2024, pp. 84–90. doi: 10.1109/ISCI62787.2024.10667693.

[5] K. Kishore, S. Khare, V. Uniyal, and S. Verma, "Performance and stability Comparison of React and Flutter: Cross-platform Application Development," in *2022 International Conference on Cyber Resilience (ICCR)*, IEEE, Oct. 2022, pp. 1–4. doi: 10.1109/ICCR56254.2022.9996039.

[6] J. C. De Almeida, F. Brito e Abreu, and D. S. De Almeida, "Cross-Platform Mobile App Development: The IscteSpots experience," in *2023 38th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, IEEE, Sep. 2023, pp. 11–16. doi: 10.1109/ASEW60602.2023.00006.

[7] F. R. Hardjanto, A. Nugroho, F. Hidayat, and M. T. Zulfikar, "Goods Storage Rental Application (YourStorage) Using the React Native Framework," *Engineering, MAthematics and Computer Science Journal (EMACS)*, vol. 6, no. 1, pp. 19–25, Jan. 2024, doi: 10.21512/emacsjournal.v6i1.10759.

[8] E. Coltey, D. Alonso, S. Vassigh, and S.-C. Chen, "Towards an AI-Driven Marketplace for Small Businesses During COVID-19," *SN Comput Sci*, vol. 3, no. 6, p. 441, Aug. 2022, doi: 10.1007/s42979-022-01349-w.

[9] A. Prasitsupparote, P. Pasitsuparoad, and S. Jungjit, "Design and Implementation of a Local Marketplace Application for Food and Beverage Businesses Using 4P's and 4C's Strategies," in *2024 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON)*, IEEE, Jan. 2024, pp. 318–323. doi: 10.1109/ECTIDAMTNCON60518.2024.10480075.

[10] B. J. A. Nucos, J. R. Almonteros, J. Parado, and J. B. A. Leyson, "AgriPlace: A Fair Pricing Regulated E-Marketplace for Agricultural Commodities," in *2025 5th International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*, IEEE, May 2025, pp. 1–5. doi: 10.1109/IRASET64571.2025.11008031.

[11] K. O. M. Salih, T. A. Rashid, D. Radovanovic, and N. Bacanin, "A Comprehensive Survey on the Internet of Things with the Industrial Marketplace," *Sensors*, vol. 22, no. 3, p. 730, Jan. 2022, doi: 10.3390/s22030730.

[12] O. Chaplia and H. Klym, "Designing a Node.js Project Architecture for RESTful Microservices," in *2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, IEEE, Sep. 2023, pp. 808–811. doi: 10.1109/IDAACS58523.2023.10348681.

[13] P. V. K. V. Prasad, N. V. Krishna, and T. P. Jacob, "AI CHATBOT using Web Speech API and Node.Js," in *2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*, IEEE, Apr. 2022, pp. 360–362. doi: 10.1109/ICSCDS53736.2022.9760803.

[14] S. Athreya, S. Kurian, A. Dange, and S. Bhatsangave, "Implementation of Serverless E-Commerce Mobile Application," in *2022 2nd International Conference on Intelligent Technologies (CONIT)*, IEEE, Jun. 2022, pp. 1–5. doi: 10.1109/CONIT55038.2022.9847829.

[15] S. Varshitha. G, Rupa. Ch, and Divya. D, "Remix-based Real Time Blood Bank Communication Integrating Access Control Using XGBoost and Supabase," in *2024 IEEE Students Conference on Engineering and Systems (SCES)*, IEEE, Jun. 2024, pp. 1–6. doi: 10.1109/SCES61914.2024.10652474.

[16] K. Azkiya, M. Irsan, and M. F. Fathoni, "Implementation of App Engine and Cloud Storage as REST API on Smart Farm Application," *Sinkron*, vol. 8, no. 2, pp. 902–910, Mar. 2024, doi: 10.33395/sinkron.v8i2.13386.

[17] M. D. Gustinov *et al.*, "Analysis of Web-Based E-Commerce Testing Using Black Box and White Box Methods," *International Journal of Information System and Innovation Management (IJISIM)*, vol. 1, no. 1, pp. 20–31, Jun. 2023, doi: 10.55583/ijisim.v1i1.687.

[18] D. Corradini, M. Pasqua, and M. Ceccato, "Automated Black-Box Testing of Mass Assignment Vulnerabilities in RESTful APIs," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, IEEE, May 2023, pp. 2553–2564. doi: 10.1109/ICSE48619.2023.00213.

[19] Z. Aghababaeyan, M. Abdellatif, L. Briand, R. S, and M. Bagherzadeh, "Black-Box Testing of Deep Neural Networks through Test Case Diversity," *IEEE Transactions on Software Engineering*, vol. 49, no. 5, pp. 3182–3204, May 2023, doi: 10.1109/TSE.2023.3243522.

[20] Fajar Mahardika, Ratih, and R. Bagus Bambang Sumantri, "Implementation of Payment Gateway in the Mobile-Based Pawon Mbok`E Eating House Ordering System," *Journal of Innovation Information Technology and Application (JINITA)*, vol. 6, no. 1, pp. 64–77, Jun. 2024, doi: 10.35970/jinita.v6i1.2289.