SAGA: Journal of Technology and Information Systems
Vol 3, Issue 2, May 2025, Page 424-432
ISSN: 2985-8933 (Media Online)
DOI: 10.58905/SAGA.v3i2.512

Development of Web Service for Point-of-Sale Application of Kedungudi Sky Park Hill Café

Ari Topan Iqbal Madagaskar, Yulian Firdawati*

Department of Informatics Engineering, University of Muhammadiyah Sidoarjo, Sidoarjo, Indonesia

Corresponding Author: yulianfindawati@umsida.ac.id2*

Abstract. The rapid development of information technology strengthens the need for efficient system integration in sales management at the Point of Sale (POS) in Sky Park Hill Cafe, Kedungudi. This study explores the application of API (Application Programming Interface) technology to enhance interoperability between the POINT OF SALE (POS) application and other systems. APIs enable more efficient and real-time data exchange between various system components, facilitating accurate transaction recording and timely sales reporting. With the implementation of APIs, it is expected that Sky Park Hill Cafe can improve operational efficiency, increase data transparency, and enhance customer satisfaction through a better shopping experience and more responsive services.

Keywords: System Design, Information Technology, Point of Sale, Application Programming Interface

1 Introduction

In today's digital era, businesses must adapt quickly to technological developments to remain competitive[1]. One of the increasingly important technologies is the Point of Sale (POS) application, which helps businesses in managing sales transactions effectively[2] . The development of Point of Sale (POS) application web services offers greater flexibility and efficiency compared to traditional Point of Sale (POS) systems[3] . The utilization of tools such as Postman is essential in this process, as it facilitates testing and debugging of the developed web services .[4]

Postman is a very useful tool in testing and debugging the *Application Programming Interface (API)* used in the development of web service Point of Sale (POS) applications . [5]With Postman, the development team can test various request and response scenarios between the frontend and backend easily [6] . This tool not only speeds up the development process, but also minimizes the risk of errors that may occur during implementation[7] . In addition, Postman's ability to document *Application Programming Interface (API)* makes it easier for teams to better understand and integrate various application features .[8]

Development of web service application Point of Sale (POS) cafe with Extreme Programming (XP) methodology can improve the quality and speed of development[9] . Extreme Programming (XP) emphasizes effective communication, continuous testing, and quick feedback. With short iterations, developers can customize and improve the application according to user needs. This approach not only increases user satisfaction but also ensures the developed application has high robustness and reliability . [10]

A brief literature survey shows that the use of web-based Point of Sale (POS) applications provides many advantages for small businesses such as cafes. Case studies from various literatures show increased operational efficiency and decreased operational costs. In addition, the integration of web services allows cafe owners to monitor their business remotely. Research also indicates that the use of tools such as Postman in *Application Programming Interface (API)* development can speed up the development process and reduce the risk of errors .[11]–[14]

Therefore, this research aims to develop Web Service Point of Sale Application at Café Kedungudi Sky Park Hill. Café Kedungudi Sky Park Hill is one of the culinary destinations located in Mojokerto city, which always strives to provide the best experience to customers. In an effort to improve operational efficiency and service quality, the development of a web-based Point of Sale (POS) application is the right solution. This application not only helps in processing sales transactions quickly and accurately, but also enables more effective inventory management and in-depth data analysis. The use of tools such as Postman in web service development is crucial to ensure the system runs smoothly.

SAGA: Journal of Technology and Information Systems
Vol 3, Issue 2, May 2025, Page 424-432
ISSN: 2985-8933 (Media Online)
DOI: 10.58905/SAGA.v3i2.512

By applying Extreme Programming (XP) methodology in the development of web service POS application for Café Kedungudi Sky Park Hill, it is expected to maximize the added value provided to customers. XP offers an adaptive and collaborative approach, which focuses on continuous testing, rapid feedback, and short iterations. Thus, developers can respond to changing market needs more flexibly and produce application solutions that are more in line with user expectations.

2 Methods

The research method used in this study is Extreme Programming (XP), aimed at small and medium-sized teams that face very rapid requirement changes in software engineering. Extreme Programming (XP) applies practices such as Pair Programming and Test-Driven Development to improve software quality[15] . Pair Programming improves team collaboration, while Test-Driven Development ensures code changes are continuously tested and meet functional requirements. Extreme Programming (XP) also focuses on rapid feedback and small iterations, allowing researchers to quickly test and refine their hypotheses[16] . Active customer engagement helps understand user preferences, crucial in developing relevant and useful solutions. Thus, Extreme Programming (XP) is effective in incorporating practices that improve quality, speed, and responsiveness in software development.

Kent Beck, the originator of Extreme Programming (XP), was faced with a Chrysler Compensation project that almost failed before being intervened by Ron Jeffries to implement XP. Beck successfully introduced changes that increased the flexibility, efficiency, and adaptability of the project according to the set goals. XP focuses on the values of feedback, simplicity, communication, and courage, and is designed to accommodate developments requested by clients. Here is the flow of the Extreme Programming (XP) method in Figure 2.1.

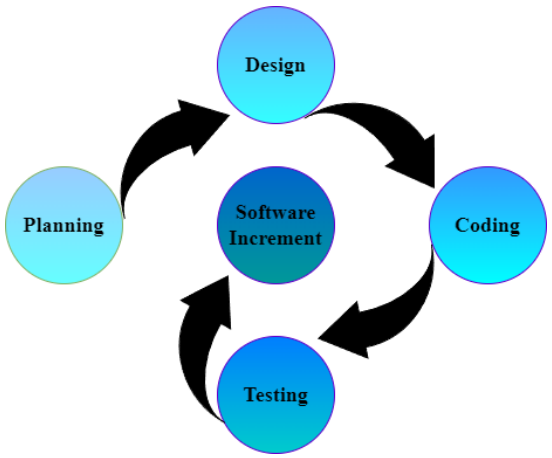

Figure 1. Extreme Programming (XP)

a. The Planning stage looks at system requirements carefully to understand the business flow and detail the main features desired, as the first step in designing the application.
b. The Design stage includes system modeling based on analysis from the planning stage and database management to describe the relationship between data implemented in MySQL.
c. The Coding stage implements the system design into program code using PHP.
d. The Testing stage uses Acceptance Testing to analyze and test the application, focusing on system features and functions thoroughly with end-to-end scenarios, including load testing to ensure system performance in anticipated usage situations.

SAGA: Journal of Technology and Information Systems
Vol 3, Issue 2, May 2025, Page 424-432
ISSN: 2985-8933 (Media Online)
DOI: 10.58905/SAGA.v3i2.512

3 Results and Discussion

3.1 System Design

A. Database

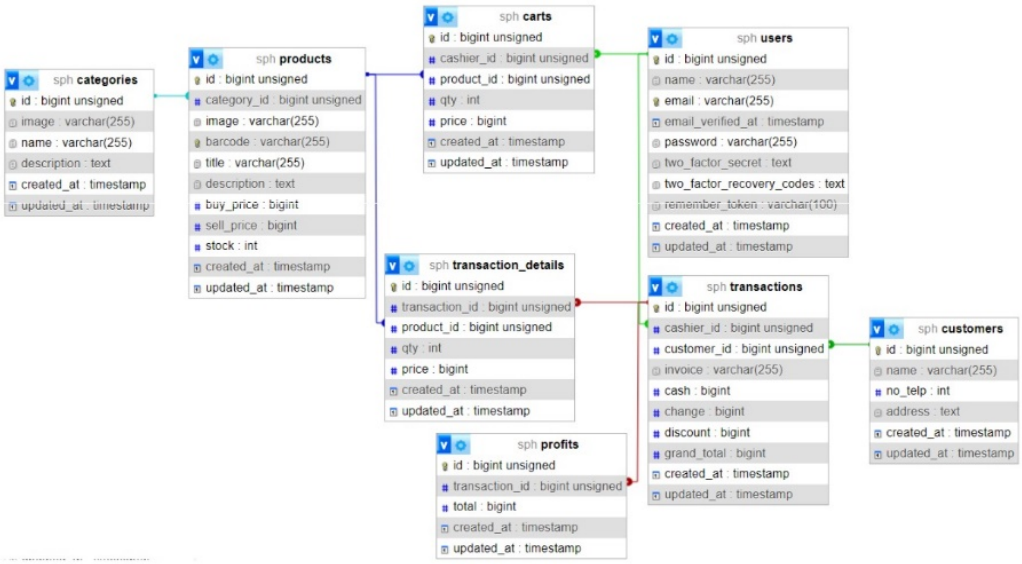

Figure 2. Database

Based on Figure 2 is a database diagram designed to support various workflows in the Point Of Sale Application. Admins can manage user and customer information, products and categories, shopping carts, and transactions including payment and shipping. In addition, the system provides features to generate reports and analysis based on transaction and customer data, which can help in identifying patterns and trends to support business decision making. The database can also be extended with additional tables to support more complex functionality, with special attention to data accuracy, consistency, and security.

B. Class Diagram

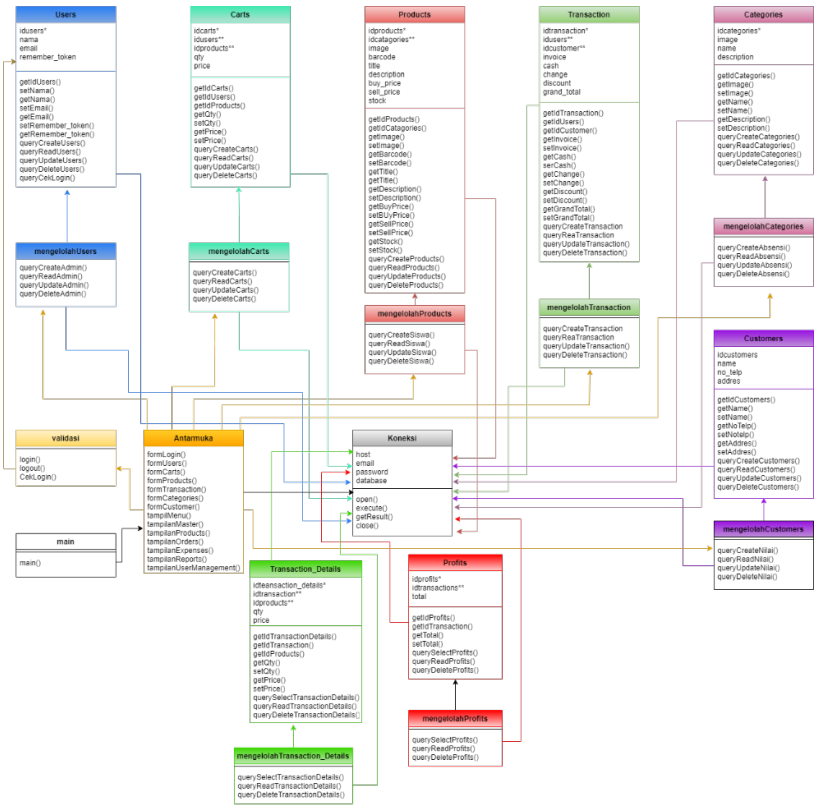

Figure 3. Class Diagram

SAGA: Journal of Technology and Information Systems
Vol 3, Issue 2, May 2025, Page 424-432
ISSN: 2985-8933 (Media Online)
DOI: 10.58905/SAGA.v3i2.512

Based on Figure 3 is a class diagram that contains classes such as Users, Carts, Products, Transactions, Categories, Customers, transaction_Details, Profits and interfaces such as formLogin(), formUsers(), formCarts(), and others. Each class has different attributes and methods, and there are relationships between classes such as the association between the Transaction class and the Users and Customers classes.

C. Use Case Diagram

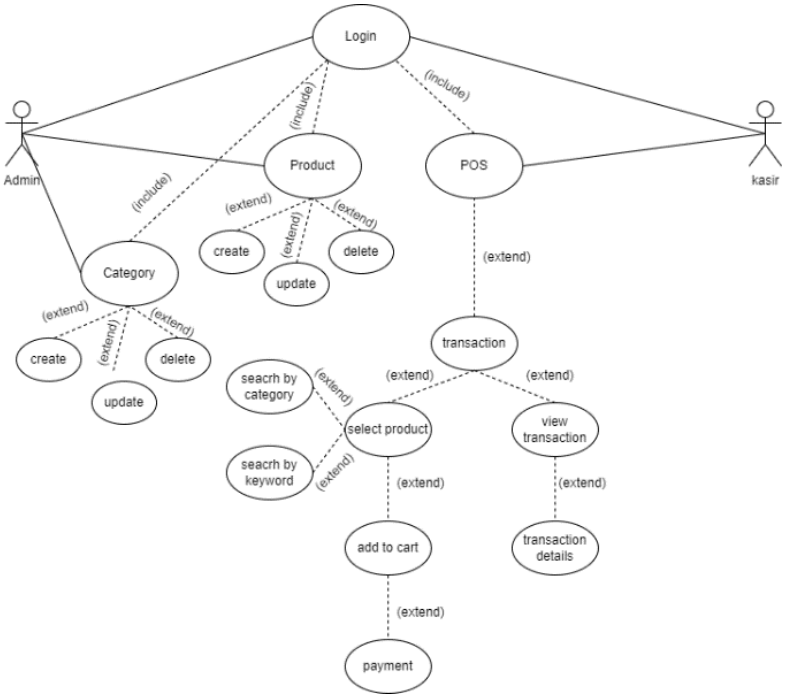

Figure 4. Use Case Diagram

Based on Figure 4 is a use case diagram on the Point of Sale Application describing the product management system and transactions, which may be used for sales or e-commerce systems. Admins have full access to manage products and categories, while cashiers have limited access to make sales transactions. Use cases include the login process for both actors, product and category management, and sales transactions involving product selection, adding to cart, and payment. This diagram provides an overview of the system functionality without in-depth implementation details.

3.2 Interface Design

A. Login Page

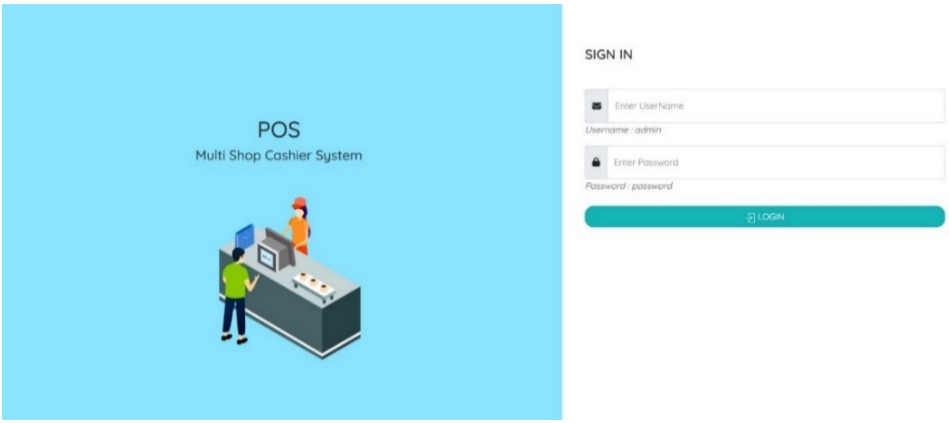

Figure 5. Login Page

In Figure 5 on this page, is used for Admin login so that they can access the website.

SAGA: Journal of Technology and Information Systems
Vol 3, Issue 2, May 2025, Page 424-432
ISSN: 2985-8933 (Media Online)
DOI: 10.58905/SAGA.v3i2.512

B. Dasboard Page

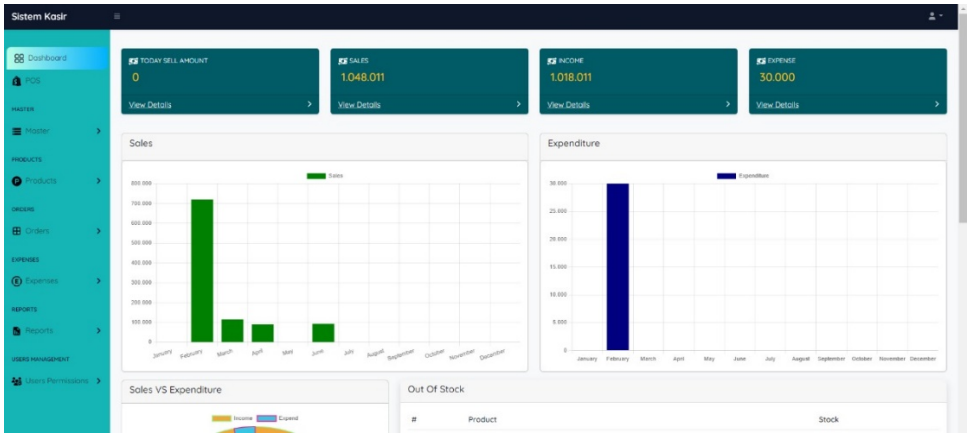

Figure 6. Dasboard Page

Figure 6 is a view of the website dashboard that displays the sales graph.

C. Products Page

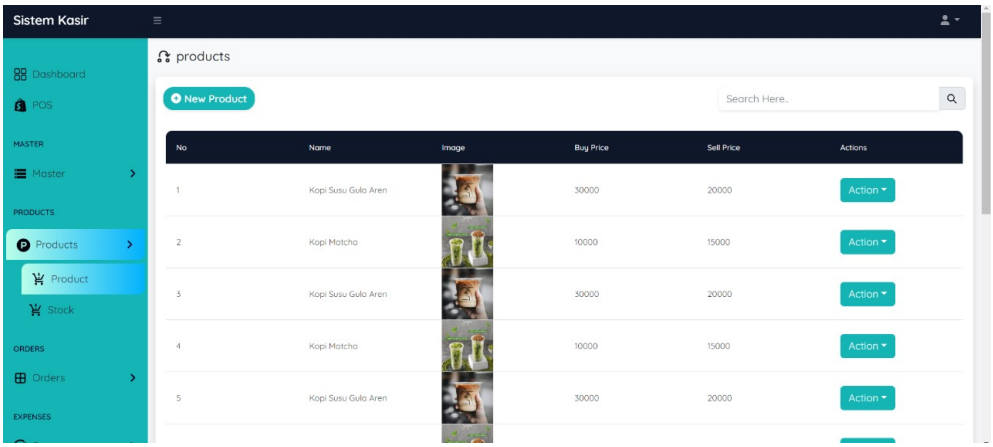

Figure 7. Product Page

In Figure 7 is a Product Page which will display the Products that will be sold at the cafe.

D. Carts Page

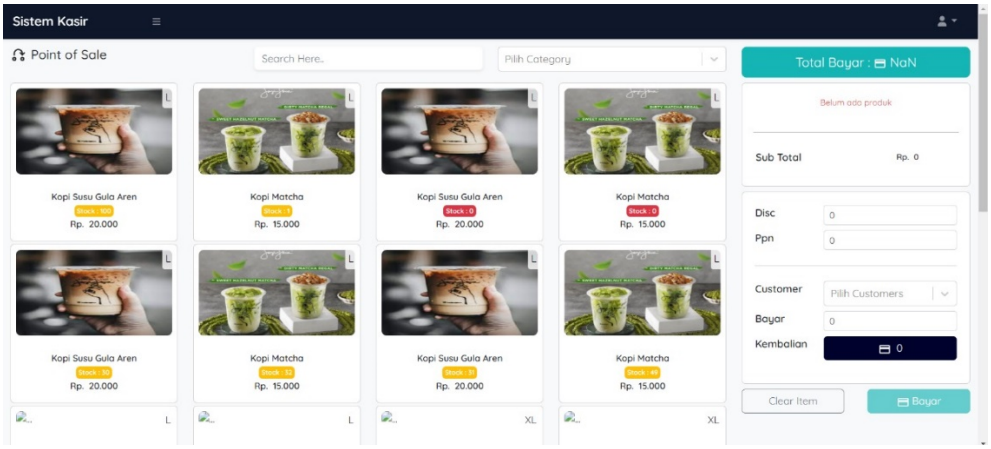

Figure 8. Carts Page

Figure 8 is a display of the Carts page which is used to import orders from customers.

SAGA: Journal of Technology and Information Systems
Vol 3, Issue 2, May 2025, Page 424-432
ISSN: 2985-8933 (Media Online)
DOI: 10.58905/SAGA.v3i2.512

E. Transaction Orders Page

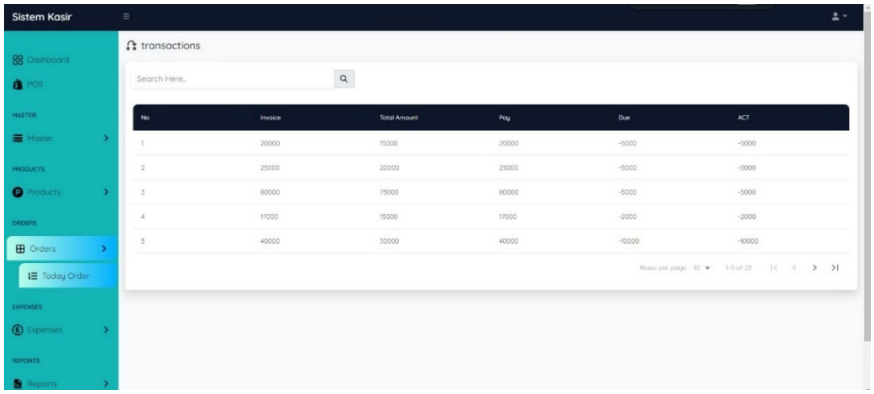

Figure 9. Order Transaction Page

Figure 9 is a display of the transaction page of the order in one day or it can also be called the order history in one day.

F. Users Page

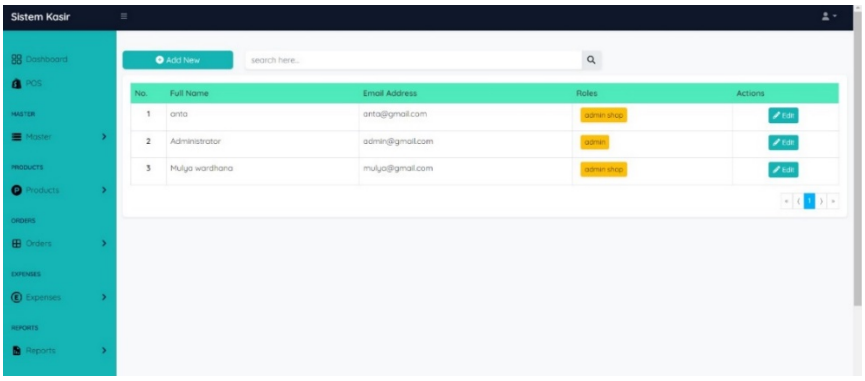

Figure 10. User Page

Figure 10 is a page that will display the User where the user is the user to become a cashier.

G. Note Display

Figure 11. Note Display

Figure 11 is a view of the customer order note that displays the order data ordered by the customer.

SAGA: Journal of Technology and Information Systems
Vol 3, Issue 2, May 2025, Page 424-432
ISSN: 2985-8933 (Media Online)
DOI: 10.58905/SAGA.v3i2.512

3.3 Results of Acceptance Testing

Researchers use *Acceptance* testing which usually involves an "*end-to-end*" scenario to test various relevant features and functions of the system that has been developed. In addition, researchers can perform Load testing using locust.io to ensure that the system or application can handle the expected number of users and transactions without experiencing significant failure or performance degradation.

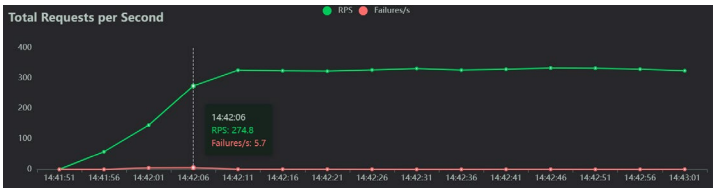

Figure 12. RPS Graph

Based on Figure 3.8 is a graph that shows the RPS increased to 300 at 14:42:06 and stabilized. The system handles 300 requests per second without significant performance degradation, with an average of 5-6 failures per second. The percentage of failures is small, indicating good availability. The system is capable of handling 300 RPS with acceptable failures, however further analysis is required to ensure performance meets requirements. The following RPS Graph results can be seen in table 3.1 *Request Statistics.*

Table 1. Request Statistics

Request Name	Type	#	Failed	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average Size (bytes)	Current RPS	Failed/s
/api/admin/carts	GET	5753	11	30	227.83	2	6347	933.21	83.3	0.1
/api/admin/login	GET	5607	25	29	217.11	2	6369	930.83	80.3	0
/api/admin/products	GET	5696	20	29	206.59	2	6332	931.72	78.3	0.1
/api/admin/transaction-orders	GET	5783	20	29	231.46	2	7107	931.77	82.3	0.1
Aggregated	-	22839	76	29	220.82	2	7107	931.89	324.1	0.3

Table 3.1 *Request Statistics* displays the results of a load test run on the server using Locust.io. This test was successful All requests were GET, with 4 endpoints tested: */api/admin/carts, /api/admin/login, /api/admin/products,* and */api/admin/transaction-orders*.

Number of requests:

a. */api/admin/carts* has the most with 5,753 requests.
b. */api/admin/transaction-orders is* the smallest with 5,783 requests.

Request performance:

a. Fastest median response */api/admin/login* 25 ms, longest */api/admin/transaction-orders* 30 ms.
b. Average fastest response */api/admin/login* 29 ms, longest */api/admin/transaction-orders* 30 ms.
c. Min fastest response */api/admin/login* and */api/admin/products* 2 ms, longest */api/admin/transaction-orders* 2 ms.
d. Max longest response */api/admin/carts* 6,347 ms, fastest */api/admin/login* and */api/admin/products* 2 ms.
e. The largest average request size */api/admin/transaction-orders* 7,107 bytes, the smallest */api/admin/login* 6,332 bytes.

RPS (Requests Per Second):

a. 324.1 RPS on average.
b. */api/admin/carts* highest 83.3 RPS.
c. */api/admin/transaction-orders* lowest 82.2 RPS.

Request failure:

a. Overall failure rate 0.3%.
b. */api/admin/login* has no failures.
c. */api/admin/transaction-orders* highest failure rate 0.1%.

Conclusion:

Based on the available data, the Request Static table shows that the system as a whole can handle loads up to about 324.1 RPS with a relatively low failure rate. However, there are some endpoints that have slower performance and higher failure rates compared to other endpoints.

SAGA: Journal of Technology and Information Systems
Vol 3, Issue 2, May 2025, Page 424-432
ISSN: 2985-8933 (Media Online)
DOI: 10.58905/SAGA.v3i2.512

Table 2. Failures Statistics

Failures	Method	Name	Message
3	GET	/api/admin/carts	ConnectionResetError(10054, 'An existing connection was forcibly closed by the remote host', None, 10054, None)
5	GET	/api/admin/transaction-orders	ConnectionResetError(10054, 'An existing connection was forcibly closed by the remote host', None, 10054, None)
6	GET	/api/admin/login	ConnectionResetError(10054, 'An existing connection was forcibly closed by the remote host', None, 10054, None)
6	GET	/api/admin/products	ConnectionResetError(10054, 'An existing connection was forcibly closed by the remote host', None, 10054, None)
8	GET	/api/admin/carts	ConnectionAbortedError(10053, 'An established connection was aborted by the software in your host machine', None, 10053, None)
14	GET	/api/admin/products	ConnectionAbortedError(10053, 'An established connection was aborted by the software in your host machine', None, 10053, None)
15	GET	/api/admin/transaction-orders	ConnectionAbortedError(10053, 'An established connection was aborted by the software in your host machine', None, 10053, None)
19	GET	/api/admin/login	ConnectionAbortedError(10053, 'An established connection was aborted by the software in your host machine', None, 10053, None)

Based on Table 3.2, the Failures Statistics Table summarizes the error events that occurred during load testing. This table provides information about the number of failures, HTTP method, endpoint, and specific error messages for each failed request. Based on Table 3.2, it can be concluded as follows:

a. Connection Problems: All failures are related to connection issues, specifically:
   1. *ConnectionResetError(10054)*: It means that the remote host forcibly closed the connection.
   2. *ConnectionAbortedError(10053)*: This means that the connection was terminated by software on the local host.
b. Impact on Endpoints: All endpoints experienced failures, with ***/api/admin/login*** and ***/api/admin/products*** having the highest number of failures.
c. Frequency: Failure frequency varies for each endpoint, with ***/api/admin/login*** having the highest frequency for *ConnectionAbortedError*.

4 Conclusion

The Point of Sale system design includes a database design that supports the management of user, customer, product, category, shopping cart, and transaction information. This database generates reports and analyzes transaction data to support business decisions with accuracy, consistency, and data security. The class diagram includes classes such as Users, Carts, Products, Transactions, Categories, Customers, and transaction_Details as well as interfaces such as formLogin() and formUsers(). The use case diagram illustrates the product and transaction management system by admin and cashier, where admin has full access and cashier has limited access. Interface design includes login page, dashboard, products, carts, transaction orders, users, and receipt display.

Acceptance testing shows that the system is capable of handling 300 requests per second with acceptable failures, the average failures per second is 5-6, and the small failure percentage indicates good availability. Performance testing showed variations in response time and request size with an overall failure rate of 0.3%. Overall, this Point of Sale system performed well in handling high loads with low failure rates.

5 Acknowledgments

Hopefully this paper will provide benefits and deep understanding for the readers. Thank you to all those who have participated in the process of writing and preparing this paper. The support and contributions provided by various parties during the research and writing stages are very meaningful. Without their help, this paper would never have been realized.

References

[1] R. S. Halim, T. Y. Chandra, and H. A. Mau, "Volume 2 Number 9 September 2023 MEDICAL FAILURE IN INDONESIA," *J. Multidisciplinary Indones.*, vol. 2, no. September, pp. 3048-3074, 2023, [Online]. Available: https://jmi.rivierapublishing.id/index.php/rp/article/view/580/666

[2] R. Laksono and J. R. Gultom, "The Use of Digital Marketing and Point of Sales (Pos) System as a Business Development Strategy at Umkm Warung Tegal Kharisma Bahari in Jakarta," *Mediastima*, vol. 28, no. 1, pp. 1-10, 2022, doi: 10.55122/mediastima.v28i1.383.

[3] O. A. I. Susilowaty and M. Rukismono, "Factors Considered in Choosing Moka POS as a Point Of Sale System in Surabaya," *Semin. National. Applied Science. V 2021*, pp. 1-7, 2021.

[4] L. G. N. Iksyam, *Implementation of mecha application business: android-based vehicle repair service using mvp as design pattern*. 2023. [Online]. Available: dspace.uii.ac.id/123456789/48037

[5] V. Y. P. Ardhana, M. T. Hidayat, M. Jannah, S. Sumiati, P. Rini, and N. Sari, "Implementation of RESTful API on Laravel and Wokwi IoT Simulator for Temperature and Humidity Measurement Using Waterfall Method," *Arcitech J. Comput. Sci. Artif. Intell.*, vol. 3, no. 2, p. 93, 2023, doi: 10.29240/arcitech.v3i2.9334.

[6] I. Setiawan, R. A. Nughroho, and H. Supratikta, "Application of microservices architecture on dwidaya tour information system migration with rapid application development (rad) approach," pp. 1167-1174, 2024, doi: 10.62567/micjo.v1i3.138.

[7] T. Bratakusuma, I. U. Azmi, and S. Ayuningtiyas, "Back End Development on Stationery Application of Bank Indonesia Purwokerto Representative Office Using Nodejs," *Semin. Nas. Inov. and Development. Technol. Applied. Cilacap*, pp. 119-127, 2022, [Online]. Available: https://ejournal.pnc.ac.id/index.php/senovtek

[8] M. I. Syahbana, *Design of Daily Production Monitoring Application for Foreman Tebenan Unit Pt. Perkebunan Nusantara Vii Based on Android*. 2023.

[9] H. Setiawan, S. Dwi, and Y. Kusuma, "OKTAL: Journal of Computer Science and Science APPLICATION OF EXTREME PROGRAMMING METHOD ON DESIGNING ANDROID-BASED POINT OF SALES APPLICATION (CASE STUDY KEDAI KOPI MOKARA)," vol. 2, no. 12, pp. 3268-3282, 2023.

[10] D. Oscar, Hendri, M. chwanul Muslim, and M. Fahmi, "Journal of Vocational Electronic Engineering and Informatics Design of Point Of Sales (POS) Information System," *J. Vocat. Tech. Electron. and Inform.*, vol. 11, no. 3, 2023, [Online]. Available: http://ejournal.unp.ac.id/index.php/voteknika/

[11] R. Nandang Pratama and Y. A. Susetyo, "Implementation of Python API with Flask Framework as Cloud Run Service for Update Process at PT XYZ," vol. 5, no. 2, pp. 669-676, 2024.

[12] A. Simangunsong and A. A. Nababan, "Pkm Utilization of Point Of Sale (Pos) Application at Digital-Based Pos Coffee," *J. Pengabdi. Kpd. Masy. Nusant.*, vol. 4, no. 3, pp. 2113-2117, 2023.

[13] R. Afriansyah, M. Sholeh, and D. Andayati, "Designing Web-based Interface Programming Applications Using Representation Architecture Style for School Attendance Systems," *J. Scr.*, vol. 9, no. 1, pp. 84-93, 2021.

[14] I. G. Wiryawan, M. D. Agustiningsih, M. Yusuf, V. A. Pratama, and L. D. Wahyuningsih, "Z-Waste: A Mobile-Based Environmentally Friendly Application," *J. Resist. (Computer Systems Engineering)*, vol. 4, no. 1, pp. 26-32, 2021, doi: 10.31598/jurnalresistor.v4i1.664.

[15] R. Nuari, "New Student Registration Information System Using Extreme Programming," *J. Data Sci. Inf. Syst.*, vol. 1, no. 4, pp. 166-174, 2023, [Online]. Available: https://doi.org/10.58602/dimis.v1i4.79

[16] F. Aisyah Mutia Dawis, Yusuf Wahyu Setiya Putra, N. R. F. Dini Hamidin, Syifa Nurgaida Yutia, Maniah, and F. N. Dea Wemona Rahma, *SOFTWARE ENGINEERING PRACTICAL GUIDE FOR THE DEVELOPMENT OF QUALITY APPLICATIONS*. 2023.