Journal of Applied Informatics and Computing (JAIC)
Vol.8, No.2, December 2024, pp. 623~630
e-ISSN: 2548-6861

Optimization of Direct Sales and Sales Canvasser Sales Target Monitoring With RESTful API Implementation on Web-Based Monitoring System

Restian Dwi Friwaldi [1]*, Suyud Widiono[2]**
* Informatika, Universitas Teknologi Yogyakarta
** Sains Data, Universitas Teknologi Yogyakarta
restian.5210411216@student.uty.ac.id [1], suyud.w@uty.ac.id [2]

Article Info

Article history:

Received 2024-10-04
Revised 2024-11-04
Accepted 2024-11-05

Keyword:

Restful API,
Sales Monitoring,
Direct Sales,
Sales Canvasser,
Website.

ABSTRACT

RESTful API offers advantages in real-time data exchange, scalability, and ease of integration with other systems. This research aims to develop a web-based monitoring system using RESTful API to optimise the monitoring of direct sales targets and sales canvassers at. This system is built with Laravel framework and Agile method, focusing on ease of use and real-time data access. Tests were conducted using Apache JMeter with load scenarios of 500, 750, and 1000 users. The test results showed response times of 758 ms for 500 users, 762 ms for 750 users, and 880 ms for 1000 users, all below the target of 900 ms. The error rate was recorded at 0.00%, indicating the high reliability of the system. The throughput achieved was 90.30 requests per second for 500 users, 124.30 requests per second for 750 users, and 159.10 requests per second for 1000 users, exceeding the target of 150 requests per second. Recommendations for further development are the integration of mobile applications for accessibility and real-time monitoring of sales performance.

I. INTRODUCTION

In an era of increasingly fierce business competition, innovation and technology utilisation are key elements to improve the efficiency and effectiveness of company operations. One aspect that is very important to pay attention to is sales management, because of its significant role in expanding market share and maintaining customer loyalty [1]. In the highly dynamic telecommunications industry, direct sales and sales canvasser strategies are two crucial methods to bridge the direct relationship between companies and consumers [2]. A key challenge in sales strategy is to effectively monitor and manage sales targets. Lack of automation and data integration hampers access to critical information, causing delays in strategic decision-making. As a result, sales performance drops and the company's competitiveness weakens [3]. The implementation of RESTful API is an effective solution because it automates the process of monitoring sales performance in real-time and provides faster and more accurate access to data [4].

Previous research has examined the application of RESTful APIs in various contexts, especially in the development of real-time monitoring and data management systems. Research conducted by [5] showed that the application of RESTful API on a web-based project monitoring system can significantly improve the efficiency of project tracking. In addition, research by [6] developed a RESTful API-based hydroponic plant monitoring system that is able to optimise the delivery of sensor data in real-time.

Other research by [7] revealed that the implementation of RESTful APIs in Android-based monitoring applications can improve the efficiency of data tracking and facilitate access to real-time information. This emphasises the importance of RESTful API in creating a responsive and flexible system, allowing users to monitor without technological barriers. Research by [8] who implemented REST API in stock management, showed that the use of REST API in a web-based application facilitates the management of stock data, using Extreme Programming methods that significantly improve the efficiency of stock management.

Although various studies have shown the successful use of RESTful API in various sectors, studies related to its application in the telecommunications industry are still very limited, especially for monitoring the sales performance of direct sales and sales canvassers. Therefore, this research aims to fill the gap by developing a RESTful API-based monitoring system specifically designed to monitor sales performance in the telecommunications industry. This system is able to automate the monitoring process, accelerate decision-making, and improve company performance in the face of increasing competition.

RESTful APIs offer many advantages, such as ease of implementation, flexibility in learning and use, and support for common data exchange formats such as JSON and XML [9]. In addition, RESTful APIs allow communication and data exchange between systems to be more efficient, ultimately reducing information delays, one of the major bottlenecks in the sales process [10]. With fast and efficient performance, RESTful APIs are ideal for supporting monitoring systems that require real-time data [11].

This research can make an important contribution in the application of RESTful API to improve the effectiveness of monitoring direct sales and sales canvassers in the telecommunications industry.

II. RESEARCH METHODS

A. Research Framework

The research scheme to be implemented in this study is shown in Fig. 1.

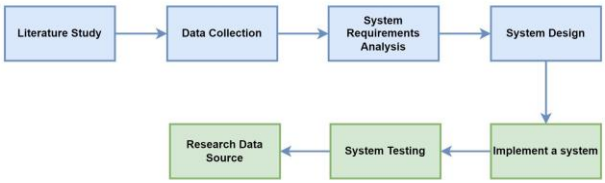

Fig 1. Research Framework

The following is a detailed explanation of each stage illustrated in the research framework in Figure 1.

1) Literature Study.

The initial stage of this research is a literature study, which is carried out by studying and analysing various relevant sources such as journals, scientific articles, or electronic books related to the topic of sales target monitoring and RESTful API implementation.

2) Data Collection.

The second stage is data collection, where researchers collect data from direct sales and sales canvassers through interviews and from related secondary data sources. Through interviews and from related secondary data sources.

3) System Requirements Analysis.

Requirements analysis aims to identify the functional and non-functional requirements needed for a web-based monitoring system.

4) System Design.

Analysis of the workings and user interface of the application to be built, which is included in the system planning stage. System design involves database design using Entity Relationship Diagram (ERD), web-based user interface design as front-end, and system modelling using UML (Unified Modeling Language), such as Use Case Diagram and Inter-Table Relationship Diagram.

5) System Implementation.

The researcher implemented it into PHP program code by utilising the Laravel framework to build the RESTful API and the user interface (front-end). Laravel was chosen as the main framework in the development of the web-based monitoring system due to a number of significant advantages. The Laravel framework provides various libraries and built-in features, such as authentication, session management, and routing systems, which substantially increase efficiency in the application development process. The security aspect is also an excellent attribute of Laravel, with strong protection against CSRF (Cross-Site Request Forgery) and XSS (Cross-Site Scripting) attacks, which is especially important for applications that handle sensitive data, such as sales information [12].

Laravel also has good compatibility with RESTful API implementations, enabling real-time data integration in monitoring systems. In terms of performance and scalability, Laravel demonstrates optimised caching capabilities and is able to handle simultaneous user access efficiently, making it a great choice for systems that require high performance [13]. Extensive community support and comprehensive documentation also make development and troubleshooting easy, making Laravel an ideal solution for real-time sales monitoring applications in the dynamic telecoms industry. [14].

In Table 2, a comprehensive comparison between Laravel, CodeIgniter, Express.js, and Django is presented based on important criteria relevant in the context of monitoring system development.

GraphQL offers great flexibility in data retrieval, allowing applications to retrieve only the required portion of data [15]. However, in the context of monitoring systems that require simple and static data structures such as sales monitoring systems, RESTful APIs are more suitable. Sales monitoring systems generally do not require overly complex and personalised queries, so the use of GraphQL can be redundant. In addition, GraphQL requires more complicated security settings, which can add overhead compared to RESTful APIs [16].

TABLE I
FRAMEWORK COMPARISON BASED ON DEVELOPMENT CRITERIA

Criteria	Laravel	CodeIgniter	Express.js	Django
Programming Language	**PHP**	**PHP**	**JavaScript (Node.js)**	**Pyhton**
Security	CSRF, XSS, password encryption protection	Less complete than Laravel	Needs additional configuration (e.g. helmet.js)	There are built-in protections for CSRF, XSS, and built-in authentication
Ease of Development	Supports Eloquent ORM, many built-in features such as authentication	Minimalist, but not many built-in features	Fast and lightweight, but limited inbuilt features	Built-in ORM support, automatic routing, many ready-to-use features
RESTful API Support	Full support and easy integration with APIs using JSON	Implementable but requires manual configuration	Flexible in implementing REST API	REST API support through Django REST Framework (DRF), easy and fast
Scalability	Excellent, supports caching and high traffic handling	Good for small to medium-sized applications	Good, suitable for microservices-based applications	Good with caching support, load balancing, suitable for large scale
Community and Documentation	Large community, very extensive and active, many tutorials and very complete documentation.	Quite large, but documentation is incomplete	Large and active, many additional modules available on npm	Large community and complete documentation, supporting the Python ecosystem
Performance under High Load	Stable in managing many simultaneous requests (suitable for sales monitoring)	Adequate, but not as optimal as Laravel for very high loads	Fast and scalable, suitable for real-time applications	Stable, suitable for enterprise applications with high access load

Source: [12], [32], [33], [34]

On the other hand, WebSocket is more suitable for applications that require continuous real-time communication, such as chat or gaming applications [17]. For sales monitoring, full real-time data updates are not always required, so using WebSocket can be inefficient. RESTful APIs are able to handle data update needs that do not require persistent connections, and are more resource-efficient, making them a more lightweight and efficient choice [18].

The development of this system started on 22 April 2024 and was completed on 21 May 2024, with a total development duration of four weeks. The time invested in development totalled 240 hours.

6) System Testing

This test examines the performance of the RESful API that has been developed through testing using the load testing method with the Apache JMeter application. Testing begins with determining the API Endpoint to be tested. The endpoints to be tested are shown in Table 2.

TABLE II
ENDPOINTS USED FOR LOAD TESTING

No	Endpoint	Method	Description
1	/login	Post	Used to perform user login
2	/sales	Get	Used to get sales data
2	/produk	Get	Used to get product data
3	/penjualan	Post	Used to perform sales results
4	/target	Get	Used to get sales target data

After determining the endpoint, then observe the error rate, response time, and throughput generated. According to [19] The error rate in Apache JMeter is defined as the proportion of requests that fail to be sent or received during the performance testing process, indicating a problem in the system being evaluated. According to [20], throughput in Apache JMeter represents the rate of request processing per unit of time, which directly correlates to the system's capacity to handle fluctuating workloads. Response time or latency is the duration required to process a request and send a response. This parameter reflects the minimum time required by a system using a particular load balancing algorithm to provide a response. The lower the response time value, the better the performance of the system [21].

To obtain the error rate and throughput in this study, load testing was conducted 3 times with different number of threads (virtual users), namely 500, 750, and 1000. The test uses ramp-up time 1 and loop count 1 on Apache JMeter.

7) Drawing Conclusions and Suggestions

The researcher draws conclusions based on the results of system testing which shows the feasibility of the system to be used in monitoring the sales targets of direct sales and sales canvassers, as well as identifying the advantages and disadvantages of the system. These findings can be a reference and suggestion for further development of a website-based monitoring system with RESTful API implementation.

A. Research Data Source

This research utilises two data sources, namely primary data sources and secondary data sources.

- Research by [22] Primary data is data collected directly from the source by researchers. Primary data is

obtained through interviews with employees who serve as Rural Sales Executives.

- Research by [23] Secondary data is supporting data that is not directly obtained from the primary source, generally in the form of documents. Secondary data includes sales records and internal company documents, including product-related information such as starter packs.

B. Needs Analysis

1) Functional Requirements

In the research conducted by [24] functional requirements refer to the functions that the system must perform and can perform. These requirements are defined in detail as a guide that describes the goals the system wants to achieve and the actors authorized to perform each function.

- The system is flexible in setting sales targets based on input from the admin.
- The system authorizes the admin to manage sales-related data.
- Product information can be set and updated through the system by the admin.
- The system facilitates sales to record sales independently.

These features will continue to be improved based on feedback from users through interview sessions. Users can provide input on necessary functions and additional features that can improve efficiency in the sales process.

2) Non-Functional Needs

According to [25] non-functional requirements are Non-functional requirements refer to constraints related to the services or functions provided by the system, such as time constraints, limitations in the development process, and standards that must be adhered to.

- The system has a simple and easy-to-use interface.
- The system can be accessed through various types of web browsers, such as Google Chrome and Mozilla Firefox.

III. RESULTS AND DISCUSSION

A. System Design

1) Proposed System Architecture

The proposed system architecture can be seen in Fig. 2. Fig. 2 explains that the web-based front-end application acts as the user interface, while the RESTful API serves as an intermediary for efficient data exchange in JSON format.

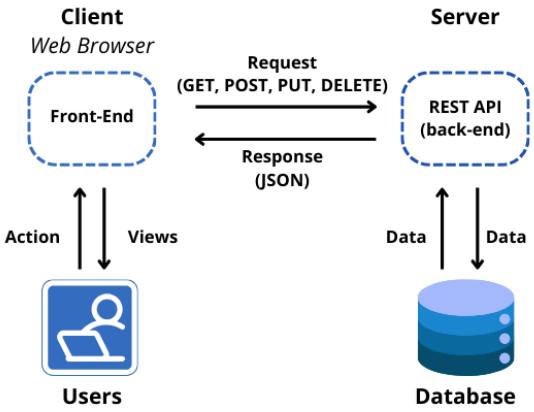

Fig. 2. Proposed System Architecture

Data is stored in the database, including information related to sales targets, sales achievements, direct sales data, sales canvassers, products, and more. With this architecture, sales data input from direct sales and sales canvassers is channeled through the front-end application to the RESTful API for processing and storage. Rural Sales Executives can access current data through the front-end application that retrieves data from the database via the RESTful API. This enables efficient data exchange, scalability, separation between user interface and business logic, and ease of integration with other systems in the future.

2) Conceptual Design

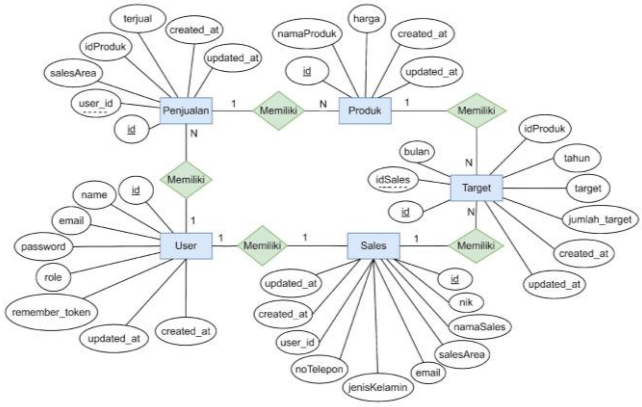

Fig. 3. Entity Relationship Diagram

As can be seen in Fig. 3, the Entity Relationship Diagram illustrates the structure and relationships between entities in the sales system, including the Sales, Product, Target, Salesperson, and User entities. Each entity has important attributes such as ID, name, and creation date. ERD is a tool for designing databases and provides an overview of how the database will work [26].

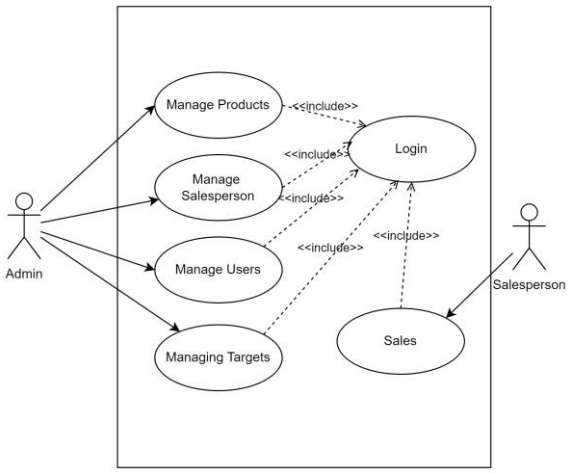

Fig. 4. Use Case Diagram

Fig. 4 shows the system use case diagram with two main actors, namely the Admin and the Salesperson. After logging in, Admin is responsible for managing Products, Salesperson, Users, and Targets. Meanwhile, the Salesperson must also log in to make sales and access sales data.

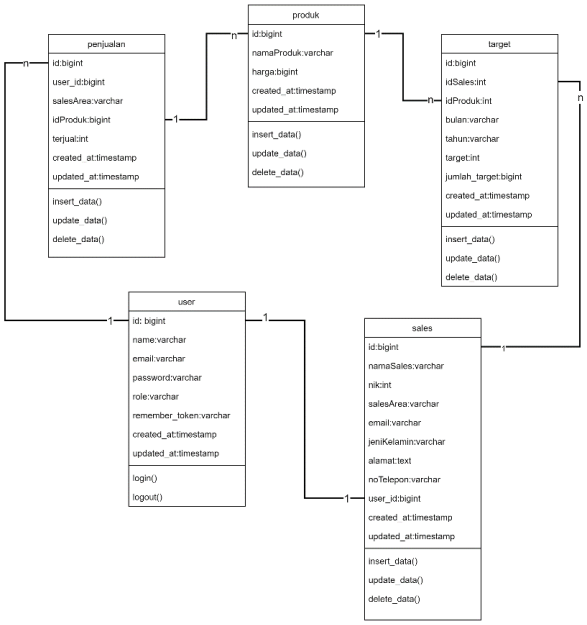

Fig. 5. Relationship Diagram between Tables

Insertable Relationship Diagram is a visual representation that describes the static structure of the system by displaying classes, data types, interfaces, and relationships and interactions between these elements [27].

B. System Implementation Results

1) Implementasi Database

The database implementation in this system is done using MySQL as the database management platform. MySQL was chosen because of its ability to handle various types of transactions and data requests quickly and efficiently, as well as easy to use [28]. The implementation results are shown in Fig. 6.

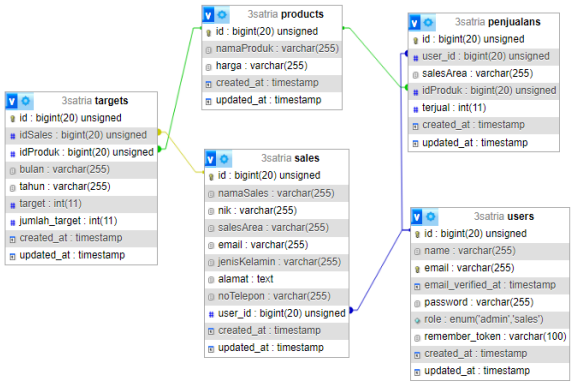

Fig. 6. Database Implementation

2) Implementasi Interface

The interface implementation in this system involves front-end development using Laravel version 11.7.0. Laravel was chosen because of its ability to efficiently manage frontend workflows and provide a variety of features that facilitate development [29]. In addition, Tailwind CSS is implemented to ensure a responsive interface, able to adapt well to various screen sizes and devices, and speed up development by providing utility classes that make it easy to organize interface elements [30], [31].

The login process aims to confirm the identity of the user who wants to access the application and identify the access rights granted to them. In the context of this monitoring system, as shown in Fig. 7, users are required to enter an email and password. If the account information is not registered in the database, a warning message will appear stating that the account was not found. With this mechanism, the system guarantees that only authorized users, such as admin and sales, can access important information, thus maintaining the security of sensitive sales data.

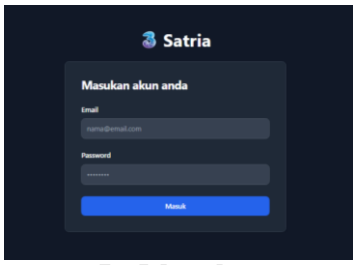

Fig. 7. Login Page

The dashboard page plays an important role in assessing the sales performance of the sales team. The dashboard displays important information, such as sales targets, the number of sales achieved, as well as monthly and annual sales data. Admins can view data on sales name, sales area, products sold, amount sold, and sales input time in one

integrated view. With the data visualization, admins can monitor sales performance in real-time, thus gaining a clear understanding of sales target achievement. This feature facilitates the admin in managing and analyzing sales more effectively. As shown in Fig. 8, the dashboard page provides quick and clear access to the information required by the admin to monitor and make decisions regarding the sales team's sales targets.

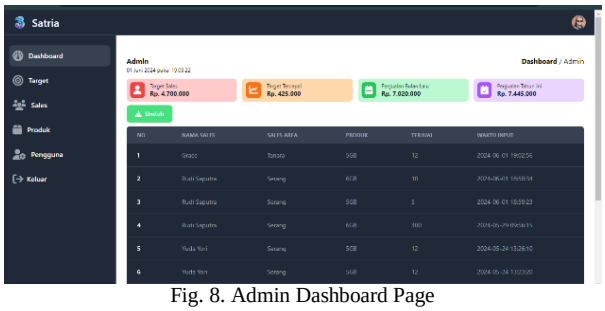

Fig. 8. Admin Dashboard Page

The target page, shown in Fig. 9, serves as a management tool for sales target management. On this page, admins can perform various actions, such as adding new targets, modifying existing targets, or deleting targets that are no longer needed. These features allow admins to customize sales targets to suit their needs and current conditions. Thus, the sales team can stay focused on achieving realistic and measurable targets, and increase effectiveness in sales management.

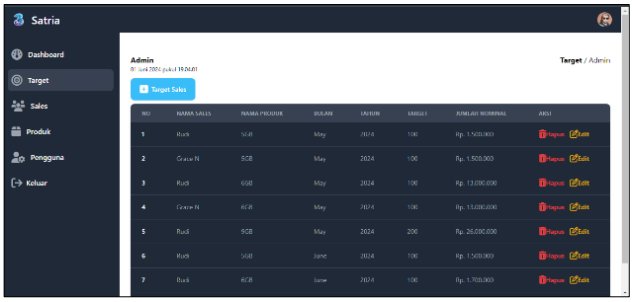

Fig. 9. Salesperson Target Data Page

The product page shown in Fig. 10 serves as a product management tool for admins, allowing the addition of new products, update of existing information, and deletion of irrelevant products. This functionality is crucial in maintaining the accuracy and smooth management of product data, which contributes to sales operations. With a good product management system, the sales team can access the necessary information for sales execution, thus supporting the achievement of set sales targets.

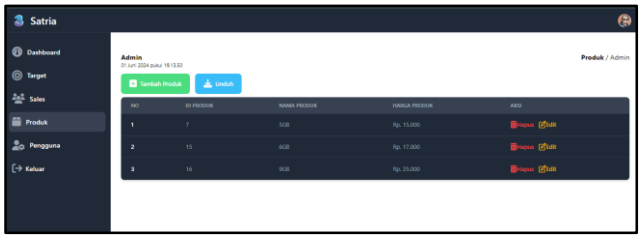

Fig. 10. Product Data Page

The dashboard page for each salesperson is shown in Fig. 11. On this page, each salesperson can monitor various information related to sales performance, including total sales this month, the target set for this month, sales achieved last month, and total sales for the whole year. Sales can also access information regarding the sales target per product that has been set by the admin. With this information, sales can evaluate their achievements in real-time, making it easier to take strategic steps to meet or exceed the sales targets that have been set.

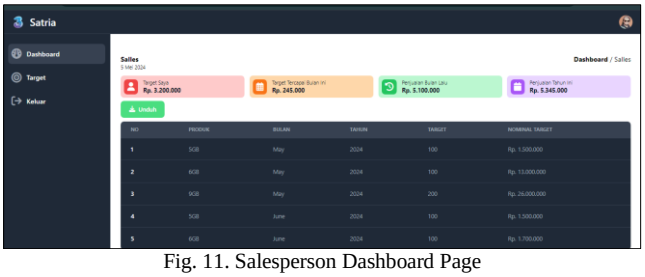

Fig. 11. Salesperson Dashboard Page

The page shown in Fig. 12 allows sales to record product sales transactions. Sales can add new sales as well as modify or delete existing sales data. This transaction recording process plays a significant role in ensuring the consistency and accuracy of sales information, which is an important factor in analyzing sales performance. With efficient data management, sales can generate more precise and informative reports, which support the evaluation of the achievement of predetermined sales targets.

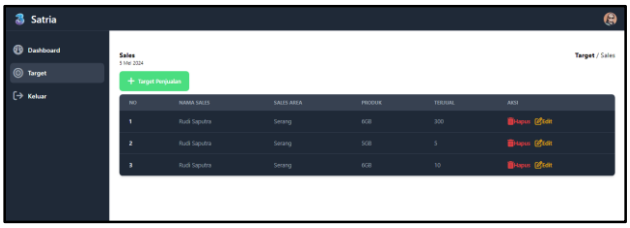

Fig. 12. Sales Target Page

C. System Testing

Load testing using Apache JMeter has been carried out three times with variations in the number of threads (virtual users) of 500, 750, and 1000. The test parameters included a ramp-up period of 1 second and a loop count of 1.

The loading testing targets are:

- Response time < 900ms
- Error Rate < 1%
- Throughput > 150/sec when accessed by 1000 users

Endpoint API load testing results are shown in Fig. 15.

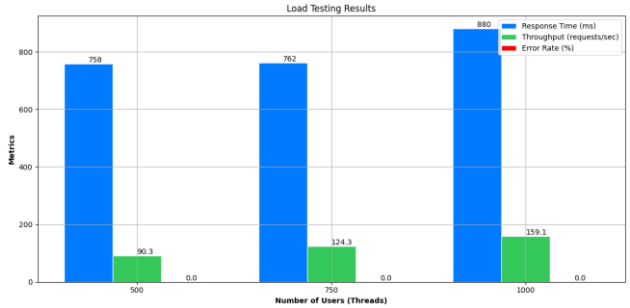

Fig. 15. Load Testing Results

Load testing has been carried out using Apache JMeter to evaluate API performance with a varying number of virtual users, namely 500, 750, and 1000. The test results showed that all scenarios were able to meet the predefined response time targets. Response times were recorded at 758 ms for 500 users, 762 ms for 750 users, and 880 ms for 1000 users, all of which were below the targeted maximum limit of less than 900 ms.

In addition, the tests showed that no errors were detected, with an error rate of 0.00% in all scenarios. This indicates that the system has high reliability in handling requests without failure.

In terms of throughput, for 500 users, the throughput achieved was 90.30 requests per second, while for 750 users the throughput increased to 124.30 requests per second. In the scenario with 1000 users, the throughput reached 159.10 requests per second, which exceeded the expected throughput target of 150 requests per second. These results show that the tested API has good performance in terms of response time and reliability (error rate), and is able to handle high loads.

IV. CONCLUSIONS

This research successfully developed a web-based monitoring system that utilizes RESTful API to increase the effectiveness of monitoring direct sales targets and sales canvassers. The implementation of RESTful API proved to be stable and optimal in handling various load scenarios without errors, as well as increasing throughput along with the increase in the number of users (threads).

The main advantage of the RESTful API is its ability to integrate and access data in real-time, which enables quick monitoring of sales activity and salesperson performance. The use of the lightweight JSON data format also speeds up data download and processing. Thus, the implementation of RESTful API simplifies the monitoring and management of sales data and supports more effective decision-making.

System development is carried out using the Agile method, providing flexibility to changing needs. Load testing using Apache JMeter involved virtual users of 500, 750, and 1000. The results showed that all scenarios met the target response time below 900 ms, with response times recorded at 758 ms for 500 users, 762 ms for 750 users, and 880 ms for 1000 users. The error rate was recorded at 0.00%, indicating the high reliability of the system.

In terms of throughput, the system performed well with 90.30 requests per second for 500 users, 124.30 requests per second for 750 users, and 159.10 requests per second for 1000 users, exceeding the target of 150 requests per second. This confirms that the API can handle high loads with good response times.

As a recommendation, integration of the monitoring system with mobile applications is highly recommended to improve the accessibility and effectiveness of real-time sales performance monitoring. With mobile integration, the sales team and Rural Sales Executive can update and monitor data directly from the field, accelerating decision making. Mobile application development is expected to provide higher flexibility and mobility, facilitating efficient monitoring and management of sales targets anywhere and anytime, without compromising system performance.

REFERENCES

[1] O. Aditia and A. Merthayasa, ‘Upaya Peningkatan Daya Saing Bisnis Perusahaan melalui Manajemen Perubahan’, *Syntax Idea*, vol. 5, no. 7, Jul. 2023, doi: 10.46799/syntax-idea.v5i7.2416.

[2] I. Syafitri, H. Okprana, and I. S. Saragih, ‘Pemilihan Canvaser Terbaik Menggunakan Metode Analitical Network Process Di Indosat Ooredoo’, KLIK: Kajian Ilmiah Informatika dan Komputer, vol. 2, no. 2, pp. 60–68, 2021, doi: https://doi.org/10.30865/klik.v2i2.262.

[3] N. Hanayah and F. Ridho, ‘Rancang Bangun Aplikasi Monitoring Telkomsel Orbit Witel Medan Berbasis WEB Menggunakan Framework Codeigniter’, Jurnal SIKOM (Sistem Informasi Komputer), vol. 1, no. 1, 2024.

[4] M. Iqbal and S. Royal, ‘Penerapan Sistem Terintegrasi Menggunakan Restful Api Pada Dealer Management System Panca Niaga Sei Piring’, Journal of Science and Social Research, no. 1, pp. 219–224, 2023, doi: doi:10.54314/jssr.v6i1.1161.

[5] A. P. S. Wijaya, I. Arwani, and W. H. N. Putra, ‘Pengembangan Sistem Informasi Monitoring Project berbasis Web menggunakan API Trello (Studi Kasus: CV. Kisah Kita Event Organizer)’, vol. 6, no. 7, pp. 3085–3092, 2022, [Online]. Available: http://j-ptiik.ub.ac.id

[6] K. L. Rivaldo, I. K. A. Mogi, I. P. G. H. Suputra, N. A. Sanjaya, D. M. B. A. Darmawan, and I. B. G. Dwidasmara, ‘Sistem Monitoring Tanaman Hidroponik Berbasis Internet of Things menggunakan Restful API’, *Jurnal Elektronik Ilmu Komputer Udayana*, vol. 11, 2022, doi: https://doi.org/10.30812/bite.v2i1.805.

[7] I. O. Suzanti, N. Fitriani, A. Jauhari, and A. Khozaimi, ‘REST API Implementation on Android Based Monitoring Application’, in Journal of Physics: Conference Series, IOP Publishing Ltd, Jul. 2020. doi: 10.1088/1742-6596/1569/2/022088.

[8] R. Amalia Praptiwi, R. A. Praptiwi, A. T. Priandika, and D. Alita, ‘Implementasi REST API pada Manajemen Stok Barang Berbasis Aplikasi Web (Studi Kasus: PT Jon Kuliner Indonesia)’, Universitas Teknokrat Indonesia Jl. ZA Pagar Alam, vol. 3, no. 1, p. 19, 2024, doi: 10.14710/jtk.v3i1.46234.

[9] M. W. Heryatno, ‘Pengembangan Sistem Informasi UII Perkuliahan dengan RESTful Api’, 2020. Accessed: Aug. 17, 2024. [Online]. Available: http://hdl.handle.net/123456789/23536

[10] R. K. Safitri and H. P. Putro, ‘Implementasi REST API untuk Komunikasi Antara ReactJS dan NodeJS’, 2021, Accessed: Jun. 10,

2024. [Online]. Available: https://journal.uii.ac.id/AUTOMATA/article/view/17381
[11] D. Widya Sari, S. Kosasi, Gat, David, and I. Dewa Ayu Eka Yuliani, 'Pemanfaatan RESTful Web Services pada Perangkat Lunak Penyewaan Lapangan Badminton', InfoSys Journal, Feb. 2022, doi: 10.22303/infosys.6.2.2022.103-114.
[12] S. M. Husain, L. Azhari, M. L. Aksani, and S. A. Saputra, 'Analisis Dan Implementasi Fitur Keamanan Aplikasi Pada Framework Laravel', JIKA (Jurnal Informatika), vol. 8, no. 3, p. 281, Jul. 2024, doi: 10.31000/jika.v8i3.11198.
[13] Mangapul Siahaan and R. W. Wijaya, 'Performance Comparison Between Laravel and ExpressJs Framework Using Apache JMeter', JOURNAL OF INFORMATICS AND TELECOMMUNICATION ENGINEERING, vol. 7, no. 2, pp. 545–554, Jan. 2024, doi: 10.31289/jite.v7i2.10571.
[14] F. Sinlae, E. Irwanda, Z. Maulana, and V. E. Syahputra, 'Penggunaan Framework Laravel dalam Membangun Aplikasi Website Berbasis PHP', Jurnal Siber Multi Disiplin, Jul. 2024, doi: 10.38035/jsmd.v2i2.
[15] Y. Darmi, K. Pinandita, and U. Muhammadiyah Bengkulu, 'Implementasi Perbandingan Metode GraphQL dan REST API pada Teknologi Node.js', Journal of Inf*ormation Technology and Computer Science (INTECOMS)*, vol. 7, no. 1, 2024, doi: https://doi.org/10.31539/intecoms.v7i1.8656.
[16] G. Humaswara Prathama, G. Ngurah, and D. Paramartha, 'Rancang Bangun Application Programming Interface Menggunakan Graphql Pada Multi-Platform Application', Jurnal Teknologi Informasi dan Komputer, Jan. 2022, doi: https://doi.org/10.36002/jutik.v8i1.1586.
[17] M. Aryo Wibowo, I. Made Arsa Suyadnya, and K. Oka Saputra, 'Rancang Bangun Aplikasi Game Multiplayer sebagai Alat Bantu Proses Pembelajaran Berbasis WEBSOCKET', vol. 8, no. 3, 2021, doi: https://doi.org/10.24843/SPEKTRUM.2021.v08.i03.p15.
[18] M. A. Albar, E. Anjarwani, B. Irmawati, N. Agitha, and R. Afwani, 'Prosiding Saintek Implementasi Restful Api Pada Sistem Informasi Tracer Study Universitas Mataram Berbasis Mobile', LPPM Universitas Mataram, vol. 4, 2022.
[19] A. Tangella and P. Katari, 'Testing Lifestyle Store Website Using JMeter in AWS and GCP', Blekinge Institute of Technology, 2022.
[20] M. Reza Maulana, E. Budi Susanto, S. Wahyu Binabar, and S. Widya Pratama Pekalongan, 'Analisis Kinerja Website Pemerintah Kota Pekalongan', vol. 19, no. 1, Jun. 2021, doi: https://doi.org/10.54911/litbang.v20i.144.
[21] M. Syahrir, 'Metode Load Balancing Haproxy Pada Opennebula', 2024. Accessed: Nov. 03, 2024. [Online]. Available: https://repository.poliupg.ac.id/id/eprint/8797
[22] Y. Indrasari, 'Efesiensi Saluran Distribusi Pemasaran Kopi Rakyat di Desa Gending Waluh Kecamatan Sempol (IJEN) Bondowoso', *Jurnal Manajemen Pemasaran*, Apr. 2020, doi: 10.9744/pemasaran.14.1.44─50.
[23] Y. Adellia and A. Prajawinanti, 'Implementasi model evaluasi cipp pada pelaksanaan program kelompok belajar TBM Leshutama era pandemi covid-19', Pustaka Karya : Jurnal Ilmiah Ilmu Perpustakaan dan Informasi, vol. 9, no. 2, p. 14, Dec. 2021, doi: 10.18592/pk.v9i2.5516.
[24] A. Nurkholis, A. Budiman, D. Pasha, S. Ahdan, R. Andika, and Z. Amalia, 'Digitalisasi Pelayanan Aadministrasi Surat pada Desa Bandarsari', Journal of Technology and Social for Community Service (JTSCS), vol. 3, no. 1, pp. 21–28, 2022, doi: https://doi.org/10.33365/jsstcs.v3i1.1493.
[25] A. Aulia Aziiza and A. N. Fadhilah, 'Analisis Metode Identifikasi dan Verifikasi Kebutuhan Non Fungsional', Applied Technology and Computing Science Journal, vol. 3, no. 1, 2020, doi: https://doi.org/10.33086/atcsj.v3i1.1623.
[26] S. M. Pulungan, R. Febrianti, T. Lestari, N. Gurning, and N. Fitriana, 'Analisis Teknik Entity-Relationship Diagram Dalam Perancangan Database', Jurnal Ekonomi Manajemen dan Bisnis, vol. 01, no. 2, pp. 143–147, 2022, doi: 10.47233/jemb.v2i1.533.
[27] L. P. Sumirat, D. Cahyono, Y. Kristyawan, and S. Kacung, DASAR-DASAR Rekayasa Perangkat Lunak. 2023. [Online]. Available: www.madzamedia.co.id
[28] S. Nasukha, 'Pengembangan Sistem Informasi Manajemen Pendidik dan Tenaga Kependidikan', Jurnal Al-Fath, vol. 01, no. 1, 2024.
[29] A. Herdiansah, R. Indra Borman, and S. Maylinda, 'Sistem Informasi Monitoring dan Reporting Quality Control Proses Laminating Berbasis Web Framework Laravel', Jurnal TEKNO KOMPAK, vol. 15, no. 2, Aug. 2021, doi: https://doi.org/10.33365/jtk.v15i2.1091.
[30] S. Azhariyah and M. Mukhlis, 'Framework CSS: Tailwind CSS Untuk Front-End Website Store PT. XYZ', Jurnal Informatika, Mar. 2024, doi: https://doi.org/10.57094/ji.v3i1.1601.
[31] K. Arya, B. Wirya Kesuma, I. Nyoman, Y. Anggara Wijaya, I. Gede, and J. E. Putra, 'Implementasi Next.Js, Typescript, Dan Tailwind Css Untuk Pengembangan Aplikasi Frontend Sistem Inventory Perusahaan Apar (Studi Kasus: CV Indoka Surya Jaya)', *JIKOM: Jurnal Informatika dan Komputer*, vol. 14, no. 2, pp. 95–108, Oct. 2024, doi: https://doi.org/10.55794/jikom.v14i2.195.
[32] A. Amarulloh, Kurniasih, and Muchlis, 'Analisis Perbandingan Performa Web Service Rest Menggunakan Framework Laravel, Django, Dan Node Js Pada Aplikasi Berbasis Website', Jurnal Teknik Informatika STMIK Antar Bangsa, Feb. 2023, doi: https://doi.org/10.51998/jti.v9i1.515.
[33] W. Muthia Kansha, Saherih, and Muchlis, 'Analisis Perbandingan Struktur dan Performa Framework Codeigniter dan Laravel dalam Pengembangan Web Application', Jurnal Teknik Informatika STMIK Antar Bangsa, Feb. 2023, doi: https://doi.org/10.51998/jti.v9i1.511.
[34] W. Hadinata and L. Stianingsih, 'Analisis Perbandingan Performa Restfull Api Antara Express.Js Dengan Laravel Framework', *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 12, no. 1, Jan. 2024, doi: 10.23960/jitet.v12i1.3845.