JURNAL INFORMATIKA, Vol. 11 No. 2 October 2025, Page 138-148
ISSN (Print) 2355-6579 | ISSN (Online) 2528-2247
DOI: https://doi.org/10.31294/inf.v12i2.25747

Performance Evaluation of RESTful API in Sales Target Monitoring System for Direct Sales and Sales Canvassers

Suyud Widiono[1*], Restian Dwi Friwaldi[2], Afwan Anggara[3]

[1,2,3] Universitas Teknologi Yogyakarta
Jln. Siliwangi (Ring Road Utara), Jombor, Sleman, Yogyakarta, Indonesia

Correspondence e-mail: suyud.w@uty.ac.id

Submission:	Revision:	Acceptance:	Available Online:
28-04-2025	22-09-2025	02-10-2025	08-10-2025

Abstract

In an increasingly competitive digital era, manual sales target monitoring often leads to delayed information and inefficiency in decision-making. This research aims to develop a web and mobile-based sales target monitoring system integrated with RESTful API to enhance the efficiency of monitoring the performance of direct sales and sales canvassers. The system is developed using the Laravel framework for the back-end and Flutter for the mobile application, with Agile methodology applied in the development process. Testing is conducted using the Black Box Testing method to ensure the accuracy of system functionalities, including user authentication, sales data management, and sales target monitoring. Additionally, load testing is performed using Apache JMeter with scenarios of 500, 750, and 1000 users. The test results show that the system has stable performance with an average response time of 758 ms for 500 users, 762 ms for 750 users, and 880 ms for 1000 users, all below the threshold of 900 ms. The error rate is recorded at 0.00%, and the system throughput exceeds the set target, indicating the system's reliability in handling simultaneous user requests. The conclusion of this research shows that the implementation of RESTful API in the sales monitoring system can improve operational efficiency, enable real-time data exchange, and support faster, data-driven decision-making. As a recommendation, further development could include broader integration with mobile applications and the implementation of AI-based analytics for sales strategy optimization.

Keywords: RESTful API, Sales Monitoring, Agile Methodology

1. Introduction

In today's highly competitive business environment, innovation and the effective use of technology are critical for improving organizational efficiency and effectiveness (Friwaldi & Widiono, 2024). Marketing strategies remain essential in maintaining customer loyalty and expanding market share (Ramadani, 2021). The digital era has posed new challenges to traditional firms, demanding the integration of conventional marketing approaches with digital innovations (Untung Supriadi, 2024). Strong digital literacy within organizations is crucial to support technology adoption and to develop adaptive strategies that respond to market dynamics (Wono et al., 2023). Effective use of social media strengthens company–customer engagement, improves brand awareness, and influences strategic decision-making (Wono et al., 2023). Business innovation, together with rapid technological advancement, creates significant competitive advantages in responding to fast-changing market conditions (Wono et al., 2023). Companies that succeed in the digital age are those capable of combining traditional marketing strategies with digital transformation, investing in digital literacy, building smart social media strategies, and fostering a culture of innovation (Untung Supriadi, 2024; Wono et al., 2023). Prior studies emphasize that real-time information accelerates strategic decision-making in the telecommunications industry, where proper data integration is essential for enhancing unit performance and data management quality (Fadillah & Komalasari, 2024). Furthermore, integrated information systems with robust security measures are strongly correlated with improved data protection and operational performance (Nugroho et al., 2024).

Nevertheless, the main challenge in implementing sales strategies lies in effectively monitoring and managing sales targets. Lack of integration often prevents unified data access, accurate tracking, and innovation, particularly in the logistics sector. Potential solutions include adopting unified platforms or industry standards such as EDI or open APIs to improve interoperability (Karjono et al., 2024). Integrated

information systems are expected to enhance efficiency and provide a strong foundation for future developments in telecommunications (Fadillah & Komalasari, 2024).

Insufficient automation and poor data integration negatively affect sales performance and competitiveness. Strategic decision-making can be improved by adopting big data analytics and artificial intelligence, enabling firms to achieve competitive advantages and long-term growth (Halim & Aspirandi, 2023). While digitalization brings opportunities such as broader market access and new business models, it also poses risks such as cybersecurity threats and privacy concerns (Akhtar Ariq et al., 2023). Business intelligence solutions, such as Microsoft Power BI, facilitate more efficient decision-making by visualizing sales data (Pratama Bukhari et al., 2024). Moreover, the integration of human resource management practices—such as clear HR policies and active HR manager involvement in business strategy—has been shown to positively influence organizational performance and market share (Sigit Auliana & Iis Nuraisah, 2021). These findings highlight the importance of data-driven decision-making and strategic HR integration to ensure competitiveness and business sustainability.

Although numerous studies have demonstrated the successful implementation of RESTful APIs in various sectors, their application in the telecommunications industry remains underexplored, especially for monitoring the performance of direct sales and canvassers. For example, Iqbal & Nurwati (2023) focused on dealer management systems but did not assess system performance under heavy concurrent usage. Eko Septian & Hutabri (2024) highlighted financial data integration but did not address real-time sales target monitoring. Similarly, Ikhwanuzaki & Handayani (2024) emphasized web–mobile integration for transactions, yet overlooked aspects of large-scale reliability and load testing.

This study differs from prior research by focusing on performance evaluation of RESTful APIs in the context of direct sales and canvasser target monitoring within the telecommunications sector—a domain that has received little attention. The system developed in this research is not only functionally tested using Black Box Testing but also stress-tested with Apache JMeter under workloads of up to 1000 concurrent users to assess stability and scalability.

Accordingly, the objectives of this research are: (1) to design and implement a web and mobile-based sales target monitoring system integrated with RESTful API, (2) to verify its functionalities using Black Box Testing, and (3) to evaluate system performance using Apache JMeter. The key contribution of this study is providing empirical evidence that RESTful APIs can maintain stable performance under heavy user loads, while also offering a replicable implementation framework applicable to telecommunications and other industries..

2. Research Methods

This study adopts the Agile Methodology, a system development approach that emphasizes iterative progress, continuous stakeholder feedback, and flexible adaptation to changing requirements. Projects are divided into short cycles, or sprints, allowing ongoing evaluation and adjustments (Adriana N Dugbartey & Olalekan Kehinde, 2025). While Agile enhances collaboration and responsiveness, it also faces challenges such as organizational culture shifts and the need for additional training (Hussien et al., 2024; Zhen, 2024).

The system was designed to integrate direct sales and canvassers, with the RESTful API serving as the backbone for data exchange. The development process consists of the following stages:

- Figure 1. Research Framework – illustrates the research workflow, from literature review, requirements analysis, design, implementation, to testing.
- Figure 5. System Architecture – presents the interaction between the web application, mobile application, RESTful API, and MySQL database.
- Figure 6. Use Case Diagram – shows the interaction of system actors (admin and sales) with the main system functions.
- Figure 7. Entity Relationship Diagram (ERD) – describes the database schema covering users, products, sales, and sales targets.

Each figure is referenced in the text and briefly explained to ensure clarity.

The back-end of the system was developed using Laravel, while the mobile application was built with Flutter, and the web interface was styled with Tailwind CSS. Functional testing was carried out using Black Box Testing to verify user authentication, sales data management, and target monitoring features.

In addition, performance testing (load testing) was conducted using Apache JMeter, an open-source tool widely used for RESTful API evaluation (Abbas et al., 2017). JMeter was chosen for its ability to simulate thousands of concurrent users, flexible configuration, HTTP/HTTPS protocol support, and visual reporting capabilities.

Performance thresholds were defined according to industry standards (Jiang & Hassan, 2015):

- Response time ≤ 900 ms
- Error rate ≤ 1%
- Throughput ≥ 80 requests/sec

The API endpoints under test included:

a. Login Scenario – authentication with valid and invalid credentials.
b. Sales Data Input Scenario – submission of valid and invalid sales records.
c. Target Monitoring Scenario – real-time access to sales target data.
d. Error Handling Scenario – validation of system behavior with empty or incorrectly formatted inputs.

This methodological framework ensures a comprehensive evaluation of the system, covering both functional accuracy and performance stability under high user loads.

This research uses Agile Methodology, a transformative approach to system development that emphasizes iterative progress and continuous feedback from stakeholders. This approach enhances collaboration, flexibility, and rapid adaptation to changing needs. Agile divides projects into small iterations or "sprints," allowing for continuous evaluation and adjustment (Adriana N Dugbartey & Olalekan Kehinde, 2025). Stakeholder involvement helps define needs and validate work results (Adriana N Dugbartey & Olalekan Kehinde, 2025; Djutalov et al., 2024). Although Agile increases response speed and result quality, challenges such as cultural change and training needs can hinder successful implementation (Hussien et al., 2024; Zhen, 2024).

Figure 1. is a system development flow diagram. The following is an explanation of each part of the diagram:

Theory Fundamental & Literature Study, this section shows the initial stage in system development, which includes literature study and fundamental theory to understand the concepts to be applied. In the initial stage of this research, the researcher conducted a literature study by analyzing various relevant sources such as journals, scientific articles, and ebooks related to the topic of sales target monitoring and RESTful API implementation.

Data Collection, the process of collecting the necessary data for research and analyzing system requirements, where the researcher collects data from direct sales and sales personnel through interviews and from related secondary data sources.

System Requirements Analysis, identification of functional and non-functional system requirements, including user authentication, sales data management, and target monitoring. This system requirements analysis stage is based on the collected data, aiming to determine the features and specifications the system must have.

The system is designed to integrate direct sales and sales canvassers, with RESTful API as the backbone of data exchange. The development process includes the following stages:

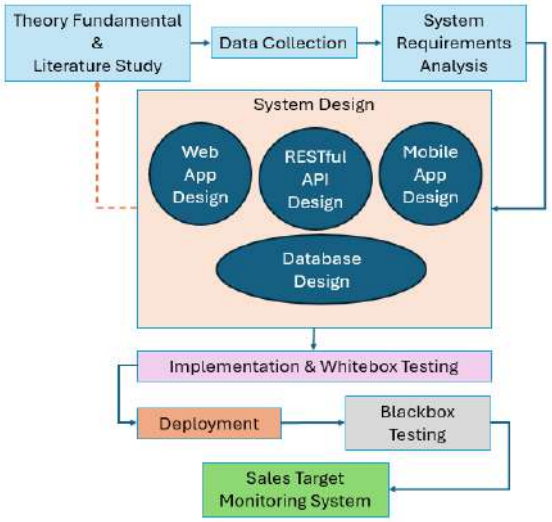

Figure 1. Research Framework

System Design, designing the system architecture, including front-end (web and mobile interfaces), back-end (RESTful API), and database (MySQL). Entity-Relationship Diagram (ERD) and flowchart are used to model the system's structure and workflow. This system design stage consists of several main components:

- Web App Design: Web application design.
- RESTful API Design: API design used for system communication.
- Mobile App Design: Mobile application design.
- Database Design: Database design to store the necessary information.

Implementation & Whitebox Testing, development using the Laravel framework for back-end and Tailwind CSS for front-end. The mobile application is built using Flutter. System implementation based on the created design, followed by whitebox testing (internal testing of the program code).

Deployment, the process of launching or deploying the system after initial testing.

Blackbox Testing, testing system functionalities including user authentication, sales data management, and target monitoring from the user's perspective without viewing the internal code.

Sales Target Monitoring System, the final result of this development process is a sales target monitoring system that has gone through all the above stages.

This diagram shows the workflow in system development, covering several stages from initial research to implementation and testing, with a focus on Black Box Testing and load testing using Apache JMeter. Black Box Testing evaluates software functionality without viewing its internal structure, while load testing assesses performance under various conditions (Jiang & Hassan, 2015). Black Box Testing is conducted based on software specifications to ensure external functions work as needed (Ostrand, 2002). Various types of testing help identify defects and improve software quality (Ostrand, 2002; Saeed & Amjad, 2010) (Ostrand, 2002). Load testing with Apache JMeter simulates multiple users to measure system performance (Abbas et al., 2017).

3. Results and Discussion

The developed system successfully integrates direct sales and canvasser data through a RESTful API, enabling real-time data exchange. The web and mobile interfaces provide convenient access for administrators and sales staff to track targets and performance.

Table 1. System Performance Test Results

Users	Response Time (ms)	Throughput (req/sec)	Error Rate (%)
500	758	90.30	0.00
750	762	124.30	0.00
1000	880	159.10	0.00

Performance evaluation using Apache JMeter confirmed that the system can handle up to 1000 concurrent users. As shown in Table 1 and Figures 5–7, the average response time increased with higher loads: 758 ms (500 users), 762 ms (750 users), and 880 ms (1000 users). Despite this increase, all values remain below the 900 ms threshold (Jiang & Hassan, 2015), indicating stable system performance.

Tables and Figures are presented in Table 1 and Figure 1, and an explanation of the tables and figures must be given.

Throughput increased along with the number of users, reaching 159.10 req/sec at 1000 users, well above the minimum threshold of 80 req/sec. Meanwhile, the error rate remained 0% across all scenarios, reinforcing the system's reliability.

Compared with prior studies, these results highlight significant advantages. For instance, the REST API-based dealer system by Iqbal & Nurwati (2023) did not evaluate performance under large concurrent loads. Similarly, Eko Septian & Hutabri (2024) focused on financial data integration without addressing real-time monitoring, while Ikhwanuzaki & Handayani (2024) emphasized web–mobile integration but did not consider load testing. In contrast, this study provides empirical evidence that RESTful APIs can maintain stable performance with up to 1000 users, offering a scalable solution for sales monitoring in the telecommunications sector.

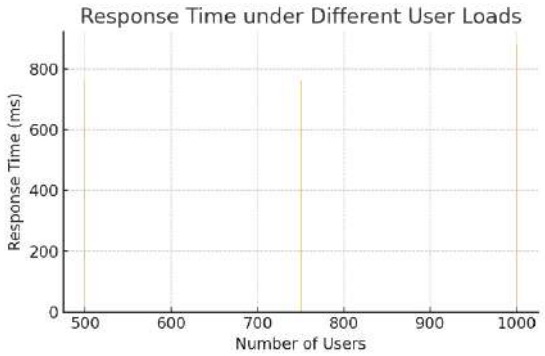

Figure 2. Response Time under Different User Loads

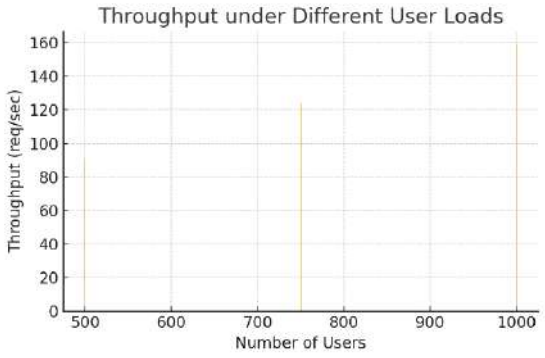

Figure 3. Throughput under Different User Loads

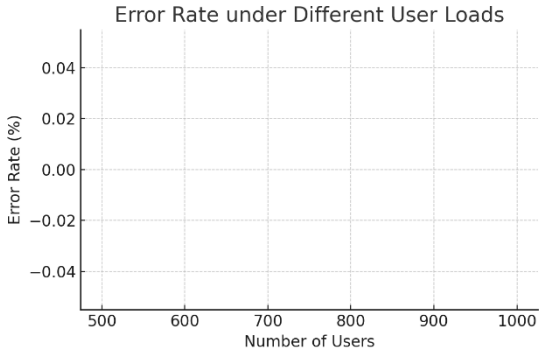

Figure 4. Error Rate under Different User Loads

The changes in response time are illustrated in Figure 2, showing that although the response time increased with higher loads, it remained within acceptable limits. The value of 880 ms under 1000 concurrent users is below the 900 ms threshold (Jiang & Hassan, 2015) and still complies with the ITU-T E.800 (2008) recommendation, which defines 1000 ms as the upper limit for interactive telecommunication services. This confirms that the system remains responsive even under heavy loads.

Meanwhile, Figure 3 demonstrates the upward trend of throughput, reaching 159.10 req/sec at 1000 users. This is significantly higher than the minimum standard of 80 req/sec (Jiang & Hassan, 2015), confirming that the system maintains strong processing capacity.

Furthermore, Figure 4 presents the error rate, which consistently remained at 0% across all test scenarios. This high reliability aligns with Nielsen's (1993) usability guideline, which emphasizes that even minimal failures in interaction can degrade user satisfaction.

3.1. System Architecture Model

Figure 5 shows the proposed system architecture model, explaining that the web-based front-end application acts as the user interface, while the front-end for the mobile application is embedded on each user's device. RESTful API serves as an efficient intermediary for data exchange in JSON format.

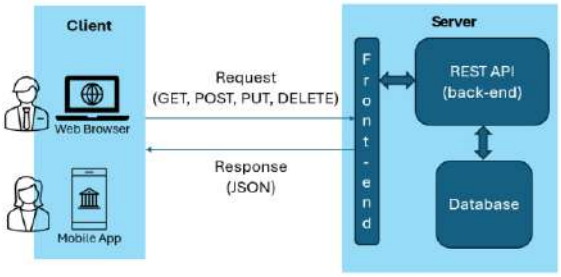

Figure 5. System Architecture

The Architecture Diagram presented in Figure 5 illustrates a client-server system working with REST API to exchange data between web/mobile applications and the server through the HTTP protocol. This diagram represents the architecture in the development of modern cloud-based or web service applications.

The server consists of several main components: Front-end: The user interface displayed in the web or mobile application. REST API (Back-end): An API service that handles client requests, processes them, and returns results in JSON format. Database Server: The data storage used by the system.

Communication flow: Client (Web/Mobile) sends a request to the server. REST API in the backend processes the request and retrieves/modifies data in the Database. REST API returns a response in JSON format to the Client. Client displays the received data to the user.

The Client (User) part consists of two types of applications that can be used by the user: Web Browser: A web-based application accessed through browsers (Chrome, Firefox, etc.). Mobile App: A mobile application running on devices like phones or tablets.

The client sends requests to the server using HTTP methods, such as:

- GET → To retrieve data from the server.
- POST → To send new data to the server.
- PUT → To update existing data on the server.
- DELETE → To delete data on the server.

The server responds to requests with JSON data format.

3.2. System Functionality

The system has several main features, including:

- User Authentication: Ensures only registered users can access the system.
- Sales Data Management: Allows adding, editing, and deleting sales data.
- Target Monitoring: Provides real-time information on sales target achievement.
- Reporting: Generates automatic reports for analysis and decision-making.

Details of the system functionality are illustrated in the following Use-Case Diagram:

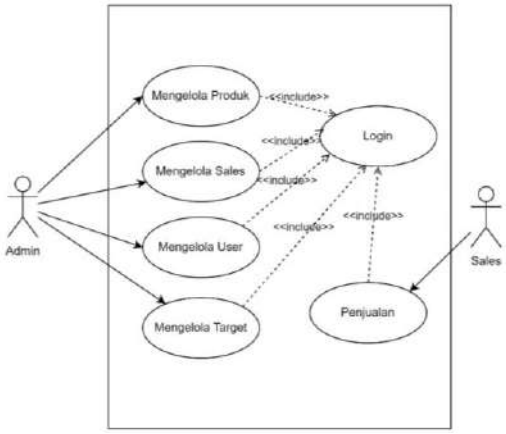

Figure 6. Proposed Use Case Diagram

Figure 6 is a Use Case Diagram used in UML (Unified Modeling Language) based system modeling, designed for the system. It is a design of the functional elements of a system used to manage various business aspects, namely product, sales, user, and target aspects. The diagram also illustrates how the Admin and Sales actors interact with the system. The Admin actor has more access, while the Sales actor only has access to perform sales. All users must log in before accessing system features. This diagram is very useful for understanding the system workflow and who is authorized to perform certain actions.

Actors in the System, there are two actors in this system: Admin and Sales, Admin (as a user with full access rights), and Sales (as a user with limited access rights).

Use Case (Functions in the System), several use cases that can be performed by each actor:

a. Admin can perform:
   - Manage Products
   - Manage Sales
   - Manage Users
   - Manage Targets

b. Sales can perform:
   - Sales

The following is a brief explanation of each element mentioned:

1) Use Case Mengelola Produk: This section is used to add, edit, or delete products from the system.
2) Use Case Mengelola Sales: This section is used to manage sales data, including transaction information and sales performance.
3) Use Case Mengelola Users: This section is used to manage system users, such as adding or deleting users, and setting access rights.
4) Use Case Mengelola Targets: This section is used to set or monitor sales or performance targets.
5) Use Case Login: This section is where users can log into the system with their credentials.
6) Use Case Penjualan: This term refers to the process or data related to sales.

Relationships between Use Cases are marked with <<include>>, meaning that a function requires another function to run. Every use case in the system requires login first, shown by the <<include>> relationship from all processes to Login. Sales require login first, so Sales must log in before performing sales transactions.

3.3. Database Design

Figure 7 shows the database design visualized in an Entity Relationship Diagram (ERD), used to model the database structure used in the developed system. It is a database design diagram and provides an overview of the entity tables and database fields to be created. This diagram is used to design the database structure in the sales system, where there is a relationship between users, sales, products, sales transactions, and sales targets.

The following is an explanation of the ERD in Figure 7, consisting of Main Entities, Relationships Between Entities, and Attributes in Each Entity. The ERD in Figure 7 shows several main entities in the system:

- User (system user)
- Sales (salesperson)
- Sales (sales transaction)
- Product (sold item)
- Target (sales target)

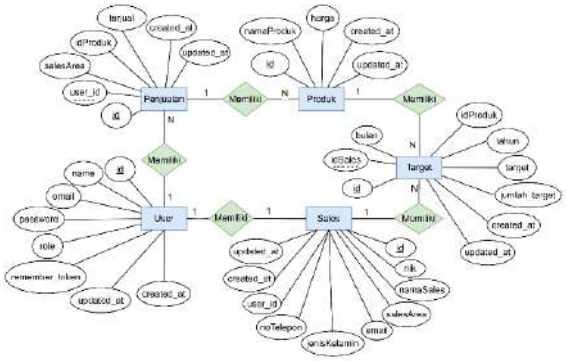

Figure 7. Proposed ERD

The diagram also shows Relationships Between Entities: The User entity has a relationship with the Sales entity, meaning there are users who act as salespersons. The Sales entity performs Sales, which is related to Products. Target is related to Sales, meaning there are targets that must be achieved by salespersons.

The diagram also shows Attributes in Each Entity, each entity has several attributes that store related information:

- User has attributes id, name, email, password, role, etc.
- Product has attributes id, name, price, stock, etc.
- Sales stores transaction data with attributes id, date, quantity, etc.

3.4. REST API Back-end Design

Table 2 User Authentication Endpoints contains a list of API endpoints used to handle user authentication and management processes in the system.

Table 2. Authentication Endpoints.

Endpoint	HTTP Method	Function
/registerUser	POST	Input new user.
/loginUser	POST	User authentication.
/logout	POST	User logs out of the system.
/user/{id}	PUT	Update user.
/user/{id}	DELETE	DELETE Delete user.

The following is an explanation of each Authentication Endpoint:

1) /registerUser

Used to register a new user into the system using the HTTP POST method. The data sent includes name, email, password, and user role. Upon success, the user is saved in the database.

2) /loginUser

Used to authenticate the user, user data is sent using the HTTP POST method based on email and password. If the credentials are correct, the system generates an access token used to access other services in the system.

3) /logout

Used to log the user out of the system and remove the authentication token, making the user's access in this session invalid, so the user must log in again to access the system.

4) /user/{id}

Used to update user data using the HTTP PUT method based on ID. Data that can be updated includes name, email, password, and user role.

5) /user/{id}

Used to delete user data from the system based on user ID, sending the user ID to delete system data using the HTTP DELETE method. After deletion, the user can no longer access the system.

There are several other Endpoints built: User Management Endpoint, Sales Management Endpoint, Product Management Endpoint, Sales Target Management Endpoint, and Sales Management Endpoint.

3.5. Implementation

3.5.1. Database Implementation

The RDBMS used is MySQL, which is used as the system's data storage. Data stored in database tables includes user data, sales data, sales targets, products, and sales data. The REST API Back-End performs CRUD operations on the tables in the database according to requests received from the Front-End. The database structure is designed to facilitate inter-table relationships and maintain system performance when handling sales transactions.

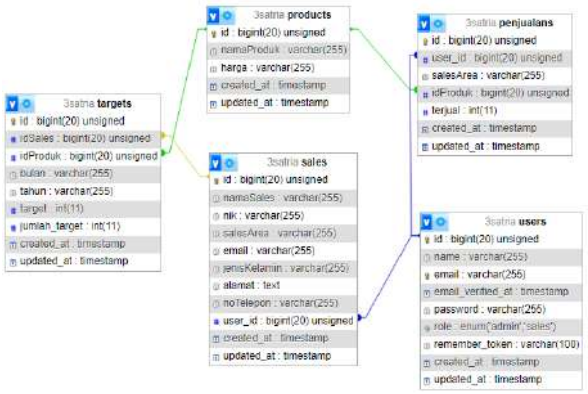

Figure 8. ERD Implementation to Database

Figure 8 shows the database structure implementation of the ERD design, illustrating the relationships between tables in the system's database. The diagram depicts the database structure for the sales management system, focusing on users, sales teams, products, sales transactions, and sales targets. The relationships between tables allow the system to manage data effectively, ensuring each sale and target is linked to the appropriate user and product. The relationships between database tables are explained as follows:

- users → sales (1-to-1) → A user can be a salesperson.
- sales → sales (1-to-many) → A salesperson can perform many sales transactions.
- products → sales (1-to-many) → A product can be sold in many sales transactions.
- sales → targets (1-to-many) → Each salesperson can have many sales targets.
- products → targets (1-to-many) → Each sales target is linked to a specific product.

3.5.2. Back-End Implementation

In the Back-End Server implementation, the system is built using Laravel 10 as the main framework for developing the RESTful API. Laravel is chosen for its structured architecture, good security support, and ease of database management using Eloquent ORM. The database used in this system is MySQL RDBMS, which plays a role in storing and managing user data, sales, products, sales targets, and sales transactions.

The system consists of several main endpoints that support business processes. The first endpoint is the User Authentication Endpoint, which includes registration (`/registerUser`), login (`/loginUser`), logout (`/logout`), and user data management (`/user/{id}`) for updating and deleting user information. This endpoint allows users to securely access the system using an authentication token generated after login.

In addition to authentication, there is a User Management Endpoint that allows the admin to manage registered user data in the system. The admin can perform CRUD (Create, Read, Update, Delete) operations on the user table, such as adding new users, editing user data, viewing user lists, and deleting unnecessary accounts.

Next, the Sales Management Endpoint is designed to manage information related to the sales team. In this endpoint, the admin can register new salespersons, update their information, and delete inactive sales data. This sales data is closely related to user information (`users`) and sales transactions (`sales`).

For managing the products sold, the system provides a Product Management Endpoint, which allows the admin to add new products to the system, update product data such as product name and price, and delete unavailable products. Each sales transaction will refer to the products stored in the products table, ensuring the relationships between data remain structured.

Additionally, the system has a Sales Target Management Endpoint, used to set targets that must be achieved by the sales team. The admin can set targets based on month, product, and the amount to be sold by each salesperson. This

target data will then be used to monitor the salesperson's performance in achieving the set goals.

Finally, the Sales Management Endpoint is responsible for handling sales transactions. This endpoint allows salespersons to record sales that have been made, including the products sold and the quantity sold. Each transaction will be recorded in the sales table, linked to the users table (as the salesperson performing the transaction) and the products table (the product sold).

All endpoints in this system are implemented using Laravel API Routes, where each request is controlled by a controller that handles business logic. Data validation is performed using Laravel Request Validation, while database access is managed with Eloquent ORM to ensure efficiency in data retrieval and manipulation. By using JWT or Sanctum authentication middleware, each request to the system is first checked to ensure only authorized users can access certain features.

With a structured architecture and well-integrated between Laravel 10 and MySQL, this system is able to provide an efficient solution in managing user data, sales, products, sales targets, and transactions occurring in business processes.

3.5.3. Front-End Implementation

The interface in this system involves Front-End development using Laravel version 11.7.0. The Laravel framework is chosen for its ability to manage front-end workflows efficiently and provide various features that facilitate development. Additionally, 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 facilitate interface element organization.

The login process aims to confirm the identity of users who want to access the application and identify the access rights granted to them. In the context of this system, users are required to enter their email and password. If the account information is not registered in the database, a warning message will appear stating that the account is not found.

With this mechanism, the system ensures that only authorized users, such as admins and salespersons, can access important information, thus maintaining the security of sensitive sales data.

Other parts of the Web-based Front-End implementation include: Admin Dashboard Page, Salesperson Target Data Page, Product Data Page, Salesperson Dashboard Page, Sales Target Page

a. Web App Login

b. Mobile App Login

Figure 9. Login Page

Other parts of the Mobile-based Front-End implementation besides the Login Form above include: Sales Dashboard, Sales Page, Add and Edit Sales Form.

3.6. System Testing

Testing on this system was carried out using 2 (two) methods, namely the Backend API load testing method and the Black-Box Testing testing method.

3.6.1. API Back-end Load Testing Results

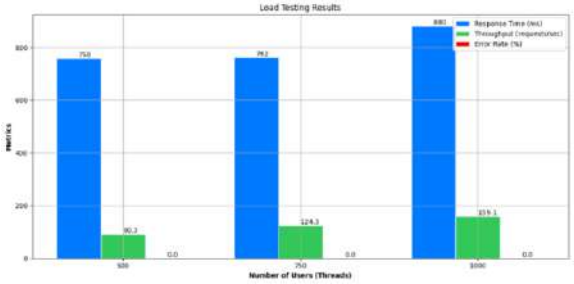

Figure 10. API Back-end Load Testing Results

The graph in Figure 10 shows the results of load testing using Apache JMeter to measure API performance based on three main metrics:

1) Response Time (ms), Shown by the blue bar graph. Illustrates the average time the API takes to respond to user requests. It can be seen that the response time increases as the number of users increases (500, 750, and 1000). For 500 users, the response time is 758 ms, then slightly increases to 762 ms for 750 users, and significantly increases to 880 ms for 1000 users. This indicates that additional load begins to impact API response time.
2) Throughput (requests/sec), shown by the green bar graph. Measures the number of requests processed by the API per second. Throughput increases as the number of users increases, from 90.3 requests/sec (500 users) to 124.3 requests/sec (750 users), and finally 159.1 requests/sec (1000 users). This shows that the system is still able to handle more requests without experiencing overload.
3) Error Rate (%), shown by the red bar graph. Measures the percentage of errors that occur

during testing. The value is 0% in all scenarios, meaning the API successfully handles all requests without failure, despite the increase in the number of users.

3.6.2. Black-Box Testing Results

Black-box testing is conducted to ensure that all system features function as expected. The test results show that the system can:

- Validate user login correctly.
- Add, edit, and delete sales data.
- Monitor sales targets in real-time.
- Generate accurate reports.

System testing is conducted using the Black Box method to ensure that all functionalities in the system work according to the specified requirements. This method focuses on testing system inputs and outputs without viewing the source code or internal structure.

1) Black Box Testing Method

Black Box Testing is conducted by testing each main system feature, including:

- User Management (registration, login, logout, and access rights management)
- Sales Data Management (input, update, and deletion of sales data)
- Sales Target Monitoring (setting targets, achievement, and data visualization)
- RESTful API Integration (communication between front-end and back-end)

2) Testing Scenarios and Results

Testing is conducted based on several scenarios reflecting real-world system usage. The following is a summary of the results:

Table 3. Authentication Endpoints

No	Feature	Testing Scenario	Expected Result	Result
1	Login	Input correct credentials	User successfully logs in	☑OK
2	Login	Input wrong password	Error message appears	☑OK
3	Input Sales Data	Input valid data	Data saved in database	☑OK
4	Input Sales Data	Input empty data	Error warning appears	☑OK
5	Target Monitoring	Access the sales target page	Data is displayed correctly	☑OK
6	GET API Endpoint	Accessing data with the GET method	Data is displayed in JSON format	☑OK
7	POST API Endpoint	Sending data using the POST method	Data successfully added	☑OK

The test results show that all main system features function according to the specified requirements. No critical errors are found during testing, and all scenarios run as expected. Therefore, the system is considered to have met user needs and is ready to be implemented in company operations.

3.7. Discussion

The implementation of this system shows significant improvement in the efficiency of sales target monitoring. RESTful API enables fast and integrated data exchange, while Agile methodology ensures the system can adapt to changing needs. The user-friendly web and mobile interfaces make it easier for administrators and sales personnel to access information and perform their tasks.

The developed sales target monitoring system using RESTful API has been successfully implemented in the form of a web-based application. The Laravel framework is used to build the backend RESTful API, while the frontend is developed using responsive web-based technology and mobile-based using Flutter. MySQL is used as the database management system to store information related to sales, sales targets, and user data.

The system architecture is designed to enable efficient communication between front-end and back-end through RESTful API. With this approach, the system can access and update data in real-time, thereby improving accuracy and efficiency in sales target monitoring.

Black Box Testing is conducted to ensure that each system feature functions according to the specified requirements. The test results show that all main features, including login, sales data input, target monitoring, and RESTful API communication, work well without critical errors.

Load Testing with Apache JMeter aims to measure system performance in handling multiple users simultaneously. The testing scenarios involve 500, 750, and 1000 users with the following results:

- 500 users: Response time 758 ms, throughput 90.30 requests/sec
- 750 users: Response time 762 ms, throughput 124.30 requests/sec
- 1000 users: Response time 880 ms, throughput 159.10 requests/sec

The test results show that the system can handle increasing numbers of users without significant performance degradation.

Performance analysis of the implemented RESTful API shows stable and optimal performance in handling data requests. With the use of JSON format as the data exchange medium, the system can quickly access and display information. Additionally, the API architecture allows good scalability for future development.

The success of this system is also supported by the use of modern technologies such as Laravel and Flutter, ensuring optimal system performance and good scalability. Black-box testing confirms that this system is reliable and meets user needs.

4. Conclusion

This research has successfully developed a web-based sales target monitoring system with RESTful API implementation to enhance the effectiveness of monitoring direct sales and sales canvassers' targets. The use of RESTful API enables real-time data exchange, more flexible integration with other systems, and increases the speed of access and information management.

The conducted tests show that this system has high reliability and stable performance. Based on the Black Box Testing results, all main system features, including login, user management, sales recording, and target monitoring, have functioned according to the specified requirements. Additionally, load testing with Apache JMeter shows that the system can handle increasing numbers of users without significant performance degradation. The response time remains below the specified threshold (900 ms), with an error rate of 0.00% and throughput meeting performance standards. The main advantage of this system is its ability to improve efficiency and accuracy in monitoring sales target achievement. With the use of the Laravel framework and MySQL database, this system has a scalable structure and can be further developed.

However, there are still some limitations, such as the completeness and integration of features in the mobile application with other external APIs, such as accessing maps to track salesperson locations and others. Therefore, for further development, it is recommended to integrate the system with other external systems with the mobile application to increase flexibility and effectiveness in real-time sales monitoring. Additionally, enhancing AI-based analytics features can help in more accurate and strategic sales target predictions.

With the obtained results, this research contributes to the application of RESTful API for sales monitoring systems, especially in the telecommunications industry, and serves as a foundation for the development of more advanced monitoring technologies in the future.

Reference

Abbas, R., Sultan, Z., & Bhatti, S. N. (2017). Comparative Study of Load Testing Tools: Apache JMeter, HP LoadRunner, Microsoft Visual Studio (TFS), Siege. Sukkur IBA Journal of Computing and Mathematical Sciences, 1(2), 102–108. https://doi.org/10.30537/sjcms.v1i2.24

Adriana N Dugbartey, & Olalekan Kehinde. (2025). Optimizing project delivery through agile methodologies: Balancing speed, collaboration and stakeholder engagement. World Journal of Advanced Research and Reviews, 25(1), 1237–1257. https://doi.org/10.30574/wjarr.2025.25.1.0193

Akhtar Ariq, M., Hanggara Putra Anwar, N., & Aulia Rahma, S. (2023). Dampak Digitalisasi Terhadap Bisnis Dan Perdagangan. Journal of Comprehensive Science (JCS), 2(6), 1801–1816. https://doi.org/10.59188/jcs.v2i6.401

Djutalov, R., Irham, M., Putra, A. D., Elistiawan, A., & Syah, R. F. (2024). Agile Methodology in Web and Mobile Based App of Tracer Study and Career Center. Brilliance: Research of Artificial Intelligence, 4(1), 453–462. https://doi.org/10.47709/brilliance.v4i1.4372

Eko Septian, D., & Hutabri, E. (2024). Optimasi Sistem Akuntansi Berbasis Web dengan Integrasi RESTful API: Studi Kasus pada PT Segara Catur Perkasa dalam Bidang Pemanduan dan Penundaan Kapal Menggunakan Metode Scrum. Jurnal Informasi Dan Teknologi, 70–79. https://doi.org/10.60083/jidt.v6i1.476

Fadillah, I. Z., & Komalasari, R. (2024). Rancang Bangun Sistem Informasi Manajemen Operation OLT Aktual dan Ter-Integrasi. TeknoIS : Jurnal Ilmiah Teknologi Informasi Dan Sains, 14(1), 80–90. https://doi.org/10.36350/jbs.v14i1.234

Friwaldi, R. D., & Widiono, S. (2024). Optimization of Direct Sales and Sales Canvasser Sales Target Monitoring With RESTful API Implementation on Web-Based Monitoring System. Journal of Applied Informatics and Computing, 8(2), 623–630. https://doi.org/10.30871/jaic.v8i2.8644

Halim, M., & Aspirandi, R. M. (2023). Peran Akuntansi Manajemen Strategik Terhadap Pengambilan Keputusan Bisnis Melalui Analisis Big Data Dan Artificial Intelligence: Suatu Studi Literature Review. JIAI (Jurnal Ilmiah Akuntansi Indonesia), 8(1), 110–128. https://doi.org/10.32528/jiai.v8i1.11878

Hussien, H. S., Mhd Salim, M. H., Mat Nayan, N., Ijab, M. T., Julita, A. R. B., Mukhtar, Z., Shahril, M. S., Ramli, N. F. L., & Omar, A. M. (2024). Development Of Student-Centered Graduate Supervision Management System Based On Agile Methodology. Journal of Information System and Technology Management, 9(36), 53–66. https://doi.org/10.35631/JISTM.936004

Ikhwanuzaki, M. F., & Handayani, I. (2024). Implementasi Web Service Menggunakan Restful Api pada Aplikasi Pemesanan Sarung Goyor Suhutex. Jurnal Riset Dan Aplikasi Mahasiswa Informatika (JRAMI), 5(1), 191–199.

https://doi.org/10.30998/jrami.v5i1.10486
Iqbal, M., & Nurwati, N. (2023). Penerapan Sistem Terintegrasi Menggunakan Restful Api Pada Dealer Management System Panca Niaga Sei Piring. Journal Of Science And Social Research, 6(1), 219. https://doi.org/10.54314/jssr.v6i1.1161
Jiang, Z. M., & Hassan, A. E. (2015). A Survey on Load Testing of Large-Scale Software Systems. IEEE Transactions on Software Engineering, 41(11), 1091–1118. https://doi.org/10.1109/TSE.2015.2445340
Karjono, K., Diah Kusumawati, E., Karmanis, K., & Kusumaningrum, D. (2024). Transformasi Pemasaran Industri Logistik Dalam Meningkatkan Efisiensi Dan Keunggulan Kompetitif. Majalah Ilmiah Bahari Jogja, 22(2), 125–136. https://doi.org/10.33489/mibj.v22i2.364
Nugroho, A., Syaifudin, R., & Fauziawan, A. I. (2024). Analisis Dampak Keamanan IoT dan Integrasi Sistem Informasi terhadap Perlindungan Data dan Kinerja Operasional di Perusahaan Telekomunikasi Yogyakarta. Jurnal Multidisiplin West Science, 3(05), 611–623. https://doi.org/10.58812/jmws.v3i05.1197
Ostrand, T. (2002). Black-Box Testing. In Encyclopedia of Software Engineering. Wiley. https://doi.org/10.1002/0471028959.sof022
Pratama Bukhari, A., Hafidz, R., & Prio Pamungkas, R. W. (2024). Analisis Business Intelligence Data Penjualan Pt Ambulance Pintar 2021. Jatl (Jurnal Mahasiswa Teknik Informatika), 8(4), 7184–7189. https://doi.org/10.36040/jati.v8i4.10141
Ramadani, A. (2021). Strategi Pemasaran. https://doi.org/10.31219/osf.io/68v59
Saeed, U., & Amjad, M. A. (2010). BLACK BOX TESTING STRATEGIES FOR FUNCTIONAL TESTING: ISTQB: Black Box testing Strategies used in Financial Industry for Functional testing.
Sigit Auliana, & Iis Nuraisah. (2021). Integrasi Strategis Sumber Daya Manusia Untuk Kinerja Organisasi. Jurnal Bina Bangsa Ekonomika, 14(1), 79–92. https://doi.org/10.46306/jbbe.v14i1.60
Untung Supriadi. (2024). Menghadapi Tantangan Era Digital: Inovasi dalam Manajemen Pemasaran untuk Perusahaan Tradisional. JPNM Jurnal Pustaka Nusantara Multidisiplin, 2(1). https://doi.org/10.59945/jpnm.v2i1.138
Wono, H. Y., Supriaddin, N., Amin, F., Indriastuti, Y., & Sufa, S. A. (2023). Media Sosial, Literasi Digital, Dan Inovasi Bisnis Trikotomi Baru Dalam Manajemen Strategi. Branding: Jurnal Manajemen Dan Bisnis, 2(1). https://doi.org/10.15575/jb.v2i1.29329
Zhen, Z. (2024). Research and Practice of Agile Software Development Methods. Applied and Computational Engineering, 114(1), 186–190. https://doi.org/10.54254/2755-2721/2024.18284