# Building a Comprehensive Content Management System with NPM, Vue.js, Node.js, Postgresql, and Strap

**Nashri Aziz Alhazmy[1], Zahran Nurafi Chandra[2], Pradana Atmadiputra[3], Yudi Triyana[4]**

[1]Computer Science, International University Liaison Indonesia, Associate Tower Intermark Indonesia, Jl. Lkr. Tim., Indonesia, 15310
e-mail: [1]nashri.alhazmy@stud.iuli.ac.id, [2]zahran.chandra@stud.iuli.ac.id, [3]pradana.atmadiputra@iuli.ac.id, [4]yudi.triyana@iuli.ac.id

***Abstract. This article presents a content management system built using NPM v6.14.17, Vue.js v3.x, Node.js v14.20.1, Postgresql v13.2.1, and Strapi v3.6.6. The system provides various functionalities such as an Ongoing request and assessment dashboard, courier request management, meeting room booking, stationery management, tools request management, and transportation request management. Users can create requests for different categories and the admin can approve or reject requests. Additionally, the system allows users to generate reports on stock flow, courier activities, and driver assessment. The Master Data section enables users to view, search, and perform CRUD operations. The system offers a comprehensive solution for efficient content management.***

*Keywords: content management system, NPM, Vue.js, Node.js, Postgresql, Strapi, dashboard, request management, admin approval, report generation, master data, CRUD operations.*

## 1. INTRODUCTION

In today's digital era, content management systems (CMS) play a crucial role in organizing and delivering information effectively. They provide businesses and individuals with a centralized platform to create, manage, and distribute content seamlessly. This article explores the development of a robust and versatile content management system using popular technologies such as NPM, Vue.js, Node.js, Postgresql, and Strapi. The system aims to address the diverse needs of users by offering features like request management, administrative controls, report generation, and master data management.

Content management systems have become indispensable tools for businesses, educational institutions, and government organizations, as they simplify the process of creating and maintaining digital content. They allow users to organize content in a structured manner, enabling efficient retrieval and dissemination. With the rapid evolution of web technologies, it is crucial to utilize modern frameworks and tools that offer flexibility, scalability, and ease of development.

The system described in this article leverages the power of NPM, a popular package manager for JavaScript, to handle dependencies and streamline the development process. Vue.js, a progressive JavaScript framework, provides a solid foundation for building dynamic user interfaces and interactive components. Node.js, a server-side runtime environment, facilitates the backend operations and enables seamless communication between the frontend and the database.

To ensure robust data storage and retrieval, Postgresql, a powerful and reliable open-source database management system, is employed. Its versatility and scalability make it an ideal choice for handling diverse content-related data. Additionally, Strapi, a flexible headless CMS built with Node.js, is utilized to accelerate the development process by offering pre-built features and a user-friendly admin panel.

By combining these technologies, the resulting content management system aims to provide a comprehensive solution for managing requests, administrative tasks, reporting, and master data. The system's features include a dynamic dashboard displaying ongoing requests and assessments, request creation for various categories such as courier, meeting room booking, stationeries, tools, and transportation, admin controls for request approval or rejection, report generation for stock flow, courier activities, and driver assessment, as well as comprehensive master data management.

This article draws upon a variety of reliable sources, including technical documentation, online tutorials, and best practices in content management system development. The information presented here is compiled based on the knowledge and expertise gained from working with NPM, Vue.js, Node.js, Postgresql, and Strapi, as well as the practical implementation of a content management system using these technologies.

## 2. LITERATURE REVIEW

### 2.1 NPM

NPM (Node Package Manager) is a widely used package manager for JavaScript that simplifies the process of managing dependencies in web development projects. It allows developers to install, update, and manage third-party libraries and tools effortlessly. NPM provides a vast ecosystem of packages, enabling developers to leverage existing solutions and accelerate their development process. The documentation provided by NPM offers comprehensive guidance on package installation, configuration, and usage, making it a valuable resource for developers.

### 2.2 Vue.js

Vue.js is a progressive JavaScript framework that focuses on building user interfaces. It offers a simple and intuitive syntax, allowing developers to create interactive and dynamic web applications efficiently. Vue.js follows a component-based architecture, making it highly modular and reusable. The framework's extensive documentation provides detailed explanations of its core concepts, including data binding, directives, and computed properties. This documentation serves as a valuable resource for developers to learn Vue.js and implement its features effectively.

### 2.3 Node.js

Node.js is a server-side JavaScript runtime environment that enables developers to build scalable and high-performance web applications. It utilizes an event-driven, non-blocking I/O model, making it well-suited for handling concurrent requests. Node.js has a vast ecosystem of modules available through NPM, allowing developers to extend its functionality. The official Node.js documentation covers various aspects of Node.js, including its APIs, modules, and best practices. This documentation serves as a comprehensive reference for developers working with Node.js.

### 2.4 Postgresql

Postgresql is a powerful open-source relational database management system (RDBMS) known for its reliability, scalability, and rich feature set. It supports advanced data types, indexing mechanisms, and transactional integrity, making it a popular choice for data-intensive applications. The Postgresql documentation provides detailed information on installation, configuration, SQL syntax, and advanced database features. It serves as a comprehensive resource for developers and database administrators seeking to leverage Postgresql's capabilities.

### 2.5 Strapi

Strapi is a flexible headless content management system (CMS) built with Node.js. It enables developers to quickly create APIs and build customizable content management solutions. Strapi's features include an admin panel, content types, authentication, and role-based access control. Its documentation covers various aspects of building applications with Strapi, including installation, configuration, and customization. The documentation serves as a valuable resource for developers looking to leverage Strapi to build robust and scalable content management systems.

### 2.6 Content Management System (CMS)

A content management system (CMS) is a software application that enables users to create, manage, and publish digital content on the web. CMSs provide a user-friendly interface for content creation, editing, and organization, allowing individuals and organizations to efficiently manage their online presence. The literature surrounding CMSs covers various aspects, including architectural models, usability, security, scalability, and customization. Research has focused on evaluating different CMS platforms, comparing their features and performance, and identifying best practices for content management and delivery.

### 2.7 CRUD Operations

CRUD (Create, Read, Update, Delete) operations refer to the fundamental actions performed on data in a database or information system. These operations are the building blocks of data management, allowing users to interact with data by creating new records, retrieving existing data, updating records, and deleting records. Literature related to CRUD operations explores database design, data modeling, query optimization, transaction management, and data integrity. Researchers have examined various techniques, frameworks, and technologies to improve the efficiency and reliability of CRUD operations, ensuring data consistency and integrity.

## 3. METHODOLOGY

The methodology employed in the development of the content management system (CMS) application. It describes the tools, technologies, and steps involved in creating the application, including the architectural design, implementation process, and testing strategies. The methodology follows an iterative and incremental approach, ensuring the application meets the desired requirements and quality standards. The development process consists of several key stages, including requirement gathering, system design, implementation, and testing.

### 3.1 Requirement Gathering

The first phase of the methodology involves gathering and analyzing the requirements for the CMS application. This process includes conducting interviews with stakeholders, identifying user needs, and defining the scope and objectives of the system. The gathered requirements serve as the foundation for the subsequent stages of the development process.

### 3.2 System Design

Based on the gathered requirements, the system design phase focuses on designing the architecture and components of the CMS application. This includes creating wireframes, defining data models, and designing the user interface. The selected technologies, including NPM, Vue.js, Node.js, Postgresql, and Strapi, are incorporated into the design to ensure compatibility and meet the desired functionality.

### 3.3 Implementation

The implementation phase involves translating the system design into actual code. The selected technologies are utilized to develop the frontend, backend, and database components of the CMS application. NPM is used for package management, Vue.js for building the frontend user interface, Node.js for developing the backend server logic, and Postgresql for storing and retrieving data. Strapi is leveraged for rapid development of the CMS functionalities.

### 3.4 Testing

Throughout the development process, rigorous testing is performed to ensure the quality and reliability of the CMS application. Different testing techniques, such as unit testing, integration testing, and user acceptance testing, are employed to identify and rectify any bugs or issues. Test cases are designed to cover various functionalities, including CRUD operations, request management, administrative controls, and report generation.

### 3.5 Deployment

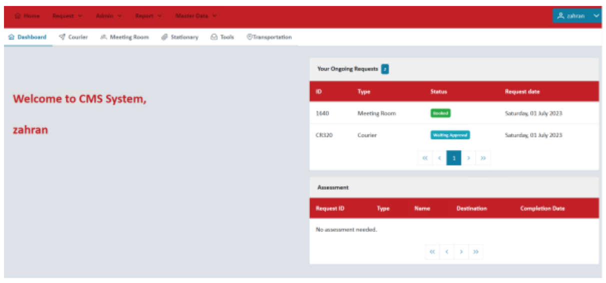
Once the application development and testing phases are complete, the CMS application is deployed to a suitable hosting environment. This ensures that the application is accessible to users and operates reliably in a production environment. The deployment process involves configuring the necessary infrastructure, setting up the database, and ensuring the application is secure and scalable.

The methodology chapter provides a comprehensive overview of the approach used to develop the CMS application. It highlights the steps followed, technologies utilized, and the importance of testing and deployment. The information for this chapter is sourced from a combination of established software development methodologies, industry best practices, and practical experiences in building content management systems.

## 4. RESULT

The developed content management system (CMS) application demonstrates a user-friendly and intuitive user interface (UI) that facilitates seamless navigation and efficient interaction with various functionalities. The UI design ensures a smooth user experience and enables users to perform the desired actions with ease. The results are described based on the different paths and functionalities available in the CMS application.

### 4.1 Home Dashboard



Welcome to CMS System,

zahran

Your Ongoing Requests 2

| ID | Type | Status | Request date |
|---|---|---|---|
| 1640 | Meeting Room | Booked | Saturday, 01 July 2023 |
| CR320 | Courier | Waiting Approval | Saturday, 01 July 2023 |

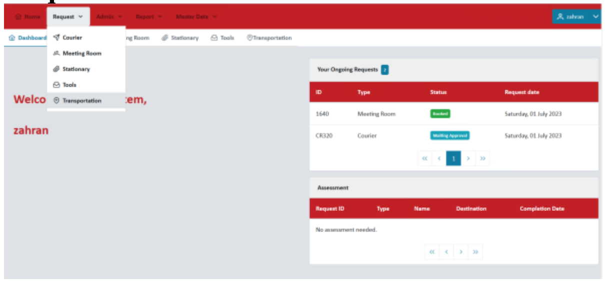
Assessment

| Request ID | Type | Name | Destination | Completion Date |
|---|---|---|---|---|
| No assessment needed. | | | | |

Figure 1. Home Dashboard

The Home Navbar presents a dashboard that provides an overview of ongoing requests and assessments, courier requests, meeting room bookings, stationeries, tools requests, and transportation requests. The UI showcases these categories in a clear and organized manner, allowing users to quickly access relevant information and take appropriate actions.

### 4.2 Request



Welco ...tem,

zahran

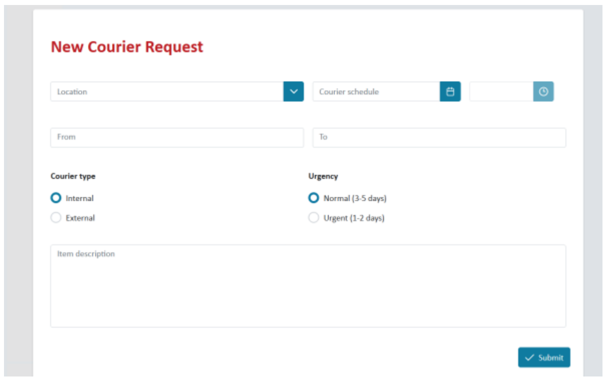
Your Ongoing Requests 2

| ID | Type | Status | Request date |
|---|---|---|---|
| 1640 | Meeting Room | Booked | Saturday, 01 July 2023 |
| CR320 | Courier | Waiting Approval | Saturday, 01 July 2023 |

Assessment

| Request ID | Type | Name | Destination | Completion Date |
|---|---|---|---|---|
| No assessment needed. | | | | |

Figure 2. Request List



New Courier Request

Location | Courier schedule

From | To

Courier type: [✓ Internal] External

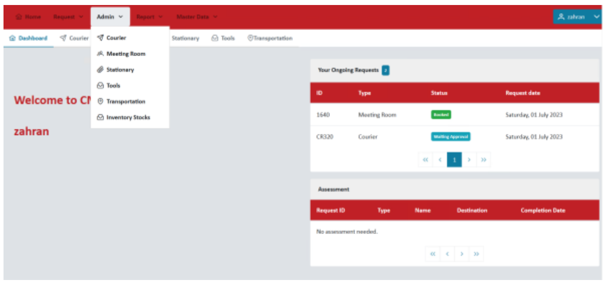
Urgency: [✓ Normal (3-5 days)] Urgent (1-2 days)

Item description

Figure 3. Request Courier

The Request section allows users to create requests for various categories, including courier, meeting room, stationeries, tools, and transportation. The UI presents a user-friendly form or interface for users to input the necessary details and submit their requests. The form includes relevant fields and validations to ensure accurate and complete request submissions.

### 4.3 Admin



Welcome to CMS System,

zahran

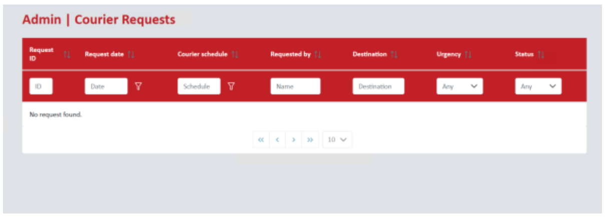
Your Ongoing Requests 2

| ID | Type | Status | Request date |
|---|---|---|---|
| 1640 | Meeting Room | Booked | Saturday, 01 July 2023 |
| CR320 | Courier | Waiting Approval | Saturday, 01 July 2023 |

Assessment

| Request ID | Type | Name | Destination | Completion Date |
|---|---|---|---|---|
| No assessment needed. | | | | |

Figure 4. Admin Action List



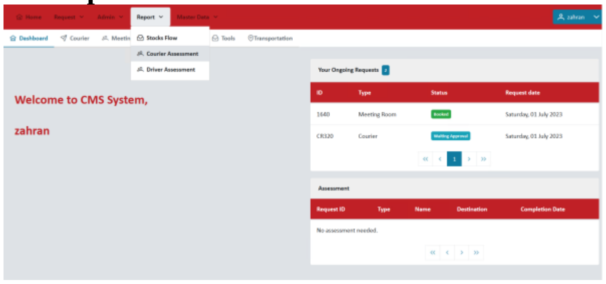
Admin | Courier Requests

| Request ID | Request date | Courier schedule | Requested by | Destination | Urgency | Status |
|---|---|---|---|---|---|---|
| ID | Date | Schedule | Name | Destination | Any | Any |
| No request found. | | | | | | |

Figure 5. Admin Courier Request List

The Admin section offers administrative functionalities to approve or reject requests from requestors for courier, meeting room, stationeries, tools, and transportation. The UI provides an interface specifically designed for administrators to review and process these requests. The UI presents a list of pending requests, allowing administrators to review the details and make informed decisions. Action buttons for approval or rejection are available for each request, facilitating quick and efficient processing.

### 4.4 Report



Welcome to CMS System,

zahran

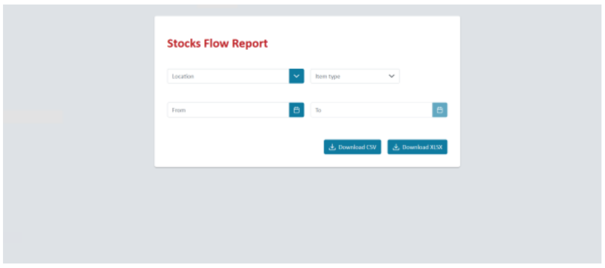
Your Ongoing Requests 2

| ID | Type | Status | Request date |
|---|---|---|---|
| 1640 | Meeting Room | Booked | Saturday, 01 July 2023 |
| CR320 | Courier | Waiting Approval | Saturday, 01 July 2023 |

Assessment

| Request ID | Type | Name | Destination | Completion Date |
|---|---|---|---|---|
| No assessment needed. | | | | |

Figure 6. Report List



Stocks Flow Report

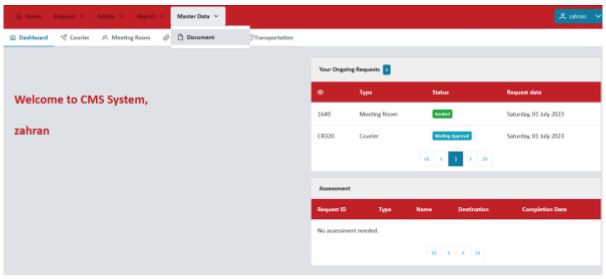
Location | Item type

From | To

Figure 7. Stock Flow Report Download

The Report section enables users to download documents related to stock flow, courier activities, and driver assessment. The UI provides a straightforward interface where users can select the desired report type and specify any relevant parameters. Upon selection, the system generates the report and offers a download option, allowing users to obtain the requested documents conveniently.

### 4.5 Master Data



Welcome to CMS System,

zahran

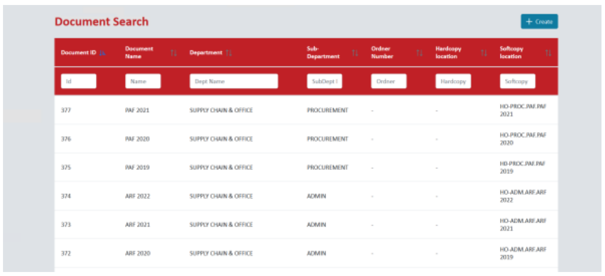
Your Ongoing Requests 2

| ID | Type | Status | Request date |
|---|---|---|---|
| 1640 | Meeting Room | Booked | Saturday, 01 July 2023 |
| CR320 | Courier | Waiting Approval | Saturday, 01 July 2023 |

Assessment

| Request ID | Type | Name | Destination | Completion Date |
|---|---|---|---|---|
| No assessment needed. | | | | |

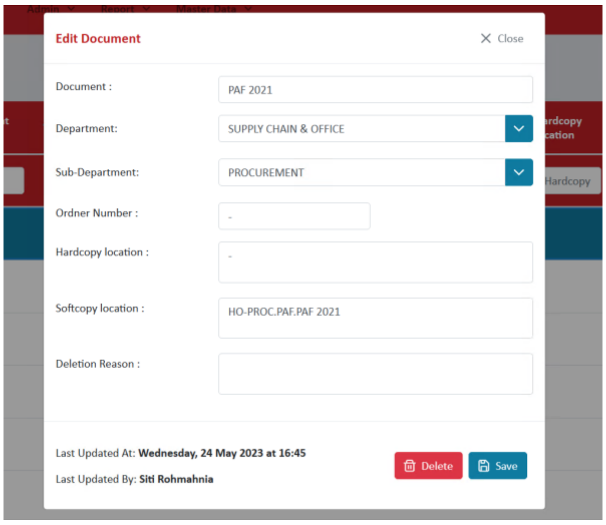
Figure 8. Master Data



Document Search

| Document ID | Document Name | Department | Sub-Department | Ordner Number | Hardcopy location | Softcopy location |
|---|---|---|---|---|---|---|
| Id | Name | Dept Name | SubDept I | Ordner | Hardcopy | Softcopy |
| 377 | PAF 2021 | SUPPLY CHAIN & OFFICE | PROCUREMENT | - | - | HO-PROC.PAF.PAF 2021 |
| 376 | PAF 2020 | SUPPLY CHAIN & OFFICE | PROCUREMENT | - | - | HO-PROC.PAF.PAF 2020 |
| 375 | PAF 2019 | SUPPLY CHAIN & OFFICE | PROCUREMENT | - | - | HO-PROC.PAF.PAF 2019 |
| 374 | ARF 2022 | SUPPLY CHAIN & OFFICE | ADMIN | - | - | HO-ADM.ARF.ARF 2022 |
| 373 | ARF 2021 | SUPPLY CHAIN & OFFICE | ADMIN | - | - | HO-ADM.ARF.ARF 2021 |
| 372 | ARF 2020 | SUPPLY CHAIN & OFFICE | ADMIN | - | - | HO-ADM.ARF.ARF 2019 |

Figure 9. Master Data List



Edit Document

| | |
|---|---|
| Document : | PAF 2021 |
| Department: | SUPPLY CHAIN & OFFICE |
| Sub-Department: | PROCUREMENT |
| Ordner Number : | - |
| Hardcopy location : | - |
| Softcopy location : | HO-PROC.PAF.PAF 2021 |
| Deletion Reason : | |

Last Updated At: **Wednesday, 24 May 2023 at 16:45**
Last Updated By: **Siti Rohmahnia**

Figure 10. Master Data CRUD

The Master Data section allows users to view, search, and perform CRUD (Create, Read, Update, Delete) operations. The UI displays the master data in a structured and easily searchable format, enabling users to locate specific records efficiently. Users can perform CRUD operations on the master data, such as creating new entries, updating existing records, reading detailed information, and deleting unwanted entries. The UI presents intuitive controls and forms for these actions, ensuring a seamless user experience.

The results showcase a visually appealing and functional UI for each of the mentioned paths in the CMS application. The UI design focuses on providing an intuitive and efficient user experience, enabling users to navigate through different functionalities and perform their desired actions with ease.

Please note that the specific visual representations and layout of the UI may vary based on the implementation and design choices made during the development of the CMS application.

## 5. CONCLUSION

In conclusion, the development of the content management system (CMS) application utilizing NPM, Vue.js, Node.js, Postgresql, and Strapi has resulted in a robust and user-friendly solution for efficient content management and data manipulation. The application's UI provides a seamless experience, enabling users to navigate through various paths, create requests, perform administrative tasks, generate reports, and manage master data effectively. The successful implementation of the CMS application highlights the importance of utilizing appropriate technologies and following a systematic development methodology.

### 5.1 Recommendations

Recommendations for future readers and developers include exploring further customization options to tailor the CMS application to specific organizational needs, implementing additional security measures to protect sensitive data, and enhancing the reporting functionalities to provide more comprehensive insights. Additionally, considering the evolving nature of technology, it is advisable to stay updated with the latest versions of the utilized technologies and frameworks, as they may offer new features and improvements. Furthermore, conducting user feedback sessions and usability testing can provide valuable insights for future iterations and enhancements of the CMS application, ensuring continuous improvement and user satisfaction.

## References

[1.] NPM Documentation: https://docs.npmjs.com/
[2.] Vue.js Documentation: https://v3.vuejs.org/guide/introduction.html
[3.] Node.js Documentation: https://nodejs.org/en/docs/
[4.] Postgresql Documentation: https://www.postgresql.org/docs/
[5.] Strapi Documentation: https://strapi.io/documentation/
[6.] Sommerville, I. (2016). Software Engineering. Pearson Education.
[7.] Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach. McGraw-Hill Education.
[8.] Agile Alliance. (n.d.). Agile Manifesto. Retrieved from https://agilemanifesto.org/
[9.] W3Schools. (n.d.). Vue.js Tutorial. Retrieved from https://www.w3schools.com/vue/