



Design and Development of a PHP-Based Raskin Data Processing System: A Case Study of Kutambaru Village, Munte District, Karo Regency

Jonatan Sitorus^a, Farida Gultom^b, Sastra Wandu Nduru^c, M. Syaifuddin^d

Department of Informatics Engineering, Faculty of Engineering, Universitas Efarina, Pematangsiantar, Indonesia
Sistem Informasi, Universitas Budi Darma

Article Info

Keywords:

Raskin
Social Assistance
Population Data
PHP
Waterfall

ABSTRACT

Raskin and social assistance data processing in Kutambaru Village, Munte District, Karo Regency, had relied on manual records that were vulnerable to human error, inconsistent entries, fragmented information, and slow report retrieval. This study aimed to design, implement, and functionally validate an information system developed in PHP that integrates population, household condition, demographic, and assistance data within an administrative environment. The system was developed using the waterfall model through requirements analysis, system design, implementation, testing, deployment preparation, and maintenance planning. Requirements were obtained through observation, document examination, and discussion with a prospective village office user, then translated into flowcharts, data flow and context diagrams, relational database structures, and interface modules. The resulting application provides administrator authentication; population, housing, and hamlet data entry; public program information and beneficiary searches; demographic summaries; an administrative dashboard; population and assistance classification; and printable reports. Functional validation was conducted directly with one prospective end user from the Kutambaru Village Office. Within this limited validation scope, the user confirmed that the documented functions operated in accordance with operational needs. The findings indicate that the application can centralize dispersed records, support more consistent data retrieval, and improve the organization of assistance administration. Nevertheless, the study does not establish targeting accuracy, usability across users, security robustness, or long term administrative impact. Future development should therefore formalize eligibility criteria, implement access controls and audit trails, expand structured testing scenarios, and evaluate processing time, data quality, user satisfaction, and beneficiary targeting outcomes after deployment.

This is an open access article under the CC BY-SA license.



Corresponding Author:

Jonatan Sitorus
Department of Informatics Engineering
Universitas Efarina
Pematangsiantar, Indonesia
Email: jonatansitorus1507@gmail.com

1. INTRODUCTION

Village governments depend on reliable population and social assistance information to perform administration and support decisions affecting households. Officers must identify residents, document family relationships and living conditions, prepare beneficiary lists, update demographic summaries, and communicate program information. When these activities rely on handwritten forms or separate files, information may be recorded in different formats. Fragmentation makes verification difficult because officers must compare sources to determine which record is current. Manual processing also increases the risk of incomplete fields, inconsistent

values, duplicate entries, and slow retrieval when reports are needed. These weaknesses can reduce the traceability of administrative decisions and complicate corrections. Structured data processing is therefore a foundation for accountable village services [1].

Kutambaru Village, Munte District, Karo Regency, faced administrative difficulties in managing Raskin and social assistance information. The existing process did not provide one application where population identity, household condition, hamlet, demographic, and assistance records could be entered and reviewed together. Consequently, officers had limited support for consolidating information originating from activities. Searching for a household, checking whether its data were complete, and preparing an updated summary could require repeated examination of separate records. This situation made it harder to maintain a consistent relationship between resident data and assistance information. A centralized application was therefore needed to organize the records while preserving the responsibility of village personnel to examine, confirm, and update the information they manage.

Raskin administration involves more than storing a recipient list. A record must connect a person or household with identity data, family context, residential location, living conditions, economic indicators, and the assistance category. These elements help officers review why a record appears in a beneficiary dataset and detect missing or inconsistent information. Accurate data are important because social assistance concerns limited public resources and affects economically vulnerable households. Previous discussion of Raskin implementation emphasizes that data quality, transparency, and administrative accuracy are central to appropriate beneficiary targeting [2]. Nevertheless, a computerized record cannot determine fairness by itself. It should present information consistently and support review, while eligibility criteria, verification procedures, and final decisions remain governed by authorized institutional policies.

A web based information system offers a way to integrate these administrative activities. Through an interface, one database can support entry, editing, classification, searching, summary presentation, and report preparation. Centralization reduces the need to copy values across unrelated files and allows changes to appear in searches and reports. Public pages can provide program information or beneficiary lookup functions, while protected pages restrict administrative operations to authorized officers. Research on web based administration shows that this architecture is relevant for services requiring structured records and accessible information [3]. PHP processes forms, sessions, validation logic, and server requests, while MySQL stores related population and assistance data in relational tables. Their combination is widely used for database applications and development [4].

Developing a system requires a method connecting administrative needs with design artifacts, program modules, and verification. If programming begins before the required data and workflows are understood, forms may omit essential fields or reports may not match office practice. The waterfall model provides a sequential framework in which requirements analysis is followed by system design, implementation, testing, deployment preparation, and maintenance planning [5]. This sequence documents how a requirement is translated into a component and checked. Flowcharts describe sequences, context and data flow diagrams clarify interactions and data movement, database structures define persistent relationships, and interface layouts show how users access functions [6]. Together, these artifacts create a traceable bridge between the administrative problem and the implemented application.

This study aims to design, implement, and functionally validate a PHP based Raskin data processing system for Kutambaru Village. The application supports administrator authentication, population and household condition entry, hamlet and demographic management, public assistance information, beneficiary searches, summaries, classification, and printable reports. By combining these functions, the system is intended to improve the organization and retrieval of records used in village assistance administration. The study is limited to system development and functional confirmation with a prospective village office user. It does not measure beneficiary targeting accuracy, processing time before and after deployment, usability across officers, or long term administrative impact. These boundaries ensure that the contribution remains consistent with evidence from the implementation and validation activities.

2. RESEARCH METHOD

2.1. Research setting and data collection

The study was conducted in Kutambaru Village, Munte District, Karo Regency, North Sumatra. Requirements were derived from the village's population and assistance-administration activities, including population identity, household condition, hamlet, demographic, and beneficiary information.

Observation, document examination, and discussions with the prospective user were used to identify the existing workflow. The design was documented through a login flowchart, data flow diagram, context diagram, database tables, and interface layouts before programming.

2.2. Waterfall procedure

The development procedure followed six connected waterfall stages: requirements analysis, system design, implementation, testing, deployment preparation, and maintenance. Each stage produced information needed by the next stage, allowing the application to be developed from documented village activities rather than from

assumptions about how Raskin administration should operate. Although the stages are described sequentially, clarification could still be made when a later activity revealed that an earlier requirement needed more precise interpretation.

The first stage was requirements analysis. Observation, document examination, and discussion with the prospective user were used to identify the information handled by the Kutambaru Village Office and the functions expected from the application. This analysis covered administrator access, population identity, household condition, hamlet and demographic data, assistance information, classification, public searches, summaries, and reports. The result of this stage was a defined functional scope that distinguished protected administrative activities from information intended for public access.

The second stage was system design. The identified requirements were translated into a login flowchart, context diagram, data flow diagram, database tables, and interface layouts. The process models described who interacted with the system and how information moved between users, application functions, and storage. The database design organized related records so that population, household, location, and assistance information could be retrieved consistently. Interface design then arranged the fields, navigation, actions, and output pages required by each documented process.

The third stage was implementation. The approved design artifacts were converted into PHP modules connected to a MySQL database [7]. Administrative modules handled authentication, data entry, management, classification, dashboard summaries, and reports, while public modules presented program information and beneficiary search access. Implementation also connected form inputs with the corresponding database operations and displayed stored values through tables, summaries, or printable outputs. At this stage, the focus was to ensure that each programmed page represented a requirement already identified during analysis and design.

The fourth stage was testing and functional verification. The implemented pages were opened and operated to determine whether their visible functions matched the documented needs. A prospective end user from the Kutambaru Village Office navigated the available features and confirmed the suitability of the main functions for the intended administrative setting. This activity provided initial evidence that the application could be used as designed. However, the available validation did not constitute extensive security, performance, stress, or multiuser usability testing, so the conclusions remain limited to the functional scope reported in the study.

The fifth stage concerned deployment preparation. After the principal functions had been verified, the system was positioned for use in the village environment, including preparation of the application, database, user access, and operational guidance needed for introduction to stakeholders. Deployment was treated as a controlled transition rather than proof of long term effectiveness. The final stage was maintenance planning. Corrections, data structure adjustments, security improvements, backup procedures, and later requirement changes were recognized as continuing responsibilities. Feedback after actual use should guide these activities so that the system remains aligned with administrative practice and changing program needs.

2.3. Requirements and implementation environment

The application separates protected administrative functions from public information and search pages. Table 1 summarizes the functions derived from the original thesis.

Table 1. Functional requirements of the Raskin information system

No.	Actor	Required Function	Expected Output
1	Administrator	Authenticate to the management area	Authorized dashboard access
2	Administrator	Enter population and household-condition data	Structured resident and condition records
3	Administrator	Maintain hamlet and demographic data	Updated territorial summaries
4	Administrator	Classify population and assistance	Reviewable beneficiary categories
5	Public user	Search social-assistance information	Relevant beneficiary information
6	System	Produce population and assistance reports	Printable administrative output

Tables 2 and 3 report the implementation environment. The application was built in Adobe Dreamweaver CS6, connected to MySQL, and tested on a Windows workstation.

Table 2. Software environment

No.	Software	Version or Purpose
1	Microsoft Windows	Windows 10 Pro
2	Adobe Dreamweaver	CS6; code authoring
3	MySQL / XAMPP	Version 3.3.0; database and local server

Table 3. Hardware environment

No.	Hardware	Specification
1	Laptop	Zyrex
2	Processor	AMD Ryzen 5 7430, 2.30 GHz
3	Memory	8 GB RAM
4	Storage	120 GB
5	Display	15-inch LCD

2.4. Verification approach

Functional validation was performed directly with one Kutambaru Village employee identified as a prospective end user. The user navigated the available functions and confirmed whether the pages and data-management features aligned with office requirements.

This validation provides evidence of functional acceptance but does not provide a detailed scenario-by-scenario test table in the original thesis. Table 4 therefore transparently consolidates the documented functions and the stated user confirmation; it does not claim unreported stress, security, or usability testing.

3. RESULTS AND DISCUSSION

3.1. Implemented user interfaces

The Chapter IV implementation contains input, public-output, dashboard, classification, and reporting pages. Six representative program screens are reproduced separately below, each with an explanation of its operational role.

Figure 1 shows the administrator login page. Registered credentials are required before the officer can access protected population, housing, classification, and reporting functions, separating administrative operations from the public website.

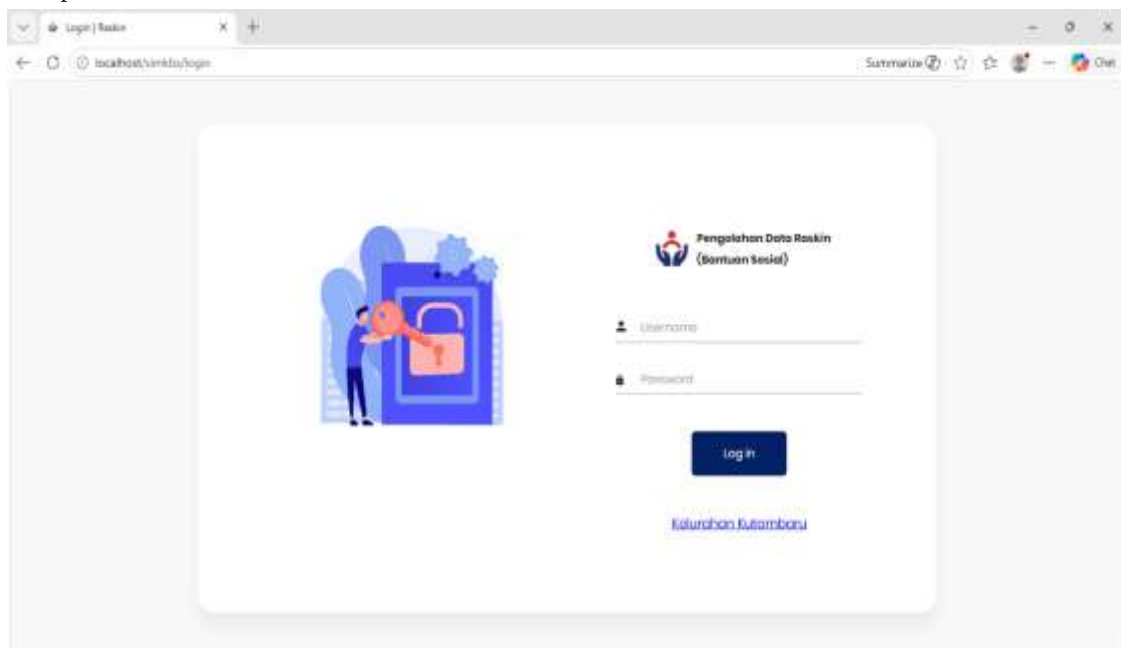


Figure 1. Administrator login page

Figure 2 presents the population data-entry interface. The form groups household number, NIK, personal identity, birth information, parents, location, religion, assistance, disability, education, work, and savings attributes so that beneficiary-related evidence can be stored in a consistent structure.

Figure 2. Population data-entry interface

Figure 3 displays the housing-condition entry page. The interface records the physical and socioeconomic indicators required to describe household circumstances, providing evidence that can complement identity data during beneficiary review.

Figure 3. Housing-condition data-entry interface

Figure 4 shows the public home page. It introduces the Raskin or social-assistance service and provides a direct action for checking beneficiary information, improving access to program communication without exposing administrative controls.



Figure 4. Public home page

Figure 5 presents the administrator dashboard. Summary cards and charts consolidate resident, household, hamlet, sex, education, and related demographic values, enabling officers to review the stored population profile from one management page.

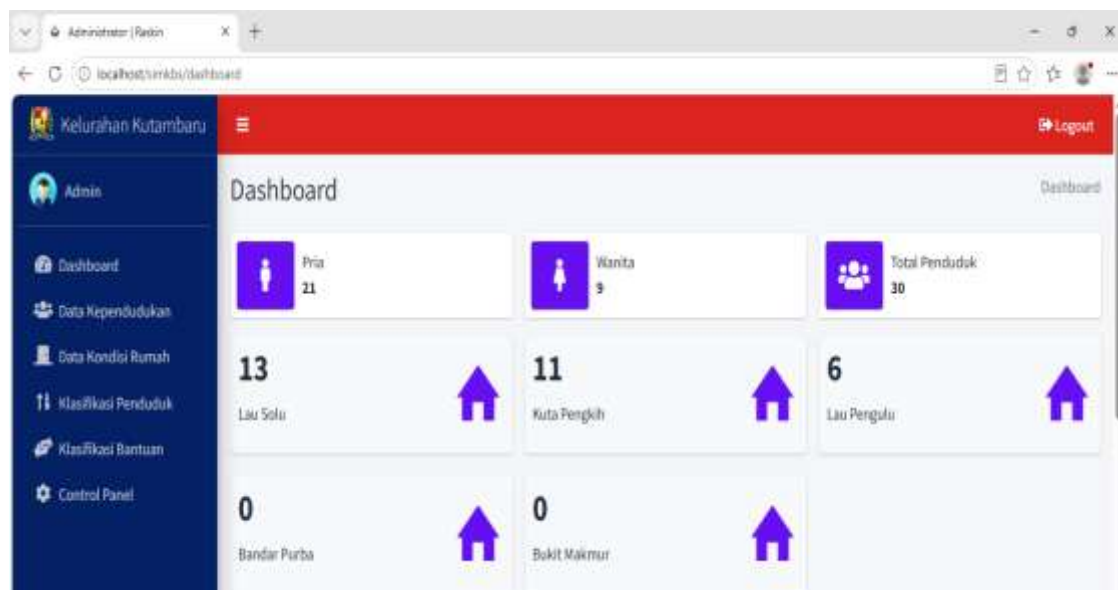


Figure 5. Administrator dashboard

Figure 6 shows the assistance-classification management page. The table organizes household identifiers and classification attributes and provides actions for reviewing records and generating assistance reports.

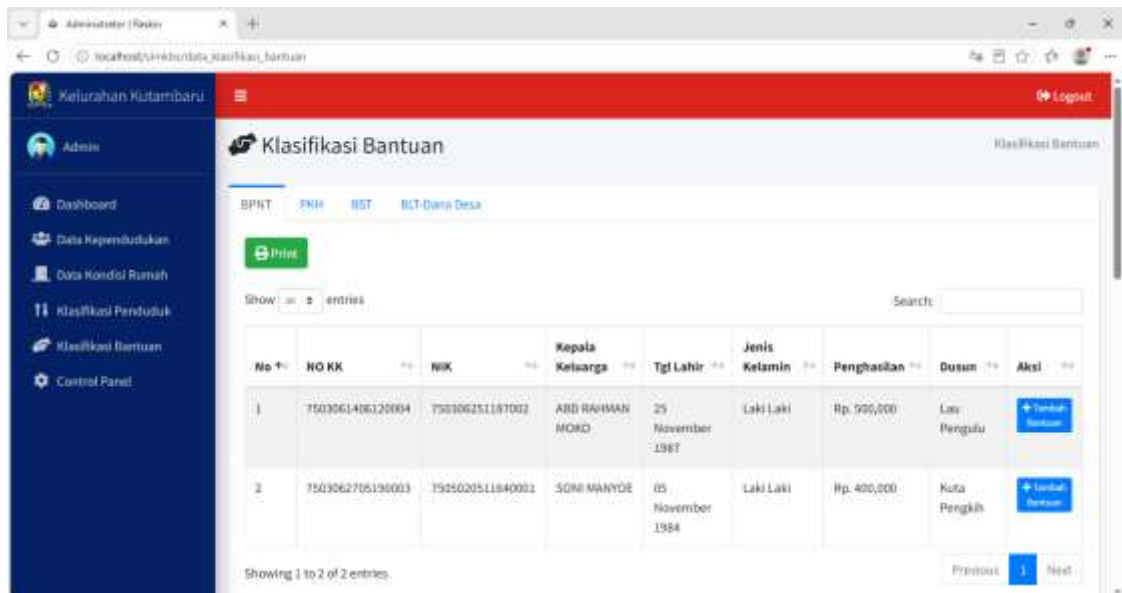


Figure 6. Assistance-classification management page

3.2. Functional verification results

The prospective user confirmed that the implemented functions operated properly and matched the village office's needs. Table 4 presents the six representative functional checks used to summarize that validation. All six were reported as conforming within the stated scope.

Table 4. Functional validation summary

No.	Test Scenario	Expected Result	Observed Result	Status
1	Submit valid administrator credentials	Administrative dashboard opens	Dashboard opened	Passed
2	Open population entry	Population form is displayed and accepts data	Form functioned as required	Passed
3	Open housing-condition entry	Housing-condition form is available	Form functioned as required	Passed
4	Open public beneficiary information	Public information and search page is displayed	Page displayed	Passed
5	Review dashboard summaries	Stored data are summarized visually	Summaries were displayed	Passed
6	Open assistance classification	Classification records and report actions are available	Functions were available	Passed

3.3. Discussion

The implementation connects data entry, public communication, administrative summaries, classification, and reports. This integration is more useful than digitizing an isolated beneficiary list because population and household context remain accessible during assistance administration.

The dashboard and classification pages also support review. However, a computerized classification page does not automatically make eligibility decisions objective. Explicit criteria, weights, validation rules, and an appeal process must be defined before the system is used as an automated decision-support instrument. Public-sector information systems should preserve transparency and accountability in benefit allocation [8].

The direct validation with a village employee is valuable because it tests relevance to the intended setting. Nevertheless, one prospective user and a general confirmation do not establish usability across all officers or complete functional coverage. Black-box testing should include invalid NIK values, duplicate households, missing fields, session expiry, report filters, database failure, and role restrictions [9].

Population and socioeconomic records are sensitive. Future implementation should apply password hashing, least-privilege authorization, audit logs, encrypted transport, backup procedures, and documented retention controls. Web application security standards provide a practical checklist for these controls [10].

A later field evaluation should compare processing time, error rates, report preparation, beneficiary-data completeness, and officer satisfaction before and after deployment. MySQL constraints and transaction handling

can improve data integrity [11], while maintainable PHP structure supports controlled system evolution [12]. Research on Raskin implementation reinforces the importance of accurate program administration [13]. Studies of benefit delivery also stress the relationship between village data and beneficiary fairness [14]. Web registration research provides relevant lessons for controlled service records [15]. Responsive interface work can guide later mobile-access improvements [16]. These measures would extend the present prototype into a more defensible administrative service.

4. CONCLUSION

A PHP- and MySQL-based Raskin data processing system was designed and implemented for Kutambaru Village using the waterfall model. The application integrates administrator authentication, population and housing-condition entry, hamlet and demographic information, public assistance searches, dashboards, classifications, and reports. A prospective village-office user confirmed that the documented functions operated according to operational needs, and six representative functional checks were summarized as passed. The system offers a structured alternative to manual data processing, but future work must define objective eligibility rules, expand testing, strengthen privacy and security, and measure administrative and targeting outcomes after deployment.

REFERENCES

- [1] M. S. Afifuddin Nur, "Pengolahan data," vol. 2, pp. 163-175, 2024.
- [2] M. N. Patama, "Analisis Program Raskin dalam meningkatkan kesejahteraan masyarakat," vol. 5, no. 1, pp. 8-17, 2017.
- [3] A. N. S. Supriady, "Sistem informasi administrasi berbasis web," Jurnal Teknik Informatika, vol. 14, no. 1, pp. 47-53, 2022.
- [4] F. Sinlae, I. Maulana, F. Setiyansyah, and M. Ihsan, "Pengenalan pemrograman web: Pembuatan aplikasi web sederhana dengan PHP dan MySQL," vol. 2, no. 2, pp. 68-82, 2024.
- [5] R. S. Pressman and B. R. Maxim, Software Engineering: A Practitioner's Approach, 9th ed. New York, NY, USA: McGraw-Hill, 2020.
- [6] I. Sommerville, Software Engineering, 10th ed. Boston, MA, USA: Pearson, 2016.
- [7] S. Bahri and P. P. Budi, "Pengembangan sistem informasi berbasis web," vol. 8, no. 3, 2020.
- [8] ugiman, "Pemerintahan desa," vol. 7, no. 1, pp. 82-95, 2018.
- [9] G. J. Myers, C. Sandler, and T. Badgett, The Art of Software Testing, 3rd ed. Hoboken, NJ, USA: Wiley, 2011.
- [10] OWASP Foundation, "OWASP Application Security Verification Standard 4.0.3," 2021.
- [11] Oracle Corporation, "MySQL Reference Manual," MySQL Documentation.
- [12] The PHP Group, "PHP Manual," PHP Documentation.
- [13] D. N. Aisyah, "Implementasi program beras miskin di Kelurahan Rowosari Kecamatan Tembalang."
- [14] G. Kongginawan, "Implementasi Program Beras Miskin bagi masyarakat berpenghasilan rendah di Kelurahan Sawang Bendar."
- [15] J. Dipati et al., "Sistem registrasi berbasis web," KOMPUTA, vol. 10, no. 2, pp. 70-77, 2021.
- [16] C. Perdana and M. A. Wijaya, "Implementasi Framework Bootstrap 5 pada perancangan front-end website," vol. 2, no. 1, pp. 30-43, 2024.