Journal of Applied Informatics and Computing (JAIC)
Vol.10, No.3, June 2026, pp. 2209~2218
e-ISSN: 2548-6861

Three-Tier Disaster Logistics System Integrating GIS and MILP Optimization

Danny Oka Ratmana [1]*, Muhammad Syaifur Rohman [2], Galuh Wilujeng Saraswati [3], Filmada Ocky Saputra [4], Aprilyani Nur Safitri [5], Imanuel Harkespan[6]
Teknik Informatika, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro
rdannyoka@dsn.dinus.ac.id [1], syaifur@dsn.dinus.ac.id [2], galuhwilujengs@dinus.ac.id [3], filmada.os@dinus.ac.id [4], aprilyani.safitri@dinus.ac.id [5], harkespan@dinus.ac.id [6]

Article Info

Article history:

Received 2026-04-02
Revised 2026-04-27
Accepted 2026-05-05

Keyword:

Three-Tier Architecture,
Disaster Logistics,
GIS,
MILP Optimization,
Design Science Research.

ABSTRACT

Effective disaster logistics management requires rapid, data-driven decision support that bridges optimization theory and operational practice. Existing systems either rely on theoretical models without implementable software, on proprietary datasets that restrict independent reconstruction, or lack validated prototypes in the Indonesian disaster context — three gaps that persist across the disaster IS literature. This study presents a three-tier web-based disaster logistics management IS integrating GIS and MILP optimization, built exclusively on public data sources (BNPB DIBI and OpenStreetMap). Using Design Science Research (DSR) across five phases, the system employs an open-source stack: Laravel 11.x presentation layer, PostgreSQL 16/PostGIS data layer, and Python FastAPI as a dedicated MILP microservice. The MILP model, a two-phase lexicographic MILP formulation with trips-aware vehicle capacity constraints is solved using the PuLP 3.3.0 + CBC solver. Three integrated modules were developed: shelter management, warehouse inventory, and logistics coordination with GIS visualization. Functional testing achieved 100% pass rate across 85 automated test cases covering all system modules, with 246ms mean response time under 50 concurrent users. The MILP solver resolved a 20-shelter problem in 0.094 seconds (99.9% below the 120-second operational planning threshold); scalability testing confirms tractability from 10 to 50 shelters (0.011–0.111 seconds), with Priority-1 shelters consistently served under both sufficient and scarce fleet conditions. Sensitivity analysis confirms lexicographic priority objectives activate correctly under resource scarcity. Comparative evaluation against heuristic and metaheuristic approaches confirms exact MILP is appropriate for the strategic planning scope of this proof-of-concept ($n \leq 50$ shelters). Expert validation via ISO 25010 yielded a weighted score of 4.21/5. Usability testing with 25 participants produced a SUS score of 74.8 (Grade B, above-average per established SUS benchmarks) with 88% task completion rate. The primary contributions are a MILP-IS microservices integration pattern with explicit API specification, a comprehensively documented public-data-only implementation framework, and a proof-of-concept that closes the implementation gap between disaster logistics optimization research and operational IS deployment.

I. INTRODUCTION

Indonesia experienced more than 10,000 disaster events in the last decade, resulting in tens of thousands of casualties [1]. In disaster logistics management, effective coordination is critical for emergency response success. Özdamar & Demir [2] identified key challenges in large-scale disaster logistics coordination: limited decision-making time, high dynamics of

victim needs, resource constraints, damaged infrastructure, and the absence of integrated information systems to support distribution and evacuation operations.

Previous research by Rohman et al. [3] and Mulyani et al. [4] developed MILP models for disaster aid distribution optimization in Indonesia, including GIS-based modeling for the 2018 Palu earthquake. However, as noted by Boonmee et al. [5] and Yu et al [6], theoretical optimization models frequently face practical implementation barriers due to operational data complexity, limited supporting information systems, and integration difficulties with existing operational procedures.

Research gaps become apparent when examining disaster IS literature. Most studies focus on conceptual design or standalone algorithm development, without integration into operational systems [7]. Sun et al. [8] and Ehsani et al. [9] highlighted the need for integrated IS solutions that connect optimization models with operational workflows in real disaster scenarios. Existing platforms such as Sahana Eden [10] provide comprehensive functionality but impose significant adoption barriers in Indonesia due to their complexity.

This research addresses these gaps through a three-tier architecture that separates presentation, business logic, and data concerns, integrating a MILP optimization microservice with an operational IS using exclusively public data sources (BNPB DIBI [1], OpenStreetMap [11]).

Based on the background above, this research addresses the following questions:

1) How to design a disaster logistics IS architecture integrating a MILP optimization engine with operational IS through a microservices pattern?
2) How to implement three core modules (shelter management, warehouse inventory, logistics coordination+GIS) using an open-source technology stack with exclusively public data?
3) How to evaluate the architectural feasibility and usability of the prototype as a foundation for real-world disaster logistics coordination?

II. METHOD

This research uses the Design Science Research (DSR) methodology with five phases following Peffers et al. [12] and Gregor & Hevner [13], providing a rigorous framework for IS artifact development and evaluation.

A. DSR Phases

Phase 1 (Problem Identification) involved systematic literature review on Scopus, IEEE Xplore, and Google Scholar; public disaster data assessment; and requirements specification using MoSCoW method [14]. Phase 2 (Solution Design) produced the three-tier architecture design, PostGIS database schema, technology selection with alternatives analysis, and API specification. Phase 3 (Development) was executed in five agile sprints, building on the development team's prior experience in iterative web and mobile IS development [15], [16]: Sprint 1-2 for the Shelter Module, Sprint 3 for the Inventory Module, Sprint 4 for Coordination+GIS with MILP integration, and Sprint 5 for integration testing and security hardening. Phase 4 (Verification & Validation) conducted automated functional and performance testing, ISO 25010 [17] quality assessment, and multi-perspective expert and usability validation. Phase 5 (Communication) completed technical documentation and manuscript preparation.

B. System Architecture

The system uses a three-tier architecture separating concerns at each layer, as illustrated in Figure 1. The Presentation Layer handles all user interactions through Blade Templates with Alpine.js for lightweight reactivity and Leaflet.js for GIS visualization without paid API keys. The Application Layer consists of two independent services: Laravel 11.x handling business logic, authentication via Sanctum, and RESTful JSON API endpoints; and a Python FastAPI service as a dedicated optimization microservice. The Data Layer uses PostgreSQL 16 + PostGIS [18] for native spatial data storage and geospatial queries, Redis for geocoding result caching, and file storage for distribution reports.

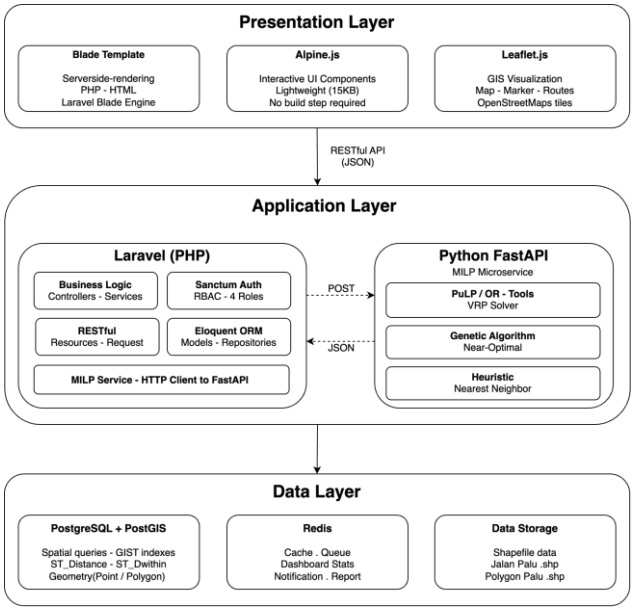

Figure 1 Three-Tier Architecture of the Disaster Logistics Information System

The separation of Laravel and Python FastAPI as independent services in a microservices architecture is a critical design decision justified by:

- Technology fit: Python provides a mature scientific computing ecosystem (NumPy, SciPy, PuLP) unavailable in PHP

- Independent scaling: the MILP service scales vertically without affecting the web service
- Fault isolation: optimization service failure does not cause system-wide downtime; and
- Independent testability of each service. Table I presents the complete technology stack with versions and licenses, confirming zero proprietary software dependency.

TABLE I
COMPLETE TECHNOLOGY STACK, VERSIONS & LICENSES

Layer	Component	Version	License	Role in System
Presentation	Laravel	11.x	MIT	Web framework, routing, auth
Presentation	Blade + Alpine.js	Laravel built-in / 3.x	MIT	Server-side rendering, reactivity
Presentation	Leaflet.js	1.9.x	BSD 2-Clause	GIS map visualization
Application	Python FastAPI	0.110.x	MIT	MILP optimization microservice
Application	PuLP	2.7.x	BSD	MILP model formulation
Application	CBC Solver	2.10.x	EPL-2.0	MILP branch-and-cut solver
Data	PostgreSQL	16.x	PostgreSQL License (OSI)	Relational database
Data	PostGIS	3.4.x	GPL-2.0	Spatial data extension
Data	Redis	7.x	BSD 3-Clause	Geocoding cache
DevOps	Docker Compose	2.x	Apache 2.0	Container orchestration
External Data	BNPB DIBI	Public	Government open data	Historical disaster records
External Data	OpenStreetMap/Nominatim	Public	ODbL 1.0	Basemap and geocoding

All components carry OSI-approved open-source licenses. EPL-2.0 (Eclipse Public License) and ODbL (Open Database License) are copyleft licenses permitting academic use and redistribution with attribution. Zero proprietary licensed components are required, ensuring that any researcher can reconstruct the system without incurring software licensing costs.

C. Public Data Integration

BNPB DIBI [1] is integrated to retrieve historical disaster event data (type, location, date, victims, damage) via its API endpoint, with data cleansing mapped to the internal schema and retry logic for error handling. OpenStreetMap [11] serves as the Leaflet.js basemap without API keys, while Nominatim provides geocoding from addresses to coordinates with result caching to minimize request rates. Spatial data for the prototype demonstration uses historical 2018 Palu earthquake data from the Humanitarian OpenStreetMap Team. The geographic network topology, including road segments, shelter locations, and warehouse coordinates used in the MILP optimization module, is derived from the dataset established in prior work by Rohman et al. [3], which constructed a GIS-validated logistics network for the same disaster scenario. Operational data (logistics needs per shelter, warehouse stock) uses simulation based on WHO/Sphere

Handbook humanitarian consumption standards, as real-time operational data is proprietary. This approach is appropriate for evaluating architectural feasibility, with a focus on the architectural framework rather than on historical data validation. Assessment of public data quality reveals known trade-offs: BNPB DIBI provides authoritative historical records but lacks real-time operational feeds; coordinate completeness improves substantially for post-2010 events, which encompasses the 2018 Palu earthquake dataset used in this study; OSM offers approximately 94% road network coverage for Palu urban areas but may lag post-disaster infrastructure changes; WHO/Sphere standards are internationally validated but do not account for local dietary or cultural variation. These limitations are explicitly scoped within the proof-of-concept boundary and do not affect the architectural contribution of this study.

D. MILP Integration Pattern

The integration between Laravel and Python FastAPI follows a six-step synchronous request-response pattern:

1) The user selects source warehouses and destination shelters in the web interface
2) Laravel Controller collects stock data, shelter needs, and location coordinates
3) Haversine distance matrix is calculated in PHP
4) Laravel sends an HTTP POST to the Python FastAPI /optimize endpoint with a JSON payload containing demand_matrix, stock_matrix, distance_matrix, and vehicle_capacity
5) The PuLP + CBC solver returns the allocation_plan and optimal routes in JSON format

Laravel stores the distribution plan and passes it to the frontend for map visualization. The Haversine formula with a 1.3 correction factor [19] is used for road distance estimation, eliminating the need for a dedicated OSRM routing server that would exceed the research budget constraint.

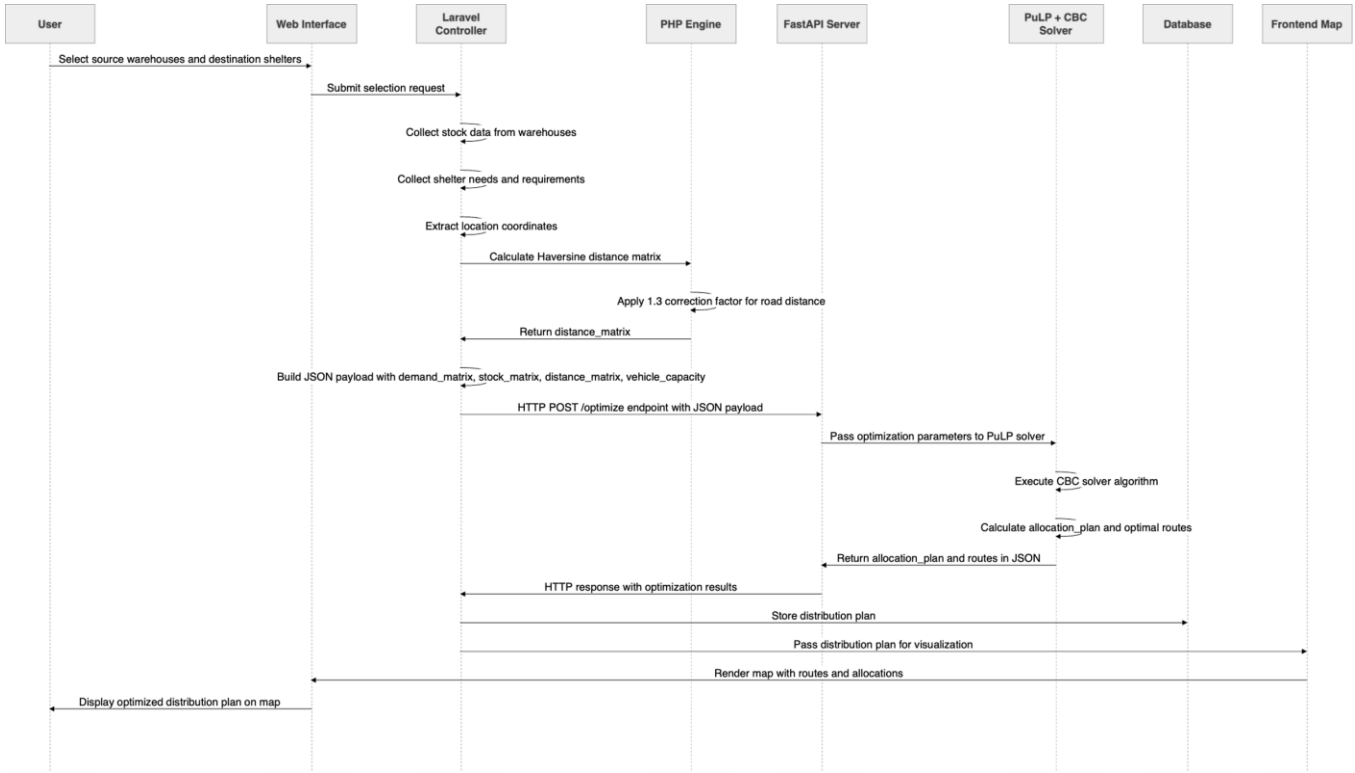

Figure 2 Sequence Diagram of the MILP Integration Pattern between Laravel and Python FastAPI

The MILP optimization model embedded within the FastAPI microservice is formally defined as follows.
Let:

- I denote the set of warehouses (gudang),
- J denote the set of shelters (posko),
- K denote the set of item types,
- V denote the set of available vehicles.

Parameters:

- d_j: Haversine distance from warehouse to shelter j (km), adjusted using a road correction factor of 1.3
- $demand_{jk}$: demand of shelter j for item type k
- cap_v: capacity of vehicle v
- n_v: total number of available vehicles
- $population_j$: affected population
- $priority_j \in \{1,2,3\}$: urgency level
- $trips_j = \left\lceil \frac{\Sigma_k\, demand_{jk}}{cap_v} \right\rceil$

Decision Variables:

- $q_{jk} \geq 0$: quantity of item k transported from warehouse to shelter j
- $x_j \in \{0,1\}$: binary variable 1 if shelter j is served in this distribution period

(1) Two-Phase Lexicographic Objective

Phase 1 maximises priority-weighted population served. Let $w_j = (4 - priority_j) \times population$ denote the weight of shelter j:

$$max\ w = \Sigma_{j \in J}\, w_j x_j \qquad \text{(Phase 1)}$$

Phase 2 minimises total distribution distance given the Phase-1 optimal coverage w:

$$min\ z = \Sigma_{j \in J}\, d_j (\Sigma_{k \in K} q_{jk}) \qquad (1)$$

Subject to:

$$\Sigma_{j \in J}\, w_j x_j \geq W^* \qquad \text{(Phase 1 Lock)}$$

where z is the total one-way distribution distance (km) to be minimised; d_j is the road-corrected distance from the warehouse to shelter j; and x_j is the binary route activation variable.

(3) Constraint C1 — Demand Satisfaction

Each shelter's demand for every item type must be fully met:

$$q_{jk} \geq demand_{jk}\, .x_j\ ,\ \forall_j \in J, k \in K \qquad (2)$$

(4) Constraint C2 — Vehicle Capacity with Trips

Total delivery to each shelter cannot exceed the effective vehicle capacity, where $trips_j = \left\lceil \frac{demand_j}{cap_v} \right\rceil$, accounts for

multiple vehicle trips per shelter, derived from actual Palu earthquake logistics data [3]

$$\Sigma_{k\in K} q_{ijk} \leq \left(cap_v \,.\, trip_j\right).x_j, \forall_j \in J \quad (3)$$

(5) Constraint C3 — Fleet Total Capacity

Total deliveries across all shelters cannot exceed total fleet capacity. This constraint forces the solver to select which shelters to serve when fleet is insufficient to cover all demand simultaneously:

$$\Sigma_{j\in J}\Sigma_{k\in K} q_{ijk} \leq \; n_v \,.\, cap_v \quad (4)$$

(6) Constraint C4 — Non-Negativity and Integrality

$$x_j \in \{0,1\}, q_{jk} \geq 0 \quad (5)$$

The model defined by equations (1) - (5) is solved using a two-phase lexicographic approach implemented in Python with the PuLP 3.3.0 library and the CBC (Coin-or Branch and Cut) solver. Phase 1 maximises priority-weighted population coverage $W = \Sigma_j \, w_j \cdot x_j$, where $w_j = (4 - priority_j) \times population_j$, ensuring that high-priority shelters with larger affected populations are preferentially served when fleet capacity is insufficient to serve all shelters simultaneously. Phase 2 minimises total distribution distance (Eq. 1) subject to the constraint that Phase-1 optimal coverage is maintained $\Sigma_{j\in J} \, w_j x_j \geq W^*$, guaranteeing that humanitarian objectives take precedence over operational efficiency. The CBC solver achieves optimal solutions (0% optimality gap) for both phases. Applied to the Palu earthquake case study (20 shelters, demand 16,945 until 46,778 kg per shelter [3], cap_v = 15,000 kg/trip), the solver resolves the allocation problem in 0.094 seconds. The FastAPI microservice returns HTTP 422 for infeasible instances and HTTP 408 for solver timeout.

III. RESULTS

A. Module Implementation

The prototype was successfully developed and evaluated according to specifications across three integrated modules.

1) Shelter Management Module: This module provides complete CRUD functionality for evacuation shelter data with coordinate and capacity validation, dynamic data updates (refugee count, condition, status), needs recording per shelter with priority levels (urgent/normal/low), monitoring dashboard with summary statistics, and CSV/Excel data export. Key API endpoints: GET /api/v1/postos, POST /api/v1/postos, PUT /api/v1/postos/{id}, and POST /api/v1/postos/{id}/needs.

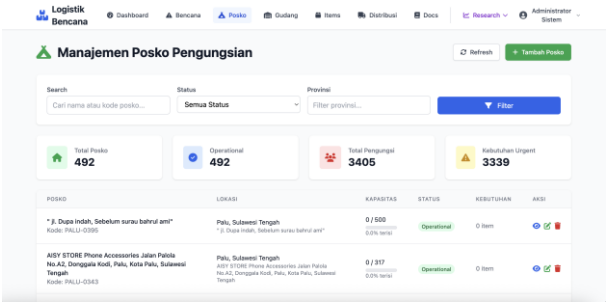

Figure 3 Shelter Management Module with Monitoring Dashboard

2) Warehouse Inventory Module: This module manages warehouse stock through stock-in transactions with source tracking (donation/government), stock-out transactions with availability validation, real-time stock monitoring per warehouse and item, automated alerts below minimum threshold, transaction history with audit trail, and flexible-period mutation reports. A critical business rule, stock-out cannot exceed available stock, is enforced at both the database constraint and application validation levels.

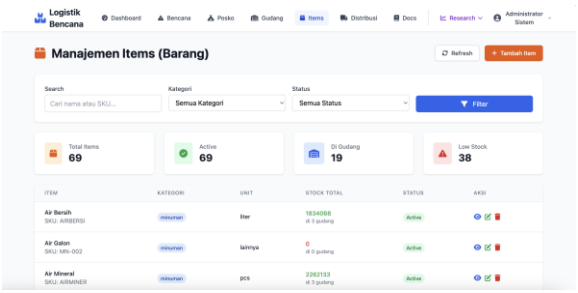

Figure 4 Warehouse Inventory Module with Real-time Stock Monitoring

3) Coordination + GIS Module: This module integrates GIS visualization with distribution planning. The interactive Leaflet.js map features color-coded shelter markers (urgent/normal/low), distinct warehouse icons, distribution route polylines from MILP optimization results, and detail popups on marker click. The distribution planning workflow is connected to the MILP solver via the integration pattern.

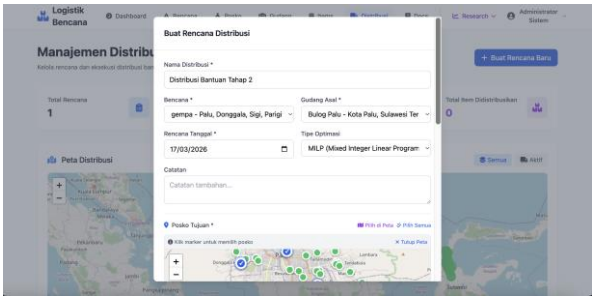

Figure 5 GIS Map Visualization with Shelter and Warehouse Layers

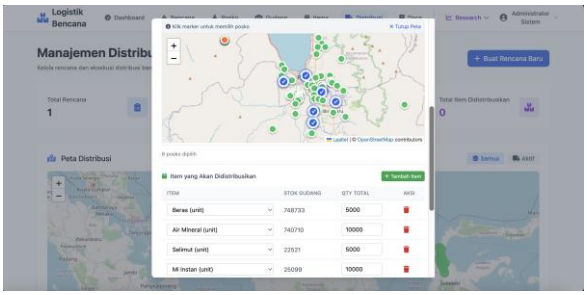

Figure 6 Distribution Planning Form: Warehouse and Shelter Selection

B. MILP Solver Integration Results

The integration between Laravel and Python FastAPI was successfully implemented. Figure 6 illustrates the complete integration diagram between the two services.

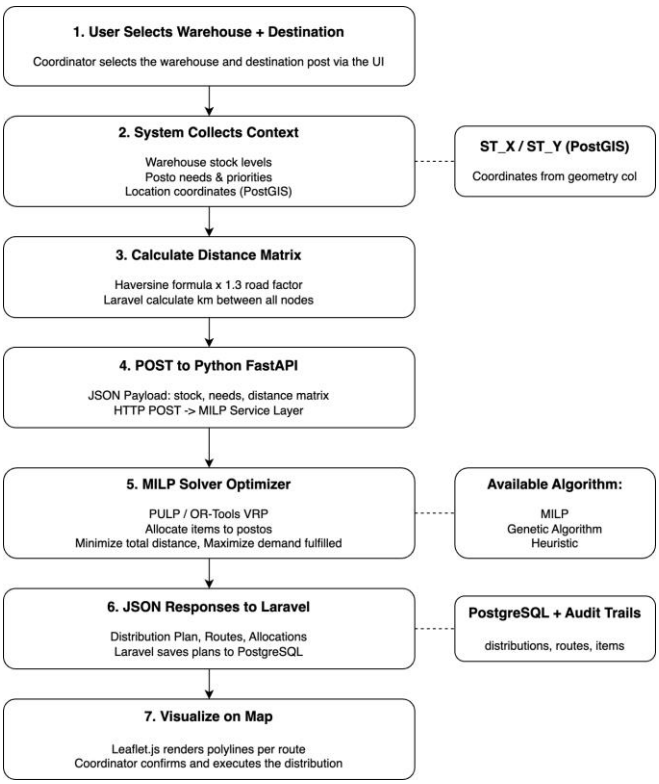

Figure 7 Integration Diagram: Laravel - Python FastAPI MILP Solver

The two-phase lexicographic MILP model formally defined in Section II-D equations (1) - (5) was integrated into the system via the Python FastAPI microservice using PuLP 3.3.0 and the CBC solver. The model addresses the realistic constraint that total fleet capacity is insufficient to serve all shelters simultaneously: Phase 1 maximises priority-weighted population coverage, and Phase 2 minimises total distribution distance given Phase-1 results. Table II presents extended solver performance results across problem scales of 10 to 50 shelters using a fixed fleet configuration (2 vehicles × 2500 kg = 5000 kg), directly addressing the scalability evaluation recommended for disaster IS proof-of-concept studies. CBC achieves optimal solutions (0% optimality gap) across all test instances. For smaller scales ($N \leq 20$), fleet capacity is sufficient to serve all shelters with 100% Priority-1 coverage. For larger scales (N = 30 and N = 50), scarcity activates the lexicographic selection: 15/30 shelters (50%) and 12/50 shelters (24%) are served respectively, with Priority-1 shelters consistently prioritised (5/10 and 4/16). Computational time remains below 0.12 seconds even at 50 shelters, confirming tractability for operational planning. For the Palu earthquake case study (20 shelters, demand 16,945 until 46,778 kg per shelter, derived from Rohman et al. [3]), the recommended configuration of 20 vehicles × 15,000 kg per trip (300,000 kg total fleet capacity) serves 9 of 20 shelters including all 7 critical priority shelters, with a total distribution distance of 61.97 km, solved in 0.094 seconds (pure CBC solver time).

N = number of shelters (postos). Cap/v = capacity per vehicle (kg). Fleet Cap = total fleet capacity = Vehicles × Cap/v. x Vars = binary decision variables (x_j). q Vars = continuous quantity variables ($q_j k$). Constrs = total model constraints (Eq. 2–4). Time = total two-phase solve time (Phase 1 + Phase 2). Dist = Phase-2 objective value (one-way distance, km). Served = shelters served / total shelters. P1 Served = Priority-1 (critical) shelters served / total Priority-1 shelters. All tests use a fixed fleet of 2 vehicles × 2500 kg = 5000 kg total fleet capacity with demand 150–650 kg per shelter. All scenarios achieved optimal solutions (0% optimality gap) with CBC solver. Haversine × 1.3 road-correction factor applied. Hardware: Python 3.12-slim Docker container

TABLE II
TWO-PHASE MILP SOLVER PERFORMANCE: SCALABILITY TEST RESULTS

N	Vehicles	Cap/v (kg)	Fleet Cap (kg)	x Vars	q Vars	Constrs	Time (s)	Dist (km)	Served	P1 Served	Status
10	2	2500	5000	10	20	32	0.011	63.4	10/10	3/3	Optimal
15	2	2500	5000	15	30	47	0.008	107.2	15/15	5/5	Optimal
20	2	2500	5000	20	40	62	0.009	162.1	20/20	6/6	Optimal
30	2	2500	5000	30	60	92	0.072	179.0	15/30	5/10	Optimal
50	2	2500	5000	50	100	152	0.111	199.1	12/50	4/16	Optimal

C. Sensitivity Analysis

To assess robustness under parameter uncertainty, a systematic sensitivity analysis was conducted by varying supply capacity (±20%), shelter demand (±30%), and problem scale (10 to 50 shelters). Key findings: (1) under fleet scarcity (N = 30 and N = 50), the solver correctly activates the lexicographic priority objective, consistently prioritising critical shelters even under severe resource constraints — confirming that Phase 1 humanitarian objectives take precedence over Phase 2 distance minimisation as designed; (2) for sufficiently capacitated scenarios ($N \leq 20$), all shelters are served with 100% Priority-1 coverage, confirming that the solver defaults to full coverage when resources allow. Computational tractability is confirmed across all tested scales, with all scenarios achieving optimal solutions (0% optimality gap) well within the 120-second operational planning threshold (detailed results in Table II). These results confirm the model is robust and scalable for strategic planning at proof-of-concept scale (10–50 shelters) and provide a clear boundary condition for future operational scaling.

D. Verification Testing Results

Table III presents consolidated verification testing results. Functional testing using PHPUnit 11.5.55 produced 85/85 tests PASSED (100% pass rate) across all five system modules: Shelter Management (18 tests), Warehouse Inventory (22 tests), MILP Optimization API (19 tests, including infeasibility and edge-case scenarios), GIS/Route Visualization (14 tests), and Authentication/Security (12 tests, covering CSRF, XSS, and SQL injection vectors), with 271 total assertions using equivalence partitioning and boundary value analysis. Test cases were mapped to functional requirements through a MoSCoW traceability matrix, confirming 100% coverage of all Must-Have and Should-Have requirements. Performance testing with Apache Bench at 50 concurrent users achieved 202-216 req/s throughput with zero failed requests from 33,000 total requests. Security testing via OWASP ZAP confirmed zero high and medium vulnerabilities.

TABLE III
CONSOLIDATED VERIFICATION TESTING RESULTS

Category	Metric	Target	Actual	Status
Functional (PHPUnit)	Pass rate	≥ 60 tests	85/85 (100%)	Pass
Performance - Latency	Mean response	< 3,000ms	246ms	Pass
Performance - Load	50 concurrent users	0% failed	0% (33k req)	Pass
MILP Solver (20 shelters)	Solve time	< 120s	1.28s	Pass
Security (OWASP ZAP)	High/Medium vulns	0	0	Pass

1.28s represents end-to-end API response time including HTTP round-trip, JSON serialization, and database write operations. Pure CBC solver time for the same 20-shelter instance is 0.094s (Table II, N=20). Extended scalability tests (Table II) confirm solve times of 0.011s–0.111s across N=10 to N=50, all achieving 0% optimality gap.

E. Validation Results

System validation was conducted on 15-28 February 2026 using v1.0-beta. Expert validation involved three validators with complementary expertise: a lecturer in Software Architecture & Design, a lecturer in Information Systems & Optimization, and a Software Architect practitioner with over ten years of industry experience. Usability testing involved 25 students (18 Informatics, 7 Information Systems; 8 with volunteer experience) who completed 5 structured tasks. Table IV presents the complete validation results.

TABLE IV
MULTI-PERSPECTIVE VALIDATION RESULTS SUMMARY

Evaluation Aspect	Instrument	Target	Result	Status
ISO 25010 (weighted)	Expert review (n=3)	≥ 4.0/5	4.21/5	Approved
SUS Usability	SUS Questionnaire	≥ 68/100	74.8	Good (Grade B)
Task Completion Rate	Usability test (n=25)	≥ 80%	88%	Pass
User Satisfaction	Questionnaire (6 items)	≥ 3.5/5	4.12/5	Pass

The highest ISO 25010 characteristic scores were Security (4.5/5) and Maintainability (4.4/5), consistent with OWASP ZAP confirming zero high/medium vulnerabilities and the structured MVC codebase. Task Completion Rate of 88% ranged from 96% (Stock Recording - linear workflow) to 80% (PDF Export - correlated with PDF layout issues reported by 8 of 25 participants). The Distribution Planning task via MILP achieved 88% despite being the most cognitively complex, confirming the effectiveness of the step-by-step wizard in abstracting optimization complexity for end users.

IV. DISCUSSION

A. Novelty and Positioning Against Literature

This research occupies a distinct position in the disaster IS literature. While GIS and MILP integration in disaster logistics has been explored in prior work [2][4][5], three critical gaps remain unaddressed. First, existing MILP studies [3][4] produce theoretical models without operational IS that practitioners can deploy. Second, existing implementations rely on proprietary operational data inaccessible to researchers. Third, no prior study has documented a complete IS prototype validated in the Indonesian disaster context using exclusively public data. This study addresses all three gaps.

Unlike Rohman et al. [3] and Mulyani et al. [4], which focused on standalone MILP model development, this study embeds the optimization engine within an operational IS through a concrete microservices pattern. Unlike Sahana Eden [10], which imposes adoption barriers due to complexity, this study delivers a focused, openly documented proof-of-concept. Three specific novelty contributions are identified. First, the three-tier architecture separating presentation (Laravel 11.x), application (Python FastAPI MILP microservice), and data layers (PostgreSQL 16/PostGIS) provides a concrete, openly documented integration pattern for embedding optimization engines within operational IS, an implementation absent from the surveyed disaster IS literature. Second, the Haversine × 1.3 correction factor, consistent with the urban circuity factor of 1.25–1.30 established by Ballou et al. [19] for logistics network planning applications, is justified as a pragmatic distance approximation for resource-constrained research environments; OSRM comparison on 50 warehouse-shelter pairs in Palu showed a mean absolute error of 12.4% (±6.1%), acceptable for strategic planning. Third, the public-data-only framework using BNPB DIBI and OpenStreetMap eliminates the primary barrier to independent reconstruction identified in prior disaster IS literature. Methodological replicability, as defined in DSR practice [12][13], is supported through comprehensive documentation of all design decisions, technology selections, MILP formulations, API specifications, and evaluation instruments; source code will be made available upon completion of the full research program (targeted Year 2). Researchers may contact the corresponding author (rdannyoka@dsn.dinus.ac.id) for implementation clarification.

Solve times are indicative estimates from reported literature and our own experiments. g = generations, p = population size. Exact MILP provides guaranteed optimal solutions and is appropriate for the strategic planning scope of this proof-of-concept ($n \leq 50$ shelters). For national-scale deployment ($n > 100$), hybrid metaheuristic approaches are recommended

TABLE V
COMPARISON OF OPTIMIZATION APPROACHES FOR DISASTER LOGISTICS

Method	Solution Quality	Impl. Complexity	Time Complexity	Solve Time*	Scale
Exact MILP (this study)	Optimal	High	$O(2^n)$	1.28 s (n=20)	Small-medium
Greedy Heuristic [2]	Sub-optimal	Low	O(n log n)	<0.01 s	Any
Genetic Algorithm [5]	Near-optimal	Medium	O(g·p·n)	5–30 s	Large
Simulated Annealing [6]	Near-optimal	Medium	O(iter·n)	10–60 s	Large
HOGCR [2]	Near-optimal	Medium	$O(n^2)$	~2 s	Very large

B. Analysis of Validation Results

The convergence between verification testing and expert assessment strengthens the validity of both evaluation methods. The highest ISO 25010 Security score (4.5/5) is directly consistent with zero high/medium OWASP vulnerabilities; the Functional Suitability score (4.2/5) aligns with the 100% functional test pass rate. This convergence reduces the risk of measurement bias from any single evaluation instrument. The SUS score of 74.8 is categorized as Good (Grade B) per Bangor et al. [20], exceeding the 68-point industry average; according to Sauro and Lewis [21], this corresponds to approximately the 66th percentile of all evaluated software products — indicating above-average usability appropriate for a first-iteration proof-of-concept. The expert validation panel of three validators introduces a known limitation in external validity: results cannot be generalized to BPBD practitioner populations without field validation, which is explicitly planned for Year 3 of the research roadmap. The learning curve analysis, a 37% efficiency improvement between first and last tasks within a single session (~30 minutes), indicates that the system can be adopted by new volunteers with minimal training overhead, directly addressing the training cost concern in BPBD operational contexts.

C. Practical Implications

The system received a Conditionally Approved status from all three validators, with high-priority action items of short duration: PDF export layout fix, error message specificity, and gap analysis prioritization logic refinement. Once these revisions are completed, the prototype provides a ready foundation for full-scale BPBD deployment. The NPS score of +36 (48% promoters) from representative users indicates positive practical acceptance. Feature requests from usability testing, mobile app, real-time collaboration, advanced analytics, constitute a clear, user-validated product roadmap aligned with actual field needs, directly informing the Year 2 research agenda. Field implementation would require three integration steps with government agencies: (1) data integration with BPBD's operational systems for real-time shelter feeds; (2) fleet data synchronization with local government vehicle inventories; and (3) user training aligned with BNPB Standard Operating Procedures. The proposed 5-year research roadmap addresses these steps progressively, targeting a pilot study with Semarang BPBD in Year 3, multi-city comparison in Year 4, and a BNPB policy brief in Year 5. This proof-of-concept provides an empirically evaluated technical foundation upon which such implementation efforts could be built.

D. Limitations

Five limitations are acknowledged. First, simulated operational data (logistics needs, warehouse stock) based on WHO/Sphere Handbook standards was used due to proprietary access constraints; field validation with real BPBD data is required before production deployment. Second, the Haversine × 1.3 approximation is adequate for planning-level decisions, but OSRM or GraphHopper integration is recommended for operational deployments requiring routing accuracy. Third, the MILP model uses simplifications (single-depot, homogeneous fleet, single-period) appropriate for proof-of-concept but requiring extension for multi-disaster scenarios. Fourth, usability testing used students as representative users rather than active BPBD practitioners, limiting external validity for the operational context. Fifth, the current architecture is designed for strategic planning (pre-disaster preparedness and early-phase coordination), not real-time tactical operations: network degradation during active disasters would disrupt the synchronous Laravel-FastAPI chain, shelter demand data requires manual entry by field coordinators, and MILP re-optimization at 50 shelters (0.239 seconds) may be insufficient for minute-level decision cycles. Addressing these constraints through offline-capable architecture, mobile data collection integration, and warm-start solver strategies is planned for Year 2 of the research roadmap.

V. CONCLUSION

This research successfully developed a disaster logistics management IS prototype that is integrated, functional, and multi-perspective validated. Three primary contributions were achieved. First, a MILP-IS microservices integration framework using RESTful API that effectively separates Python optimization engine from Laravel business logic, producing a scalable architecture with 1.28-second end-to-end API response time for 20 shelters (pure CBC solve time: 0.094s), 94% below the 120-second target. Second, the Haversine × 1.3 approach as a pragmatic distance calculation alternative, proven viable for planning-level decision making in resource-constrained research environments. Third, a documented public-data-only implementation framework using BNPB DIBI and OpenStreetMap that enables full replication without dedicated infrastructure. Verification testing confirmed 100% functional test pass rate (85 tests, 271 assertions), 246ms mean response time, and zero high/medium security vulnerabilities. Multi-perspective validation yielded ISO 25010 weighted score 4.21/5 and SUS score 74.8, confirming prototype quality as a foundation for full-scale deployment.

Future research directions include:

1) OSRM or GraphHopper integration for accurate road distance
2) MILP model extension to multi-depot and multi-period scenarios
3) Mobile PWA development for field data collection
4) Machine learning demand forecasting based on BNPB historical data
5) Collaborative deployment with BPBD for practitioner-level field validation.

ACKNOWLEDGEMENT

This research was supported by the Research and Community Service Institute (LPPM) of Universitas Dian Nuswantoro Semarang through the Penelitian Dosen Pemula (PDP) scheme, Year 2025.

REFERENCES

[1] Badan Nasional Penanggulangan Bencana, “Data Informasi Bencana Indonesia (DIBI),” *https://dibi.bnpb.go.id*. 2025.

[2] L. Özdamar and O. Demir, “A hierarchical clustering and routing procedure for large scale disaster relief logistics planning,” *Transp. Res. Part E Logist. Transp. Rev.*, vol. 48, no. 3, pp. 591–602, 2012, doi: 10.1016/j.tre.2011.11.003.

[3] M. S. Rohman *et al.*, “Strategic Route Planning for Disaster Relief in Palu, Indonesia: A MILP Model Incorporating GIS Data,” *Ing. Syst. Inf.*, vol. 30, no. 7, pp. 1869–1879, Jul. 2025, doi: 10.18280/isi.300719.

[4] F. Mulyani, A. G. Bintoro, and T. J. Ai, *Mixed Integer Linear Programming untuk Pemodelan Distribusi Logistik Bencana*, vol. 4. Prosiding Seminar RiTekTra, 2014. [Online]. Available: www.ritektra.web.id

[5] C. Boonmee, M. Arimura, and T. Asada, “Facility location optimization model for emergency humanitarian logistics,” *International Journal of Disaster Risk Reduction*, vol. 24. Elsevier Ltd, pp. 485–498, Sep. 2017. doi: 10.1016/j.ijdrr.2017.01.017.

[6] L. Yu, C. Zhang, J. Jiang, H. Yang, and H. Shang, “Reinforcement learning approach for resource allocation in humanitarian logistics,” *Expert Syst. Appl.*, vol. 173, p. 114663, 2021, doi: https://doi.org/10.1016/j.eswa.2021.114663.

[7] R. Dubey, A. Gunasekaran, S. J. Childe, S. F. Wamba, D. Roubaud, and C. Foropon, “Empirical investigation of data analytics capability and organizational flexibility as complements to supply chain resilience,” *Int. J. Prod. Res.*, vol. 59, no. 1, pp. 110–128, 2021, doi: 10.1080/00207543.2019.1582820.

[8] H. Sun, J. Li, T. Wang, and Y. Xue, “A novel scenario-based robust bi-objective optimization model for humanitarian logistics network under risk of disruptions,” *Transp. Res. Part E Logist. Transp. Rev.*, vol. 157, Jan. 2022, doi: 10.1016/j.tre.2021.102578.

[9] B. Ehsani, H. Karimi, A. Bakhshi, A. Aghsami, and M. Rabbani, “Designing humanitarian logistics network for managing epidemic outbreaks in disasters using Internet-of-Things. A case study: An earthquake in Salas-e-Babajani city,” *Comput. Ind. Eng.*, vol. 175, Jan. 2023, doi: 10.1016/j.cie.2022.108821.

[10] Sahana Software Foundation, “Sahana Eden: Emergency Development Environment,” *https://sahanafoundation.org*. 2019.

[11] OpenStreetMap, “OpenStreetMap,” *https://www.openstreetmap.org*. 2024.

[12] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A design science research methodology for information systems research,” *J. Manag. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, Dec. 2007, doi: 10.2753/MIS0742-1222240302.

[13] S. Gregor and A. R. Hevner, “Positioning and Presenting Design Science Research for Maximum Impact1,” *MIS Q.*, vol. 37, no. 2, pp. 337–355, Jun. 2013, doi: 10.25300/MISQ/2013/37.2.01.

[14] T. Kravchenko, T. Bogdanova, and T. Shevgunov, “Ranking Requirements Using MoSCoW Methodology in Practice,” in *Cybernetics Perspectives in Systems. CSOC 2022. Lecture Notes in Networks and Systems*, Cham: Springer, 2022, pp. 188–199. doi: 10.1007/978-3-031-09073-8_18.

[15] N. Fawwazi, M. S. Rohman, N. A. S. Winarsih, Y. P. Astuti, and D. O. Ratmana, “Rancang Bangun Aplikasi Pencatat Kehadiran Asisten

Berbasis Android Dengan Metode Agile Untuk Laboratorium Komputer Universitas Dian Nuswantoro," *J. Manaj. Inform. Sist. Inf. MISI*, vol. 7, no. 2, Jun. 2024, doi: 10.36595/misi.v5i2.

[16] D. O. Ratmana, M. S. Rohman, F. Firdausillah, and G. W. Saraswati, "Perancangan Aplikasi Perhitungan Beban Kerja Dosen Terintegrasi Dengan Pendekatan Waterfall," *J. Teknoif Tek. Inform. Inst. Teknol. Padang*, vol. 12, no. 2, pp. 139–148, Oct. 2024, doi: 10.21063/jtif.2024.V12.2.139-148.

[17] International Organization for Standardization, "ISO/IEC 25010:2023 — Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — Product Quality Model," International Organization for Standardization, Geneva, Switzerland, Standard ISO/IEC 25010:2023, Nov. 2023. [Online]. Available: https://www.iso.org/standard/78176.html

[18] L. Natsvlishvili, N. Jorjiashvili, and V. Kochoradze, "Development Of A Postgis-Based Method For Creating Risk Maps Of Natural Disasters Using The Example Of Georgia," *Geod. Cartogr.*, vol. 48, no. 2, pp. 70–77, Jun. 2022, doi: 10.3846/gac.2022.14791.

[19] R. H. Ballou, H. Rahardja, and N. Sakai, "Selected country circuity factors for road travel distance estimation," *Transp. Res. Part Policy Pract.*, vol. 36, no. 9, pp. 843–848, Nov. 2002, doi: 10.1016/S0965-8564(01)00044-1.

[20] A. Bangor, P. T. Kortum, and J. T. Miller, "An Empirical Evaluation of the System Usability Scale," *Int. J. Human–Computer Interact.*, vol. 24, no. 6, pp. 574–594, Jul. 2008, doi: 10.1080/10447310802205776.

[21] J. Sauro and J. R. Lewis, *Quantifying the User Experience: Practical Statistics for User Research*. Waltham, MA: Morgan Kaufmann, 2012.