IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 18, April 2026
Impact Factor: 8.2

A Data-Driven Hospital Resource and Inventory Management System for Improved Operational Efficiency

Rakshit Mehta, Aman Jalania, Parth Ghumatkar, Swami Patil, Prof. Shrikant Dhage
MIT ADT University, Pune, Maharashtra , India
rakshitm083@gmail.com, amanjalania2705@gmail.com, ompatil131102@gmail.com,
parthghumatkar@gmail.com, shrikant.dhage@mituniversity.edu.in

Abstract: *Hospitals generate large volumes of heterogeneous operational data—patient admissions, medication dispensation, staff schedules, bed occupancy records, and billing transactions. Despite the availability of data, most healthcare facilities (particularly small and medium-sized institutions) lack analytical capabilities to convert raw records into actionable insights. This results in systemic inefficiencies such as medicine overstocking and expiry, uneven staff workloads, and suboptimal bed utilization. In this research, we present a comprehensive, data-driven Hospital Resource and Inventory Management System (DRIMS) that centralizes operational data and provides descriptive and diagnostic analytics to support administrative decision-making.*

The proposed system integrates transactional databases, backend processing, scheduled analytical pipelines, and interactive dashboards to deliver key performance indicators (KPIs) including medicine consumption rates, reorder recommendations, staff utilization metrics, and bed occupancy statistics. Instead of relying on complex machine learning models, DRIMS emphasizes robust statistical methods, SQL-based aggregation, rule-based alerts, and visualization techniques that are interpretable by hospital staff. We validate the approach using a simulated dataset representative of a medium-sized hospital (6 months of operations, ~12,500 transactions) and report measurable improvements in inventory turnover, wastage reduction, and administrative efficiency.

Keywords: — Hospital Management System, Data Analytics, Inventory Management, Resource Optimization, Healthcare Informatics.

I. INTRODUCTION

Healthcare delivery relies not only on clinical expertise but also on efficient operational management. Hospitals, as complex socio-technical systems, must coordinate multiple resources—human, pharmaceutical, and infrastructural—to deliver timely care. While digital record-keeping has become more prevalent, many hospitals do not exploit this data to improve operations. Data remains fragmented across departments, and analytics capabilities are either absent or limited to ad-hoc Excel reports. Consequently, administrators face challenges in stock replenishment, staff scheduling, and capacity planning.

This paper proposes a practical, implementable solution—Data-Driven Hospital Resource and Inventory Management System (DRIMS)—that consolidates operational data, executes periodic analysis, and presents actionable dashboards and alerts. DRIMS is intentionally designed to avoid opaque predictive models; instead, it uses transparent statistical and rule-based techniques that hospital managers can understand and trust. Our objectives are to (1) reduce medicine wastage, (2) improve staff utilization, (3) optimize bed occupancy, and (4) streamline reporting for administrators.

The remainder of the paper is organized as follows: Section II reviews relevant literature and existing systems; Section III details the role of data analytics; Section IV presents system design and architecture; Section V describes methodology and implementation details; Section VI discusses evaluation results and practical implications; Section VII outlines future work; and Section VIII concludes the study.

II. LITERATURE REVIEW

Hospital information systems (HIS) and electronic medical records (EMR) have been widely studied. Projects such as OpenMRS and commercial products provide robust patient record management and billing functionality. However, research indicates a gap between record management and operational analytics.

Several studies have proposed machine learning approaches for demand forecasting and readmission prediction [1][2], but such approaches often require large labeled datasets and complex infrastructure that may not be available to smaller hospitals.

Works focusing on inventory management typically employ time-series forecasting or optimization algorithms to predict demand and minimize costs. While these methods can yield accurate forecasts under ideal conditions, they demand careful feature engineering and ongoing tuning. Our approach diverges by emphasizing descriptive and diagnostic analytics, combined with simple heuristics for reorder calculations. This makes the solution lighter, more interpretable, and easier to maintain while still delivering meaningful operational benefits.

III. ROLE OF DATA ANALYTICS

Data analytics in DRIMS is organized across three layers: descriptive analytics, diagnostic analytics, and rule-based operational logic. Descriptive analytics summarize historical activity—daily medication dispensation, hourly patient arrivals, monthly revenue—using SQL aggregations and Python data processing. Diagnostic analytics investigate anomalies and correlations (for example, linking spikes in antibiotic usage to seasonal influenza prevalence). Rule-based logic encodes domain knowledge (reorder thresholds, safety stock rules, and expiry prioritization) so that alerts and recommendations are actionable and clinically meaningful.

The selection of statistical over predictive methods is deliberate: hospital administrators typically prefer interpretable metrics and deterministic rules that can be audited. For instance, a reorder level may be computed as:

Reorder Level = (Average Daily Consumption × Lead Time) + Safety Stock

where Safety Stock = k × (Standard Deviation of Daily Consumption), and k is a tunable factor (commonly 1.65 for 95% service level). These calculations can be produced with simple SQL queries or lightweight Python scripts.

IV. SYSTEM DESIGN AND ARCHITECTURE

DRIMS adopts a modular, three-tier architecture (Presentation, Application, Data). The Presentation layer supports role-based web interfaces (Admin, Pharmacist, Doctor, Nurse) built with HTML/CSS/JavaScript and React.js components. The Application layer (Node.js + Express) exposes RESTful APIs for CRUD operations and analytic endpoints. The Data layer uses MySQL for relational storage and a lightweight data warehouse schema for analytics (star-schema with fact and dimension tables).

Key components include:

- Transactional Module: captures patient registrations, prescriptions, pharmacy dispensation, admissions/discharges, staff scheduling, and billing.
- Analytics Pipeline: scheduled ETL jobs (Python) that transform transactional data into summarized metrics and populate analytical tables.
- Dashboard & Reporting: Power BI reports or embedded Chart.js visualizations that present KPIs and support ad-hoc filtering.
- Notification Engine: a rule-based service responsible for sending email/SMS alerts for stock-outs, near-expiry items, and critical occupancy thresholds.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 18, April 2026
Impact Factor: 8.2

Figure 1 (inserted in presentation) depicts the high-level architecture: Frontend ↔ Backend ↔ MySQL (OLTP) and Analytics Warehouse (OLAP) → Visualization → Notification System. This separation enables optimized transactional performance while supporting heavy analytical queries on the warehouse.

V. DATA MODEL AND SAMPLE SCHEMA

The data model employs normalized transactional tables for day-to-day operations and a denormalized star schema for analytics. Core transactional tables include:

- Patients(patient_id, name, dob, gender, contact, address)
- Staff(staff_id, name, role, department, shift_pattern)
- Inventory(item_id, name, batch_no, expiry_date, quantity_on_hand, unit_price, supplier_id)
- Prescriptions(prescription_id, patient_id, doctor_id, date, medication_list)
- Admissions(admission_id, patient_id, ward_id, bed_id, admit_date, discharge_date)
- Transactions(txn_id, type, amount, date, related_id)

Analytical fact tables include MedicineUsageFact (date, item_id, quantity_issued, department_id) and AdmissionsFact (date, department_id, admissions_count, average_length_of_stay). Dimensions include DateDim, DepartmentDim, ItemDim, and StaffDim. This structure streamlines queries that compute rolling averages, seasonal indices, and month-over-month changes.

VI. METHODOLOGY

The development methodology follows Agile principles, with iterative sprints focusing on core modules (pharmacy, admissions, staff). Data-driven features were prioritized after the minimum viable product (MVP) for transactional operations was validated. Each sprint included unit testing for backend APIs, integration testing for ETL pipelines, and user acceptance testing (UAT) with domain experts (simulated pharmacists and administrators in our testbed).

Analytical methods are primarily statistical: moving averages (7-day, 30-day), coefficient of variation for demand volatility, seasonal index computation (ratio-to-moving-average), and safety stock formulas described earlier. ETL jobs were implemented in Python using Pandas for transformation and stored procedures for aggregations where high performance was necessary.

VII. ALGORITHM / PROCESS FLOW

We detail three principal process flows: Inventory Monitoring, Staff Utilization Calculation, and Bed Occupancy Reporting.

A. Inventory Monitoring Algorithm (Rule-based):

1. For each inventory item, compute Average Daily Consumption (ADC) over the past N days.
2. Compute Lead Time (days) from supplier data.
3. Calculate Reorder Level (RL) = ADC × Lead Time + Safety Stock.
4. If Quantity_on_Hand ⩽ RL, create a Reorder Alert with suggested reorder quantity = (Reorder Level + Target Buffer) - Quantity_on_Hand.
5. Additionally, compute Expiry Risk: list batches with expiry_date ⩽ today + ExpiryThreshold and flag for prioritized dispensing.

B. Staff Utilization Calculation:

1. Aggregate patient encounters per staff per shift.
2. Compute Utilization = (Patient Encounters) / (Standard Expected Encounters per Shift).
3. Classify staff as Underutilized, Normal, or Overutilized based on configurable thresholds.

C. Bed Occupancy Reporting:

1. For each ward, compute Daily Occupied Beds and Total Beds.
2. Occupancy Rate = (Occupied / Total) × 100.
3. Trigger alerts when occupancy exceeds safety thresholds (e.g., 90%).

Pseudo-code for inventory monitoring is provided in Appendix A. These algorithms are intentionally transparent and tunable, allowing hospital administrators to adjust safety factors and thresholds according to local policies.

VIII. IMPLEMENTATION DETAILS

The prototype was built using the following stack: Frontend - React.js with Bootstrap for responsive UI; Backend - Node.js (Express) providing REST APIs and JWT-based authentication; Database - MySQL for transactional storage and a separate MySQL-based analytics warehouse for aggregated metrics. Python (3.9) scripts executed via scheduled cron jobs perform ETL tasks and compute analytics which are then published to the analytics tables consumed by Power BI.

Security considerations included role-based access control (RBAC), HTTPS for transport security, parameterized queries to prevent SQL injection, and server-side validation for all inputs. Backup and recovery scripts were configured to run nightly, and logs were stored for audit purposes.

IX. EVALUATION AND RESULTS

To evaluate DRIMS, we constructed a synthetic dataset intended to emulate a medium-sized hospital: 6 months of operational data, ~12,500 inventory transactions, ~7,800 patient encounters, and ~3,200 admissions. The dataset included variability in demand to simulate seasonal effects (e.g., increased cough medication usage during monsoon months).

Evaluation metrics included: (1) Inventory Wastage Reduction (% decrease in expired units), (2) Inventory Turnover Ratio, (3) Staff Utilization Balance (standard deviation of per-staff workload), and (4) Average Time-to-Reorder (days). **Results indicated:**

- Inventory wastage decreased by ~32% compared to a baseline policy of fixed periodic reordering.
- Inventory turnover ratio improved by ~18% due to more responsive reorder recommendations.
- Standard deviation of staff workload decreased by ~25%, indicating better-balanced shifts.
- Average time-to-reorder reduced from 10 days (baseline) to 4 days, improving responsiveness.

These results are based on controlled simulations and should be interpreted as indicative rather than definitive. Real-world deployments will require calibration against local procurement lead times, supplier reliability, and clinical protocols.

X. DISCUSSION

DRIMS demonstrates that even lightweight, interpretable analytics can deliver substantial operational benefits to hospitals. The rule-based inventory management reduced wastage and improved turnover without requiring complex forecasting models. Similarly, staff utilization metrics helped administrators make empirically guided scheduling adjustments.

Importantly, the system's emphasis on transparency and maintainability addresses a common barrier in healthcare IT: trust. Hospital administrators and clinicians are more likely to adopt systems whose recommendations are explainable. DRIMS' reliance on standard statistical measures and configurable rules facilitates acceptance and governance

XI. LIMITATIONS

There are limitations to the current prototype. First, the evaluation uses simulated data; field deployment could reveal operational nuances not captured in simulation. Second, the absence of predictive models means DRIMS may not forecast rare surges (e.g., sudden outbreaks) as effectively as advanced ML solutions. Third, integration with legacy hospital systems can be non-trivial and may require custom ETL connectors.

Finally, while the system automates many processes, change management and staff training are essential for realizing benefits. Organizational readiness and data quality are critical success factors that fall outside the technical scope but are key to practical impact.

XII. FUTURE ENHANCEMENTS

Future work will focus on incremental additions that preserve interpretability while extending capability. Key directions include:

- Integrating lightweight time-series forecasting (e.g., Prophet) for short-term demand forecasting while maintaining explainability.
- Incorporating IoT sensors for automated bed occupancy and temperature-controlled inventory monitoring (cold chain).
- Building standardized connectors to common EMR/HIS systems to enable seamless data ingestion.
- Developing a mobile application for real-time notifications and approvals by administrators or pharmacists.

Policy and workflow extensions, such as supplier lead-time profiling and multi-supplier optimization, can further reduce procurement risk. Importantly, any added predictive modules will be accompanied by interpretable visualizations to maintain user trust.

XIII. CONCLUSION

This paper presents DRIMS, an implementable, data-driven Hospital Resource and Inventory Management System that leverages transparent statistical analytics and rule-based automation to improve operational efficiency. Through modular architecture, practical analytics, and emphasis on interpretability, DRIMS addresses common pain points in smaller healthcare institutions without imposing heavy infrastructural demands.

Simulation results suggest meaningful reductions in wastage and improvements in staff balance and responsiveness. With careful deployment, DRIMS can provide immediate operational value and form the foundation for future, more sophisticated analytics

REFERENCES

[1]. OpenMRS Project – https://openmrs.org
[2]. G. Hripcsak and D. J. Albers, 'Next-generation phenotyping of electronic health records', JAMA, 2013.
[3]. Bi, Jian-Wu, et al. 'Forecasting daily tourism demand for tourist attractions with big data: an ensemble deep learning method.' Journal of Travel Research, 2022.
[4]. Behera, S., & Mohanty, S., 'Digital Transformation in Hospital Management,' Journal of Healthcare Informatics, 2023.
[5]. K. Sharma et al., 'Data Analytics in Healthcare Management Systems,' IJCSIT, Vol. 12, No. 5, 2023.
[6]. World Health Organization (WHO), 'Health Data Management and Digital Transformation,' 2021.
[7]. M. Singh et al., 'Efficient Data Analytics in Healthcare Operations,' International Journal of Computer Applications, 2022.
[8]. H. W. Bierman and I. J. Fleming, 'Inventory Control and Management in Healthcare', Health Systems Journal, 2019.
[9]. S. Elkington et al., 'Hospital bed occupancy and its effect on patient flow', International Journal of Healthcare, 2020.
[10]. T. Hastie, R. Tibshirani, and J. Friedman, 'The Elements of Statistical Learning', Springer, 2009.
[11]. Prophet (Facebook) documentation - https://facebook.github.io/prophet/
[12]. Open Data Kit, 'Best Practices for Data Collection in Healthcare Settings', 2018.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 18, April 2026
Impact Factor: 8.2

APPENDIX A: SAMPLE SQL QUERIES & PSEUDO-CODE

Sample SQL to compute average daily consumption (ADC) for an item over the past 30 days:

```
SELECT item_id, SUM(quantity_issued)/30.0 AS avg_daily_consumption
FROM medicine_usage
WHERE date >= DATE_SUB(CURDATE(), INTERVAL 30 DAY) AND item_id = <ITEM_ID>
GROUP BY item_id;
```

Sample SQL to identify near-expiry batches:

```
SELECT item_id, batch_no, expiry_date, quantity_on_hand
FROM inventory
WHERE expiry_date <= DATE_ADD(CURDATE(), INTERVAL 30 DAY)
ORDER BY expiry_date ASC;
```

Pseudo-code (Inventory Monitoring):

```
for item in inventory:
    adc = compute_adc(item, days=30)
    lead_time = get_lead_time(item)
    safety_stock = k * stddev_consumption(item)
    reorder_level = adc * lead_time + safety_stock
    if item.quantity_on_hand <= reorder_level:
alert_reorder(item, reorder_level)
```