



Design and Implementation of a Multi-Tenant Web-Based Asset Lifecycle Management System with Automated Inventory Alerts and Vendor Performance Analytics

Arun Karthik V¹, Niranjan J², Balaaditya B³

Department of Software Systems and AIML, Sri Krishna Arts and Science College, Coimbatore - 641008, India

ABSTRACT :

Organizations that own large numbers of physical assets, such as educational institutions, hospitals and manufacturing units, often track equipment, maintenance and spare parts in spreadsheets and paper registers. This makes it hard to know the condition of an asset, plan maintenance on time, avoid running out of critical spare parts and judge how well outside service vendors perform. This paper presents the design and implementation of a web-based Asset Lifecycle Management (ALM) system that follows an asset from procurement and registration through assignment, operation and maintenance to scrapping and disposal. The system uses a React web frontend, a Node.js and Express REST API and PostgreSQL, with a separate database for each client organization so that every organization's data stays isolated. Key contributions include an approval-based maintenance workflow, a rule-based Stock and Purchase Requirement report that classifies spare parts as out of stock or needing purchase against a configured minimum stock level, automated daily stock alerts delivered once per part and status through in-app and push notifications, downloadable PDF reports, and a Service Level Agreement (SLA) and Vendor Performance report. Observations from a pilot deployment show that the alerting design removed duplicate notifications and that the quality of vendor analytics depends directly on complete workflow and SLA data entry.

Keywords: Asset Lifecycle Management; Maintenance Management; Multi-Tenant SaaS; Inventory Control; Service Level Agreement

Introduction

Physical assets such as machines, vehicles, laboratory equipment, computers and building services represent a large share of the capital of most organizations. How well these assets are managed affects running costs, safety, service quality and how long the assets remain useful. Asset management is defined by the international standard ISO 55000 as the coordinated activity of an organization to realize value from its assets (International Organization for Standardization, 2014). In practice this means knowing what assets exist, where they are, who is responsible for them, what condition they are in and what they cost to keep running.

Nomenclature

ALM	Asset Lifecycle Management	KPI	Key Performance Indicator
API	Application Programming Interface	PM	Preventive Maintenance
CMMS	Computerized Maintenance Management System	SLA	Service Level Agreement
FCM	Firebase Cloud Messaging	WO	Work Order

Many small and medium organizations still manage this information manually. Asset lists are kept in spreadsheets, maintenance is recorded in paper log books and spare parts are counted only when something breaks. Such records are difficult to keep up to date, are spread across departments and give managers little warning before a problem occurs. A missing spare part can keep a machine out of service for days, and without records it is hard to tell whether an outside vendor is responding and repairing equipment within the agreed time.

Computerized Maintenance Management Systems (CMMS) were developed to address part of this problem by recording assets, work orders and maintenance history (Campbell & Reyes-Picknell, 2015). However, many commercial systems are expensive, are designed for a single organization and are complex to configure. There is a need for an affordable, web-based system that covers the full asset lifecycle, can serve several client organizations from one deployment and actively warns users about problems instead of waiting for them to look.

This paper describes the design and implementation of such an Asset Lifecycle Management (ALM) system. The main contributions are: (i) a three-layer web architecture with a separate database for each organization; (ii) an approval-based maintenance workflow that distinguishes in-house maintenance from external vendor service; (iii) a clear, rule-based method for identifying spare parts that are out of stock or need to be purchased; (iv) an automated notification design that sends each stock alert only once; and (v) reporting modules, including a downloadable Stock and Purchase report

and an SLA and Vendor Performance report. The remainder of the paper reviews related work, states the problem and objectives, explains the architecture and implementation, discusses results and concludes with directions for future work.

Literature review

1.1. Physical asset management

ISO 55000 frames asset management as a lifecycle activity that balances cost, risk and performance from the time an asset is acquired until it is disposed of (International Organization for Standardization, 2014). Hastings (2015) describes the same lifecycle in practical terms, covering asset planning, acquisition, operation, maintenance and replacement decisions, and stresses that reliable asset registers are the foundation of all later decisions. These works motivate a system in which every stage of an asset's life is recorded against a single asset record.

1.2. Maintenance strategies and CMMS

Maintenance strategies range from corrective maintenance, carried out after a failure, to preventive maintenance at fixed intervals and condition-based maintenance driven by measured equipment condition. Garg and Deshmukh (2006) review the maintenance management literature and highlight the growing role of information systems in planning and measuring maintenance. Bevilacqua and Braglia (2000) show that the right strategy differs from one asset to another, which requires good data on failures and costs. Jardine, Lin and Banjevic (2006) review condition-based maintenance and note that its benefits depend on collecting and storing maintenance and failure data consistently. Moubray (1997) and Campbell and Reyes-Picknell (2015) emphasize that recording breakdowns, repeat failures and repair times is necessary for continuous improvement. A CMMS that captures maintenance schedules, work orders and breakdowns therefore provides the data needed for these strategies.

1.3. Spare parts and inventory control

Classical inventory control uses a reorder point or minimum stock level: when the quantity on hand falls to that level, a replenishment order is raised (Silver, Pyke, & Peterson, 1998). Spare parts are a special case because demand is irregular and a shortage can stop production or service. A simple, transparent rule that compares available stock with a configured minimum level is easy for store staff to understand and act on, and it is the approach adopted in this work.

1.4. Multi-tenant software architecture

Software as a Service (SaaS) applications often serve several client organizations, called tenants, from one deployment. Chong and Carraro (2006) describe a spectrum of data architectures, from a separate database per tenant to a fully shared schema, trading isolation against cost. Bezemer and Zaidman (2010) discuss the maintenance challenges of multi-tenant applications, and Krebs, Momm and Kounev (2012) identify isolation, configurability and performance as central architectural concerns. For asset data, which is commercially sensitive, the separate-database approach offers strong isolation and simple per-organization backup. Web services in this work follow the Representational State Transfer (REST) style described by Fielding (2000).

Problem statement and objectives

The problem addressed is the lack of a single, affordable system through which an organization can track its assets across their whole life, plan and approve maintenance, control spare parts and evaluate vendors, while keeping each organization's data separate. The specific objectives of the work are:

- to maintain a central register of assets with their type, location, assignment and status;
- to support a maintenance workflow in which schedules are approved before work orders are created and completed;
- to calculate available spare part stock and clearly identify parts that are out of stock or need to be purchased;
- to alert the right users automatically, once per problem, through in-app and push notifications;
- to provide reports that can be viewed on screen and downloaded as PDF or Excel files; and
- to serve several organizations from one deployment while keeping their data fully isolated.

System architecture and methodology

The system follows a three-layer architecture, shown in Fig. 1. The presentation layer is a single-page web application built with React and the Vite build tool. It contains the dashboard, asset screens, maintenance workflow screens, reports and the notification centre. Application state is held in a lightweight store, and reports can be exported in the browser as PDF files.

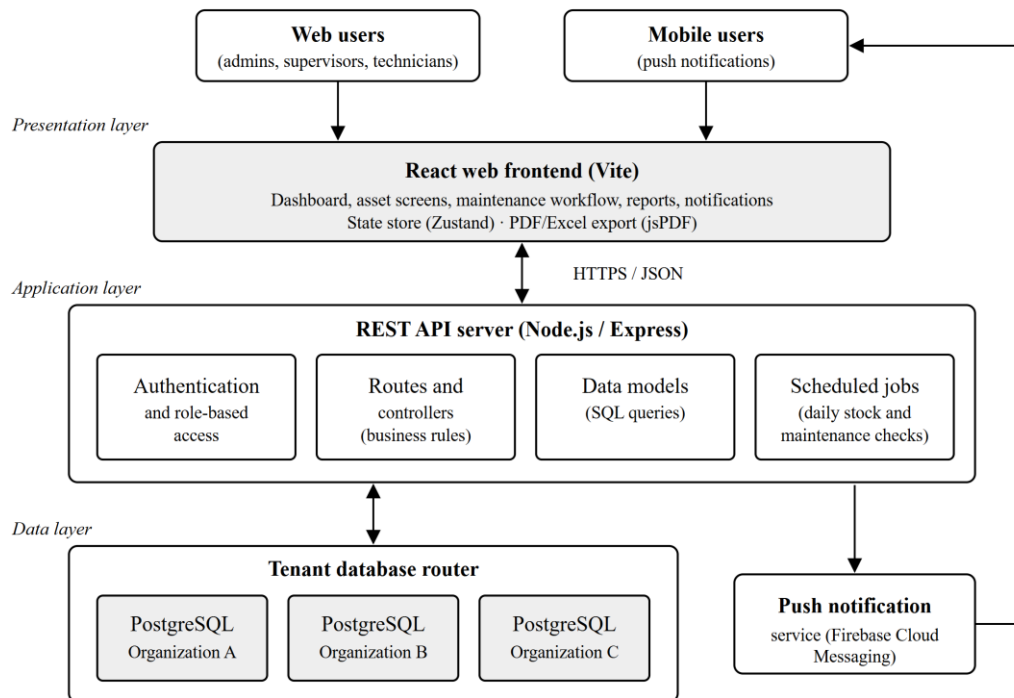


Fig. 1 - Overall architecture of the ALM system.

The application layer is a REST API built with Node.js and the Express framework. Each request passes through authentication and role-based access checks before it reaches a controller, which applies the business rules and calls data models that run SQL queries. A scheduler inside the server runs background jobs at fixed times each day, for example to check stock levels and missed consumption entries. Push notifications to mobile devices are sent through Firebase Cloud Messaging (FCM).

The data layer uses PostgreSQL. Following the separate-database model of Chong and Carraro (2006), each client organization has its own database. When a request arrives, the server identifies the user's organization from the login token and selects the matching database connection for the rest of that request, as shown in Fig. 2. Every query therefore runs only against the data of that organization, which prevents data from one organization appearing in another organization's screens or reports.

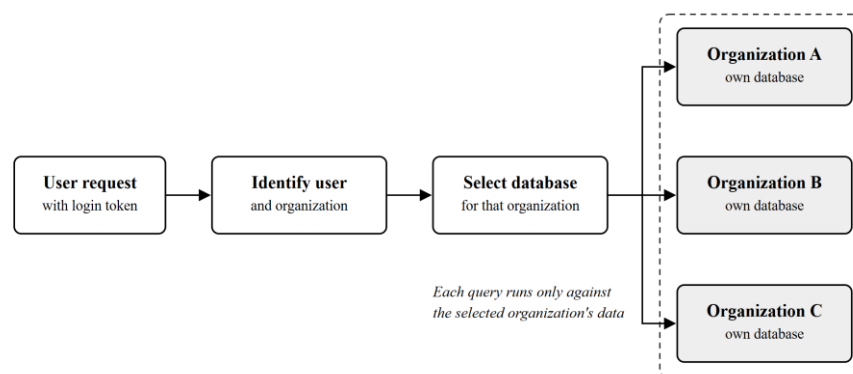


Fig. 2 - Selecting the correct organization database for each request.

Development followed an iterative approach. Features were built in short cycles, tested locally against copies of organization data, reviewed and then merged into a shared production branch using a version control system. Problems reported by users were reproduced, traced to their cause, fixed and re-tested before release. End-to-end browser tests were written for important flows such as supervisor approval of maintenance.

Implementation of the modules

1.5. Asset registry and lifecycle

The asset registry stores each asset with a unique identifier, asset type, description, location, purchase details, warranty and current assignment. The stages an asset passes through are shown in Fig. 3. After procurement an asset is registered and assigned to a department or user, after which it is in operation. When a preventive service falls due or a breakdown is reported, the asset enters maintenance and returns to operation once repaired. At the end of its useful life the asset is retired, and the system records its scrapping and any scrap sale. Keeping all these events against one asset record gives a complete history for each asset.

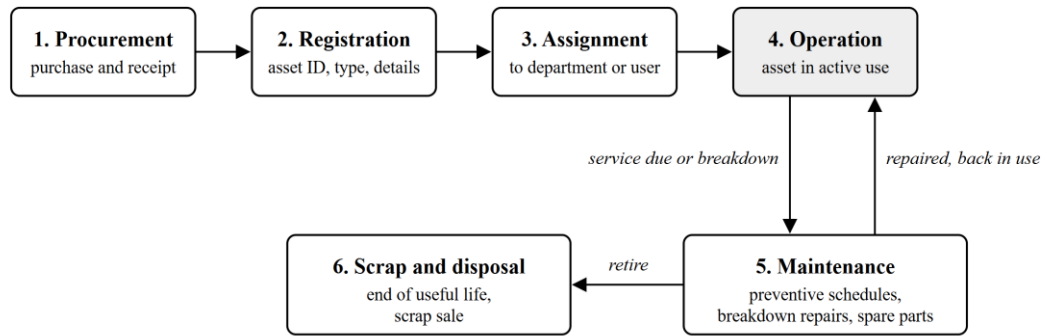


Fig. 3 - Lifecycle stages of an asset in the ALM system.

1.6. Maintenance workflow and approvals

Preventive maintenance schedules are generated for each asset type at configured intervals. A schedule is first created as a workflow item that must be approved by the responsible supervisor. Only after approval is a work order created, carried out and marked as completed with its actual start and end dates. Each work order is linked either to the in-house maintenance team or to an external vendor. Breakdowns are recorded separately with their reason and date, which allows repeat failures of the same asset to be identified. This approval step ensures that maintenance spending is authorized and that every completed job has a clear record.

1.7. Spare parts and stock control

Spare parts are grouped into categories, such as filters, seal kits, printed circuit board (PCB) components and drive belts, and each category has a configured minimum stock level. Individual units of a part are recorded as they are received, and a unit is marked as used when it is issued to a maintenance job. Requests for spare parts move through requested, reserved and issued states. The available stock of a part is the number of its units that have not yet been used. The stock status of each part category p is then decided by Equation (1), where $A(p)$ is the available stock and $M(p)$ is the configured minimum stock level.

$$S(p) = \begin{cases} \text{Out of stock,} & \text{if } A(p) = 0 \\ \text{Needs purchase,} & \text{if } 0 < A(p) \leq M(p) \\ \text{Sufficient (hidden),} & \text{if } A(p) > M(p) \end{cases} \quad (1)$$

Parts with sufficient stock are hidden from the purchase report so that store staff see only the parts that need attention. Earlier versions of the report also used reorder levels and other sources for the minimum quantity, which confused users when parts with plenty of stock appeared as needing purchase. Using the configured minimum stock as the single source of truth made the report consistent and easy to explain.

1.8. Notifications and scheduled jobs

The notification centre combines several kinds of alerts in one list: pending workflow approvals, warranty and expiry reminders, spare part requests, missed consumption entries and stock alerts. Stock alerts are raised when a part becomes out of stock or needs purchase. A scheduled job runs every morning to re-check stock levels, and the check also runs whenever the purchase report is opened. Alerts are delivered in the web application and as push notifications to users who have access to the stock reports or spare part approval screens, and to administrators.

A key design decision is that each alert is stored once per organization, part and status, identified by a group key, rather than once per recipient. A new push notification is sent only when the alert is first created. When stock is replenished and the part no longer needs attention, the alert is closed automatically. Stock alerts also do not show deadline dates or overdue labels, because these apply to approval tasks and not to stock conditions.

1.9. Reports and dashboard

The dashboard gives an overview of assets, maintenance due, breakdowns, notifications and the status of scheduled jobs. The Stock and Purchase Requirement report lists every part that is out of stock or needs purchase, with its available and minimum quantity, linked work orders and upcoming preventive maintenance, and summary cards for the number of parts to buy and parts out of stock. The report can be exported to Excel and downloaded as a formatted PDF document.

The SLA and Vendor Performance report evaluates external vendors. For each vendor, a response time target and a resolution time target are configured. For completed work orders handled by external vendors in the selected period, the report compares the actual resolution time with the target and calculates SLA compliance using Equation (2). It also shows breaches, average response time from vendor SLA records, average service rating and repeat failures.

$$C = \frac{N_{met}}{N_{scored}} \times 100 (\%) \quad (2)$$

Here N_{met} is the number of scored work orders resolved within the vendor's resolution target and N_{scored} is the number of completed external work orders for which a target is configured. In-house maintenance is excluded because the report is intended to measure outside vendors.

Results and discussion

The system was deployed for several client organizations, each with its own database. Table 1 summarizes the modules that were implemented and their main features.

Table 1 - Implemented modules and their main features.

Module	Main features
Asset registry	Asset records, types, locations, assignment, warranty and status history
Maintenance	Preventive schedules, supervisor approval, work orders, in-house and vendor jobs
Breakdowns	Breakdown reporting, reasons and repeat failure tracking
Spare parts	Categories with minimum stock, unit tracking, request, reserve and issue
Notifications	Unified notification centre, stock alerts, push notifications
Scheduled jobs	Daily stock checks and missed consumption checks with a monitoring screen
Reports	Stock and Purchase report (PDF and Excel), SLA and Vendor Performance report
Scrap	Scrapping of retired assets and recording of scrap sales

Table 2 compares the system with the manual, spreadsheet-based practice it is intended to replace. The comparison is qualitative and based on the capabilities of each approach.

Table 2 - Comparison with manual, spreadsheet-based asset tracking.

Aspect	Manual / spreadsheet	Proposed ALM system
Asset history	Spread across files and registers	Single record per asset for its whole life
Maintenance approval	Informal, often verbal	Recorded approval before work orders
Low stock detection	Found when a part is needed	Detected daily by a clear rule
User alerts	None	In-app and push notifications, once per problem
Vendor evaluation	Based on memory	SLA compliance, breaches and ratings
Reports	Prepared by hand	On screen, PDF and Excel on demand
Multiple organizations	Separate copies of files	One deployment, isolated databases

Two observations from the pilot deployment are worth discussing. First, the initial version of the stock alerts created a separate alert record for every eligible user. In one organization this produced 38 alert records for only two parts, and users received the same message many times. After the change to one alert per organization, part and status, the same situation produced two alerts, one for each part, and each push notification was sent only once. This shows the importance of designing notifications around the problem being reported rather than around the people receiving it.

Second, the SLA and Vendor Performance report initially appeared to show almost no data for one organization. Analysis of that organization's records showed that the report logic was correct but the underlying data was incomplete. Recent work orders had been carried out by the in-house team, which the report excludes by design, and the only external vendor work orders were more than a year old. In addition, 125 maintenance schedules were still waiting for approval and had never become work orders, and 15 of 17 vendor SLA records were not linked to any work order. Analytics of this kind

are only as good as the discipline with which workflows are completed and SLA details are entered, which agrees with the emphasis on reliable data in the maintenance literature (Campbell & Reyes-Picknell, 2015; Jardine et al., 2006).

Other practical lessons included keeping business rules identical across the backend, the on-screen report and exported files, and designing the dashboard so that cards and lists remain readable on different screen sizes.

Conclusion and future scope

This paper presented a web-based, multi-tenant Asset Lifecycle Management system that records assets across their whole life, supports approved maintenance workflows, controls spare parts with a simple minimum stock rule, alerts users automatically and provides on-screen and downloadable reports, including vendor SLA analytics. A separate database for each organization keeps client data isolated while allowing one deployment to serve many organizations. Experience from the pilot deployment showed that well-designed alerts reduce noise for users and that the value of analytics depends on complete and timely data entry.

Future work includes a dedicated mobile application for technicians, predictive maintenance using breakdown history, integration of Internet of Things (IoT) sensors for condition-based maintenance, automatic purchase requests when parts need purchase, guided data entry to improve SLA record quality, and a larger suite of automated tests.

Acknowledgements

The authors thank Arun Karthik V, Department of Software Systems and AIML, for guidance throughout this work, and for the opportunity to work on the Asset Management system and for providing access to the development environment and pilot deployment.

REFERENCES

- Bevilacqua, M., & Braglia, M. (2000). The analytic hierarchy process applied to maintenance strategy selection. *Reliability Engineering & System Safety*, 70(1), 71–83.
- Bezemer, C.-P., & Zaidman, A. (2010). Multi-tenant SaaS applications: Maintenance dream or nightmare? In *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOl) and International Workshop on Principles of Software Evolution (IWPSE)* (pp. 88–92). New York: ACM.
- Campbell, J. D., & Reyes-Picknell, J. V. (2015). *Uptime: Strategies for excellence in maintenance management* (3rd ed.). Boca Raton: CRC Press.
- Chong, F., & Carraro, G. (2006). *Architecture strategies for catching the long tail*. Redmond: Microsoft Corporation.
- Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* (Doctoral dissertation). University of California, Irvine.
- Garg, A., & Deshmukh, S. G. (2006). Maintenance management: Literature review and directions. *Journal of Quality in Maintenance Engineering*, 12(3), 205–238.
- Hastings, N. A. J. (2015). *Physical asset management: With an introduction to ISO 55000* (2nd ed.). Cham: Springer.
- International Organization for Standardization. (2014). *ISO 55000:2014 Asset management — Overview, principles and terminology*. Geneva: ISO.
- Jardine, A. K. S., Lin, D., & Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, 20(7), 1483–1510.
- Krebs, R., Momm, C., & Kounev, S. (2012). Architectural concerns in multi-tenant SaaS applications. In *Proceedings of the 2nd International Conference on Cloud Computing and Services Science (CLOSER)* (pp. 426–431). Porto: SciTePress.
- Moubray, J. (1997). *Reliability-centered maintenance* (2nd ed.). New York: Industrial Press.
- Silver, E. A., Pyke, D. F., & Peterson, R. (1998). *Inventory management and production planning and scheduling* (3rd ed.). New York: John Wiley & Sons.