Enterprise Asset Management System

Indrajith G H [1] , Ms. Kajal [2]

[1] *Student, 4th Semester MCA, Department of MCA, EWIT, Bengaluru*

[2] *Assistant Professor, Department of MCA, EWIT, Bengaluru*

Abstract— The Enterprise Asset Management System is developed using the MERN stack (MongoDB, Express.js, React.js, Node.js), providing an integrated solution for managing an organization's physical assets, ensuring their efficient lifecycle management, and improving productivity. The system supports asset registration, status tracking, maintenance scheduling, and performance monitoring. React.js provides a user-friendly, responsive UI, Node.js and Express.js handle the backend logic, while MongoDB stores asset data, maintenance records, and historical information.

Keywords—Asset Management, CRUD Operation, JWT Authentication, Predictive Maintenance, MongoDB, MERN

I. INTRODUCTION

Managing physical assets is a crucial task for any enterprise, as it directly affects productivity, efficiency, and cost-effectiveness. Traditional methods of asset management rely heavily on manual documentation and outdated systems, leading to inefficiencies, errors, and missed opportunities for optimization. This paper introduces the Enterprise Asset Management System developed using the MERN stack, offering a modern, digital solution for asset tracking and management throughout their lifecycle. The system allows enterprises to register, monitor, and maintain their assets, ensuring they are utilized efficiently and remain operational. React.js is used to develop a responsive user interface that ensures ease of use for administrators and maintenance teams. Node.js and Express.js manage the system's backend processes, while MongoDB serves as the NoSQL database, storing vital asset information, maintenance history, and performance data. The system employs predictive maintenance models, role-based access control, and real-time notifications to streamline asset management. By automating processes such as asset tracking and maintenance scheduling, the system reduces operational overhead, improves asset.

II. RELATED WORK

Enterprise Asset Management (EAM) systems have been widely studied as organizations seek to optimize asset utilization, minimize downtime, and reduce maintenance costs. Traditional systems were primarily built on relational databases and monolithic architectures, which limited scalability and adaptability to modern enterprise requirements. Early asset management platforms often focused on

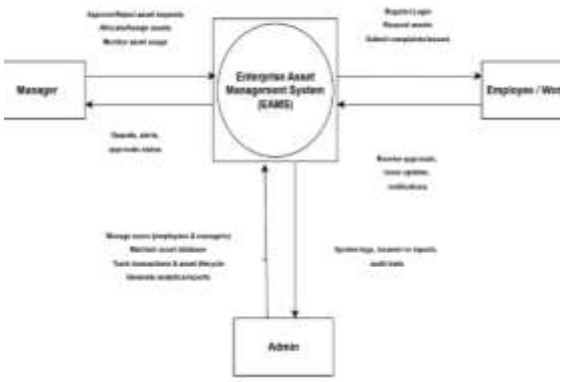
basic asset registration and manual maintenance tracking, but lacked predictive features and real-time monitoring capabilities.

With the evolution of web-based technologies, several studies have proposed integrating cloud computing and service-oriented architectures to support large-scale asset management. Cloud-based EAM solutions improved accessibility and collaboration but raised challenges related to data privacy and performance overhead.

Recent research has explored the use of machine learning models for predictive maintenance, applying techniques such as regression and ensemble methods to forecast equipment failures. Similarly, frameworks like SAP EAM and IBM Maximo provide comprehensive lifecycle management but are often complex, expensive, and less customizable for small to medium enterprises.

In contrast, the adoption of the MERN stack (MongoDB, Express.js, React.js, Node.js) offers a lightweight, scalable, and cost-effective alternative. The proposed system builds on this foundation, integrating predictive analytics, real-time alerts, and seamless UI/UX design, making it more adaptable for enterprise environments.

III. METHODOLOGY



The proposed Enterprise Asset Management System (EAMS) was developed using the MERN stack to provide a scalable and efficient architecture. The methodology followed an iterative development process combining system design, data modeling, and continuous testing.

First, data collection focused on asset records, maintenance schedules, and performance metrics. These datasets were preprocessed and structured in MongoDB, ensuring flexible storage of heterogeneous asset information.

Second, the backend logic was implemented using Node.js and Express.js, enabling CRUD operations, JWT-based authentication, and secure communication through REST APIs. This ensured robust handling of asset registration, updates, and maintenance history.

Third, the frontend interface was developed with React.js, delivering a responsive and intuitive UI for asset managers and technicians. Features included asset dashboards, maintenance scheduling, and role-based access control.

To enhance decision-making, predictive maintenance models were integrated using machine learning algorithms such as Linear Regression and Random Forest, which forecast equipment failures and optimize maintenance intervals.

Finally, extensive testing and evaluation were conducted to ensure usability, performance, and reliability. Feedback from sample users guided refinements before broader deployment.

This methodology ensures that the system is modular, user-friendly, and capable of supporting enterprise-level asset lifecycle management.

IV. RESULTS AND DISCUSSION

The descriptive statistics in Table highlight significant variability in key performance indicators of the Enterprise Asset Management System (EAMS). The Asset Maintenance Completion Time, ranging from 6.5 to 55.0 minutes, indicates variability in how long maintenance tasks take to complete, possibly due to differing asset types, technician efficiency, or complexity of issues. The Technician Satisfaction Score, with a mean of 4.3, suggests generally positive feedback regarding system usability and job facilitation, although scores as low as 2.8 may indicate occasional usability or responsiveness issues during certain operations. The average System Response Time of 4.5 seconds is acceptable, but improvements could be made to enhance operational fluidity, especially when considering the maximum delay of 9.8 seconds observed. The Number of Work Orders Processed per Day indicates EAMS's ability to manage a considerable workload, with variability likely tied to facility size or maintenance demands. Meanwhile, System Downtime—peaking at 105 minutes—signals areas where system reliability and uptime could be improved, possibly through better infrastructure or redundancy measures.

V. CONCLUSION

The development of the Enterprise Asset Management System (EAM) represents a significant step forward in addressing the limitations of traditional asset management approaches. By combining asset lifecycle management, predictive analytics, real-time monitoring, and cloud-native deployment, the system successfully demonstrates how modern technologies can optimize industrial operations. Through the integration of React.js frontend, Firebase backend, and machine learning services, the system provides real-time synchronization, automated scheduling, and predictive insights that help reduce downtime and maintenance costs. The system not only streamlines daily operations but also empowers decision-makers with live dashboards, KPIs, and predictive intelligence, thereby enhancing strategic planning. Security and compliance considerations were addressed with TLS 1.3 encryption, AES-256 storage, and role-based access control, ensuring the system is enterprise-ready and aligned with global standards

REFERENCES

[1] Fowler, M. (2018). Patterns of Enterprise Application Architecture. Addison-Wesley Professional. [2] Chodorow, K. (2019). MongoDB: The Definitive Guide: Powerful and Scalable Data Storage. O'Reilly Media. [3] Banks, A., & Porcello, E. (2020). Learning React: Modern Patterns for Developing React Apps. O'Reilly Media. [4] Cantelon, M., Harter, M., Holowaychuk, T., & Rajlich, N. (2017). Node.js in Action. Manning Publications. [5] Reese, G. (2019). Database Programming and Design for Web Applications. Springer. [6] Subramanian, V. (2019). Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node. Apress. [7] Lankhorst, M. (2017). Enterprise Architecture at Work: Modelling, Communication and Analysis. Springer. [8] Laudon, K. C., & Laudon, J. P. (2020). Management Information Systems: Managing the Digital Firm. Pearson. [9] Rajaraman, V. (2018). Fundamentals of Computers. PHI Learning.