|  |
| --- |
| CO3201 Computer Science Project Interim Report |
| **Web-Based Inventory Management System for Small Business** |
| School of Computing and Mathematical Sciences, University of Leicester |

|  |
| --- |
| Choudhury, Arif  11-21-2024 |

Table of Contents

[**Declaration** ii](#_Toc183101686)

[**Abstract** 1](#_Toc183101687)

[**1.** **Aims and Objectives** 1](#_Toc183101688)

[**1.1 Aims** 1](#_Toc183101689)

[**1.2 Objectives** 1](#_Toc183101690)

[**1.3 Challenges and Originality** 2](#_Toc183101691)

[**1.3.1 Challenges:** 2](#_Toc183101692)

[**1.3.2 Originality:** 2](#_Toc183101693)

[**2.** **Survey of Literature/ Information Sources** 3](#_Toc183101694)

[**2.1 Overview of Cloud-Based Solutions and Inventory Management** 3](#_Toc183101695)

[**2.3 Challenges with Current Inventory Management Systems** 3](#_Toc183101696)

[**2.4 Importance of Real-Time Data and Scalability** 4](#_Toc183101697)

[**2.5 Data Visualization and Reporting** 4](#_Toc183101698)

[**2.6 Conclusion** 4](#_Toc183101699)

[**3.** **Requirements** 5](#_Toc183101700)

[**3.1** **Functional Requirements** 5](#_Toc183101701)

[**3.2** **Non-Functional Requirements** 7](#_Toc183101702)

[**3.3 Hardware Requirements** 7](#_Toc183101703)

[**3.4 Software Requirements** 7](#_Toc183101704)

[**4.** **Outline of Specification and Design** 8](#_Toc183101705)

[**4.1** **System Architecture** 8](#_Toc183101706)

[**4.1.1** **Frontend** 9](#_Toc183101707)

[**4.1.2** **Backend** 9](#_Toc183101708)

[**4.1.3** **Database** 10](#_Toc183101709)

[**4.2** **Important Algorithms and Data Structures** 10](#_Toc183101710)

[**4.2.1 Role-Based Access Control (RBAC) Algorithm** 10](#_Toc183101711)

[**4.2.2 Real-time sync using Firebase** 10](#_Toc183101712)

[**4.2.3 Low-Stock Notification Algorithm** 10](#_Toc183101713)

[**4.2.4 Data Aggregation for Reporting** 11](#_Toc183101714)

[**4.2.5 Data Structure for Inventory Management** 11](#_Toc183101715)

[**5.** **Planning and Timescales** 11](#_Toc183101716)

[**5.1** **Semester 1** 11](#_Toc183101717)

[**5.2** **Semester 2** 12](#_Toc183101718)

[**References** 13](#_Toc183101719)

# **Declaration**

A close-up of a text

Description automatically generated

Arif Choudhury



# **Abstract**

Small businesses often struggle with inventory issues like shortages, inconsistencies, and inaccuracies, which can affect their operations. This project aims to address these challenges by developing a web-based inventory management system that is simple, efficient, and accessible. The system will include real-time inventory updates, user-friendly reporting tools, and data visualisation features to help business owners make informed decisions. By utilising cloud technology, the system ensures secure synchronisation, remote access, and scalability to grow with the business. Features like role-based access control and customisable reporting will make the system practical and easy to use for non-technical users. Ultimately, this project strives to reduce errors, save time, and boost the operational efficiency of small businesses.

# **Aims and Objectives**

## **1.1 Aims**

The aim of this project is to design and implement a cloud-based inventory management system that directly addresses the operational inefficiencies faced by small businesses. By providing real-time insights and seamless inventory control, this project seeks to minimise stock discrepancies, reduce operational delays, and empower business owners to make data-driven decisions. The system will enhance business agility, enabling owners to respond proactively to market demands while maintaining optimal stock levels.

Key focus areas include:

* Simplifying inventory tracking for non-technical users through an intuitive interface.
* Leveraging cloud technology to enable secure, real-time data synchronisation and remote access.
* Ensuring scalability to support business growth with larger inventories and more users.
* Introducing dynamic reporting tools that adapt to evolving business needs, aiding strategic planning.

## **1.2 Objectives**

The objectives of this project are defined to ensure the aims are met in a measurable and practical manner:

* Implement a secure, role-based user authentication system (RBAC): Facilitate differentiated access for Admins, Managers, and Staff, ensuring secure and restricted access to system features.
* Develop an intuitive, responsive user interface: Utilise React and Bootstrap to create a seamless experience, reducing onboarding time and improving task efficiency for non-technical users.
* Enable real-time inventory updates: Leverage Firebase to synchronise stock levels instantaneously, providing consistent and accurate data across multiple devices.
* Integrate advanced data visualisation: Use Chart.js to deliver interactive dashboards that display inventory trends, sales performance, and low-stock alerts for better decision-making.
* Deploy a scalable, cloud-based architecture: Ensure the system supports increased traffic and larger datasets through platforms like Firebase.
* Develop exportable, customisable reporting features: Allow users to generate reports in formats such as CSV, tailored to their specific business needs.

## **1.3 Challenges and Originality**

### **1.3.1 Challenges:**

Developing this project presents a range of technical and design challenges, such as:

* Concurrency in Real-Time Data Management: Synchronising inventory updates across multiple devices without performance degradation requires robust database operations and optimized real-time syncing using Firebase. Handling concurrent user operations is a critical challenge, necessitating careful system design to prevent conflicts.
* User Experience Design for Non-Technical Users: Achieving a balance between simplicity and functionality is a challenge. The system must maintain ease of use while offering advanced features like role-based views and interactive dashboards.
* Data Security and Privacy in a Cloud Environment: Protecting sensitive business data involves secure authentication mechanisms (JWTs) and encrypted communication (HTTPS). Balancing robust security with system performance is a key concern.

### **1.3.2 Originality:**

This project introduces a unique combination of real-time synchronisation, role-based access control, and interactive reporting tailored specifically for small businesses. Unlike existing off-the-shelf systems, which are often feature-heavy and costly, this solution emphasises simplicity and cost-effectiveness. By integrating advanced data visualisation tools, the system not only tracks inventory but also provides actionable insights, enabling businesses to optimize their stock management strategies dynamically. The use of Firebase ensures scalability while minimising infrastructure costs, setting this system apart from traditional inventory platforms.

# **Survey of Literature/ Information Sources**

## **2.1 Overview of Cloud-Based Solutions and Inventory Management**

When designing a cloud-based inventory management system for small businesses, essential insights from Chukwumuanya et al. (2024) and Shopify's guide on cloud inventory management are quite useful. Chukwumuanya et al. (2024) emphasise the usage of a MySQL-based web platform to store inventory data and apply forecasting models such as ARMA for better inventory management. Shopify emphasises the need of real-time updates, demand forecasting, and multi-location management, which are all essential elements for modern inventory systems. My solution will combine these methodologies to provide real-time tracking, automated ordering, and forecasting capabilities, with the goal of streamlining inventory procedures, lowering costs, and improving decision-making for small businesses.  
  
**2.2 Cloud-Based Inventory Systems' Advantages**Cloud-based inventory management systems have transformed how businesses manage their operations, providing considerable benefits over traditional desktop systems. Firebase's Cloud Firestore improves scalability and real-time synchronisation, allowing organisations to rapidly track inventory modifications across many devices, assuring accurate and up-to-date data. This adaptability is critical for managing fluctuating inventory demands. Unlike desktop solutions that are limited by physical hardware and manual upgrades, cloud systems, such as those powered by Firestore, provide seamless integrations, automatic updates, and comprehensive security features such as data encryption and disaster recovery (Firebase, n.d.; Linnworks, 2024). The ability to manage inventories remotely, along with advanced querying features, improves operational agility and efficiency, particularly for businesses that experience variable demand or seasonal variances.

**2.3 Challenges with Current Inventory Management Systems**  
Inventory management faces challenges such as inconsistent tracking, warehouse inefficiencies, and manual errors, leading to data inaccuracies and operational delays. Shifting demand can cause overstocking or stockouts, affecting cash flow and customer satisfaction. Supply chain complexities and poor communication further hinder decision-making. These issues highlight the need for automated solutions that provide real-time data and improve efficiency (NetSuite, 2022).

Due to the high cost of specialised software, small businesses often use error-prone spreadsheet systems. Studies show that switching to user-centred inventory management applications improves efficiency and reduces errors, emphasising the need for custom software solutions in small businesses.

## **2.4 Importance of Real-Time Data and Scalability**

Real-time data handling and system scalability are essential considerations for creating effective inventory management systems. As Ngcobo et al. (2024) point out, Enterprise Data administration (EDM) is critical to corporate performance because it ensures efficient administration of data from several sources, allowing for real-time applications. In the context of inventory management, accessing and processing real-time data is critical for controlling inventory levels, managing equipment loans, and reducing stock handling errors. The report also emphasises the significance of scalable cloud platforms (such as AWS, Azure, and Google Cloud) for handling increasing data volumes while maintaining system performance. These solutions provide flexibility, letting businesses expand their data storage and processing capacities as needed, which is critical to supporting the dynamic nature of inventory systems.

## **2.5 Data Visualisation and Reporting**

Data visualisation plays a pivotal role in enhancing decision-making processes by transforming complex data into actionable insights. As highlighted by the study on key performance indicators (KPIs) for inventory management, live dashboards are instrumental in providing stakeholders with real-time insights into the supply chain performance and facilitating informed decision-making. The research emphasizes the use of advanced technologies, including Hadoop and Spark, to process and visualise large datasets, making it easier for decision-makers to track metrics such as stock availability and estimated delivery times. By integrating these KPIs into interactive dashboards, businesses can optimise their inventory management systems and ensure timely decisions, ultimately leading to improved business performance and efficient resource allocation.

## **2.6 Conclusion**

The literature review highlights the advancements in cloud-based inventory management systems, particularly for small businesses. Cloud platforms like Firebase and MySQL enable real-time data synchronisation, scalability, and flexibility, essential for managing changing inventory needs. Integrating live dashboards and forecasting models enhances decision-making, optimises systems, reduces costs, and boosts efficiency. However, challenges like inconsistent tracking and manual processes remain, emphasising the need for automated solutions. As scalable cloud platforms and real-time data processing evolve, they will play a critical role in ensuring businesses stay agile. Data visualization tools further support decision-making, improving business outcomes. This review provides a solid foundation for developing a cloud-based inventory system that addresses current challenges and incorporates technological advancements.

# **Requirements**

## **Functional Requirements**

**FR1: User Management**

* FR1.1: Users must be able to register with unique email addresses and secure passwords.
  + Details: Password will be hashed using industry-standard encryption to ensure data security.
* FR1.2: Users must be able to login and out securely.
  + Details: The system will implement session management using JSON Web Tokens (JWT).
* FR1.3: The system must support multiple user roles:
  + Admin: Full control over users and inventory.
  + Manager: Control over industry but limited user management.
  + Staff: Can only view inventory.

**FR2: Inventory Management**

* FR2.1: Users must be able to add new products, including name, category, price, and initial stock level.
  + Details: Unique product IDs will be auto-generated.
* FR2.2: Users must be able to update product details, including stock levels.
  + Details: Updates will be reflected in real-time across all user sessions.
* FR2.3: Users must be able to delete products.
  + Details: Deleted products will be archived for reporting and auditing purposes.

**FR3: Real-Time Updates**

* FR3.1: All inventory changes should reflect immediately across all devices.
  + Details: Using Firebase Firestore’s real-time sync capabilities.

**FR4: Reporting and Analysis**

* FR4.1: The system must generate reports on stock levels, sales trends, and low-stock alerts.
  + Details: Interactive charts will be created using Chart.js.
* FR4.2: Reports must be exportable in CSV format.

**FR5: Notifications**

* FR5.1: Users must receive alerts when stock falls below a predefined threshold.
  + Details: Notifications will be displayed on the dashboard or sent via email.

The following diagram outlines the key interactions between the user roles and the system’s primary functionalities.

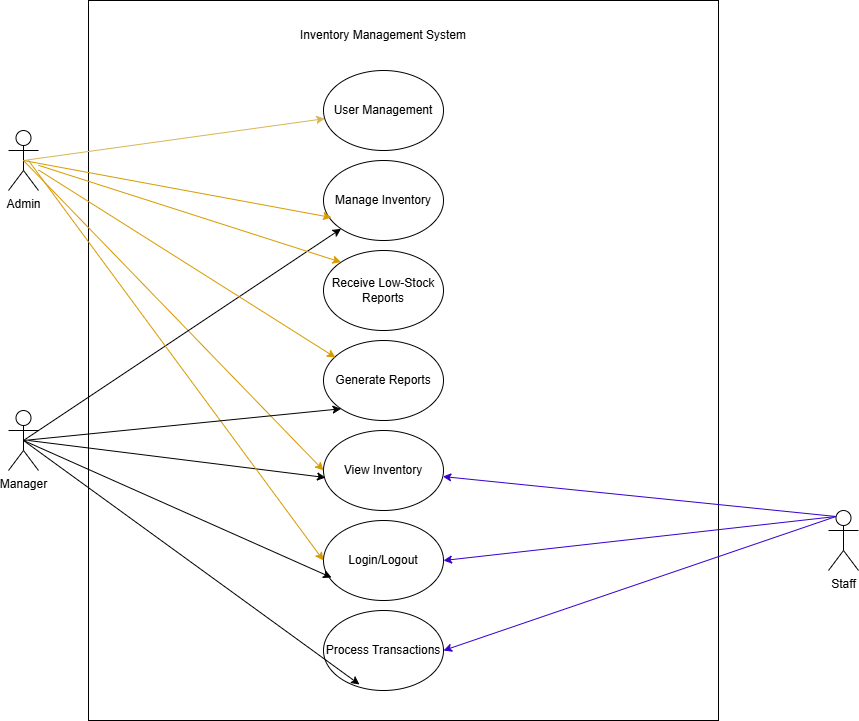


Figure 1 Use Case Diagram illustrating interactions between user roles and the system functionalities.

## **Non-Functional Requirements**

**NFR1: Scalability**

* The system must handle up to 1000 concurrent users without degradation.
  + Details: This will be achieved by leveraging cloud-based infrastructure (e.g. Firebase, Heroku) with autoscaling capabilities.

**NFR2: Performance**

* Real-time updates must propagate to all users within 2 seconds of a change.
  + Details: This will be tested using load testing tools to ensure the system can handle many real-time operations.

**NFR3: Security**

* The system must use HTTPS to ensure secure communication.
  + Details: All sensitive data, including passwords and JWT’s, must be encrypted.

**NFR4: Usability**

* The system should have an intuitive user interface accessible to non-technical users.
  + Details: A user-centric design will be followed, with feedback loops from non-technical users to ensure ease of use.

**NRF5: Availability**

* The system should have 99.9% uptime, supported by cloud-hosted deployment.
  + Details: Cloud-based services (Heroku, Firebase) will be used to ensure redundancy and failover capabilities.

## **3.3 Hardware Requirements**

* **Client-Side:**

Any device with a modern web browser (desktop, tablet, or mobile)

* **Server-Side:**

Cloud-Based deployment using platforms like Firebase

## **3.4 Software Requirements**

**Frontend Technologies:**

* HTML5, CSS, JavaScript, Bootstrap: To structure and style the user interface
* React: To build dynamic, responsive, and modular UI component
* Chart.js: For data visualisation, including inventory and sales trends.

**Backend Technologies:**

* Python with Flask: To handle business logic, data processing and API creation.
* Flask-SocketIO: For real time updates and notifications.

**Database:**

* Firebase Firestore: To store and manage inventory data in real-time.

**Other Tools and Libraries:**

* Axios: To handle API requests between the frontend and the backend.
* PyJWT: For implementing secure token-based authentication.

# **Outline of Specification and Design**

## **System Architecture**

The system employs a three-tier architecture:

* Frontend (React): User interface for inventory and reporting functionalities.
* Backend (Flask): REST API handles authentication, data processing, and CRUD operations.
* Database (Firebase Firestore): Cloud-hosted NoSQL database enabling real-time updates.

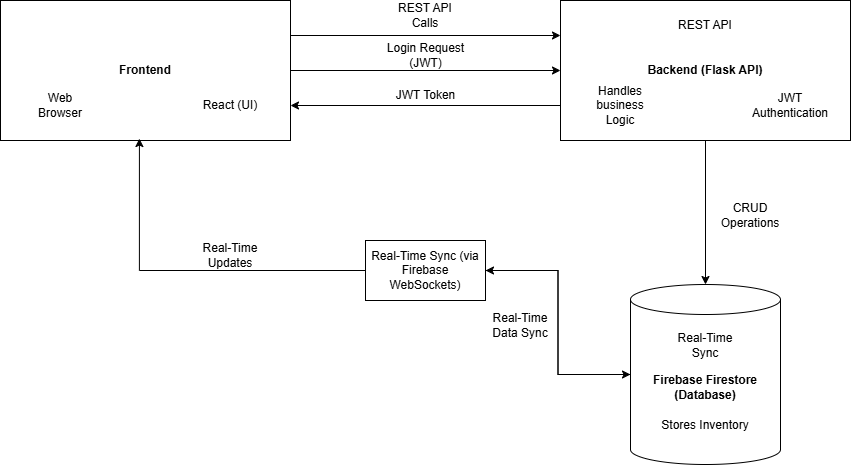


Figure 2 System architecture, showing the interactions between the frontend, backend, database, and real-time sync mechanisms.

The diagram above illustrates the components and data flow between frontend, backend and database. It highlights interactions between the UI (React), the server-side logic (Flask API with Flask-SocketIO), and the database (Firebase Firestore), ensuring real-time updates and secure data handling.

### **Frontend**

**Technologies used:** React, Bootstrap, Chart.js

**Main Functions:**

* User Interface: Provide interactive and responsive pages for inventory management.
* Role-Based Views: Displays specific UI components based on user roles (Admin, Manager, or Staff).
* Real-Time Updates: Ensure data synchronisation across all user sessions using WebSockets or Firebase’s real-time features.
* Data Visualisation: Graphically presents trends and stock levels for quick decision-making.

### **Backend**

**Technologies Used:** Python, Flask, Flask-SocketIO

**Main Functions:**

* Restful API: Facilitates communication between the frontend and the database. Handles CRUD operations for inventory management.
* Authentication & Authorisation: Implements secure login using JWTs and RBAC to ensure users have access only to authorise features.
* Real-Time Sync: Utilise Flask-SocketIO for real-time notifications and updates.
* Data Aggregation & Reporting: Process data for generating custom reports based on inventory, sales, and low-stock alerts.

### **Database**

**Technologies Used:** Firebase Firestore

**Main Functions:**

* Data Storage: Story inventory, user, and transaction data in a structured yet flexible manner.
* Real-time Updates: Automatically syncs data across all devices using Firestore’s real time capabilities.
* Data Security: Enforces access rules based on user roles, ensuring sensitive data is protected.

## **Important Algorithms and Data Structures**

### **4.2.1 Role-Based Access Control (RBAC) Algorithm**

* Purpose: Manages User permissions based on roles
* How it works:

1. Assigns roles during user registrations or by an Admin.
2. Checks user roles when accessing specific functionalities.
3. Denies or grants access based on predefined permissions.

### **4.2.2 Real-time sync using Firebase**

* Purpose: Provides real-time updates for inventory changes
* How it works:

1. Uses Firebase listeners to detect changes in the Firebase database.
2. Propagates changes instantly to all connected clients.

### **4.2.3 Low-Stock Notification Algorithm**

* Purpose: Alerts users when stock levels fall below a threshold
* How it works:

1. Continuously monitors stock levels.
2. Triggers a notification when any product’s stock levels below its predefined threshold.
3. Sends real-time alerts using WebSockets or Firebase notifications.

### **4.2.4 Data Aggregation for Reporting**

* Purpose: Generates summary reports on inventory and sales trends.
* How it works:

1. Aggregates historical data from Firestore.
2. Processes data using Python Pandas library or similar tools.
3. Produces visual reports via Chart.js, including bar charts, pie charts, and line graphs.

### **4.2.5 Data Structure for Inventory Management**

* Purpose: Organise product data for efficient access and updates.
* Data structure used:
  + Hash Map
    - Key: Product ID
    - Value: Product details (name, category, price, stock level)
  + Lists: For sequential storage of transaction history or archived products.

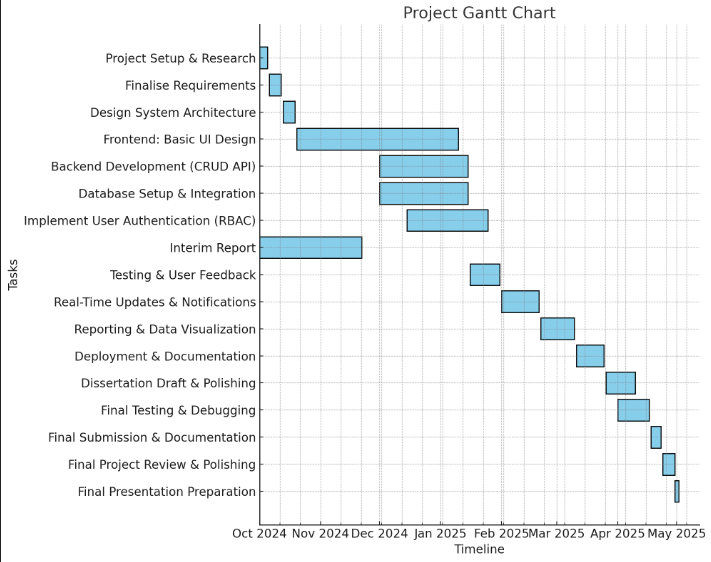
# **Planning and Timescales**

## **Semester 1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Start Date** | **End Date** | **Status** | **Milestones** |
| Project Setup & Research | October 1, 2024 | October 5, 2024 | ✅ Completed | Project Kick-off |
| Finalise Requirements & System Objectives | October 6, 2024 | October 12, 2024 | ✅ Completed | Requirements Defined |
| Design System Architecture | October 13, 2024 | October 19, 2024 | ✅ Completed | System Architecture Draft Completed |
| Frontend: Basic UI Design (Inventory Pages) | |  | | --- | | October 20, 2024 |  |  | | --- | |  | | December 15, 2024 | 🔄 In Progress | Core Inventory Functional |
| Backend Development: CRUD API | December 1, 2024 | |  | | --- | | January 15, 2025 |  |  | | --- | |  | | ⬜ Not Started | CRUD Functionality Available |
| Database Setup & Integration | December 1, 2024 | January 15, 2025 | ⬜ Not Started | Database Integrated |
| Implement User Authentication (RBAC) | December 15, 2024 | January 20, 2024 | ⬜ Not Started | Secure Authentication Working |
| **Interim Report Submission** | October 1, 2024 | November 22, 2024 | 🔄 In Progress | Interim Report Completed and Submitted by Nov 22 |
| Testing & User Feedback | January 16, 2025 | January 31, 2025 | ⬜ Not Started | - |

## **Semester 2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Start Date** | **End Date** | **Status** | **Milestones** |
| Real-Time Updates & Notifications | February 1, 2025 | February 20, 2025 | ⬜ Not Started | Live Updates Integrated |
| Reporting & Data Visualization | February 21, 2025 | March 10, 2025 | ⬜ Not Started | Interactive Reports Working |
| Deployment & Documentation | March 11, 2025 | March 25, 2025 | ⬜ Not Started | - |
| Dissertation Draft & Polishing | March 26, 2025 | April 10, 2025 | ⬜ Not Started | - |
| Final Testing, Debugging & User Feedback | April 1, 2025 | April 17, 2025 | ⬜ Not Started | - |
| Final Submission & Documentation | April 18, 2025 | April 23, 2025 | ⬜ Not Started | - |
| Final Project Review & Polishing | April 24, 2025 | April 30, 2025 | ⬜ Not Started | - |
| Final Presentation Preparation | April 30, 2025 | May 2, 2025 | ⬜ Not Started | - |



# **References**

Chukwumuanya, O. E., Onwurah, U. O., and Ihueze, C. C., "Development of a Web-Based Inventory Management System for Small Businesses," International Journal of Industrial and Production Engineering (IJIPE), vol. 2, no. 2, 2024. Available: <https://journals.unizik.edu.ng/ijipe/article/view/4096/3403>

Shopify, "What Is Cloud-Based Inventory Management? A Complete Guide," *Shopify*, 2024.

Available: <https://www.shopify.com/retail/cloud-based-inventory-management>

Firebase. (n.d.). *Cloud Firestore: Flexible, Scalable NoSQL Cloud Database*. Retrieved from: <https://firebase.google.com/docs/firestore>

Linnworks. (2024, March 20). *Forget desktop: Here’s why you should choose a cloud-based inventory management system*. Retrieved from <https://www.linnworks.com/blog/cloud-based-inventory-management-system/>

A. Jenkins, “20 Inventory Management Challenges and Solutions for 2022 and Beyond,” NetSuite, Feb. 4, 2022. [Online]. Available: <https://www.netsuite.com/portal/resource/articles/inventory-management/inventory-management-challenges.shtml>  
  
S. Sinisalo, *User-centric development of an efficient Inventory management system for a university lab*, Theseus.fi, 2024. [Online]. Available: <https://www.theseus.fi/handle/10024/863803> (May need to translate page from Finish to English)  
  
K. S. Ngcobo, S. Bhengu, A. B. D. Mudau, B. Thango, and L. Matshaka, "Enterprise Data Management: Types, Sources, and Real-Time Applications to Enhance Business Performance - A Systematic Review," *Preprints*, Sep. 2024. [Online]. Available: <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4968451>  
  
"A Survey Paper on Identifying Key Performance Indicators for Optimizing Inventory Management System and Exploring Different Visualization Tools," *SSRN*, [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/9121036>