|  |
| --- |
| 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](#_Toc183112096)

[**1.** **Aims and Objectives** 1](#_Toc183112097)

[**1.1 Aims** 1](#_Toc183112098)

[**1.2 Objectives** 1](#_Toc183112099)

[**1.3 Challenges and Originality** 2](#_Toc183112100)

[**1.3.1 Challenges:** 2](#_Toc183112101)

[**1.3.2 Originality:** 2](#_Toc183112102)

[**2.** **Survey of Literature/ Information Sources** 2](#_Toc183112103)

[**2.1 Overview of Cloud-Based Solutions and Inventory Management** 2](#_Toc183112104)

[**2.3 Challenges with Current Inventory Management Systems** 3](#_Toc183112105)

[**2.4 Importance of Real-Time Data and Scalability** 3](#_Toc183112106)

[**2.5 Data Visualisation and Reporting** 4](#_Toc183112107)

[**2.6 Conclusion** 4](#_Toc183112108)

[**3.** **Requirements** 4](#_Toc183112109)

[**3.1** **Functional Requirements** 4](#_Toc183112110)

[**3.2** **Non-Functional Requirements** 6](#_Toc183112111)

[**3.3 Hardware Requirements** 7](#_Toc183112112)

[**3.4 Software Requirements** 7](#_Toc183112113)

[**4.** **Outline of Specification and Design** 8](#_Toc183112114)

[**4.1** **System Architecture** 8](#_Toc183112115)

[**4.1.1** **Frontend** 8](#_Toc183112116)

[**4.1.2** **Backend** 9](#_Toc183112117)

[**4.1.3** **Database** 9](#_Toc183112118)

[**4.2** **Important Algorithms and Data Structures** 9](#_Toc183112119)

[**4.2.1 Role-Based Access Control (RBAC) Algorithm** 9](#_Toc183112120)

[**4.2.2 Real-time sync using Firebase** 10](#_Toc183112121)

[**4.2.3 Low-Stock Notification Algorithm** 10](#_Toc183112122)

[**4.2.4 Data Aggregation for Reporting** 10](#_Toc183112123)

[**4.2.5 Data Structure for Inventory Management** 10](#_Toc183112124)

[**5.** **Planning and Timescales** 10](#_Toc183112125)

[**5.1** **Semester 1** 10](#_Toc183112126)

[**5.2** **Semester 2** 11](#_Toc183112127)

[**References** 12](#_Toc183112128)

# **Declaration**

A close-up of a text

Description automatically generated

Arif Choudhury



# **Aims and Objectives**

## **1.1 Aims**

The primary goal of this project is to design and execute a cloud-based inventory management system that specifically addresses the operational inconsistencies experienced by small businesses. By offering real-time insights and optimised inventory management, this project aspires to reduce stock variances, decrease operational delays, and enable business owners to make decisions grounded in data. The system will improve business agility, allowing owners to respond proactively to market needs while keeping stock levels optimal.

The key areas include:

* Making inventory tracking easier for non-technical users through a straightforward interface.
* Using cloud technology to ensure secure, real-time data synchronisation and remote access.
* Providing scalability to accommodate business growth with larger inventories and more users.
* Introducing flexible reporting tools that adjust to changing business demands, supporting strategic planning.

## **1.2 Objectives**

The project's objectives are established to guarantee that the aims are achieved in a quantifiable and practical way:

* Implement a secure, role-based user authentication system (RBAC): Enable separate access for Admins, Managers, and Staff, guaranteeing secure and restricted entry to system functionalities.
* Create an intuitive, responsive user interface: Utilise React and Bootstrap to develop a smooth experience, reducing onboarding duration and improving efficiency for non-technical users.
* Allow for real-time inventory updates: Use Firebase to instantly synchronise stock levels, ensuring consistent and accurate information across various devices.
* Integrate advanced data visualization tools: Use Chart.js to create interactive dashboards that display inventory trends, sales performance, and low-stock notifications for improved decision-making.
* Deploy a scalable, cloud-based architecture: Ensure the system can handle increased usage and larger datasets through platforms like Firebase.
* Develop customisable and exportable reporting options: Enable users to generate reports in formats like CSV, personalised to their unique business requirements.

## **1.3 Challenges and Originality**

### **1.3.1 Challenges:**

Creating this project involves a variety of technical and design challenges, including:

* A major obstacle in this project is my lack of experience with Flask, React, and Firebase which I have only just touched upon. While I have a solid background in Python, HTML & CSS, I need to devote more time to understanding these new technologies. I've set aside time in my agenda for organised tutorials to make certain I can successfully incorporate them into the project.
* Concurrency in Real-Time Data Management: Synchronising inventory updates across multiple devices without compromising performance necessitates strong database operations and optimised real-time syncing using Firebase. Managing concurrent user actions presents an important challenge, requiring careful system architecture to avoid conflicts.
* User Experience Design for Non-Technical Users: Striking a balance between simplicity and functionality is a challenge. The system needs to remain user-friendly while incorporating sophisticated features like role-based views and interactive dashboards.
* Data Security and Privacy in a Cloud Environment: Protecting sensitive business information demands secure authentication methods (JWTs) and encrypted communication (HTTPS). Maintaining a balance between strong security measures and system performance is an important concern.

### **1.3.2 Originality:**

This project presents a unique combination of real-time synchronisation, role-based access control, and interactive reporting personalised specifically for small businesses. Unlike existing commercial systems that tend to be feature-heavy and expensive, this solution prioritises simplicity and affordability. By incorporating advanced data visualisation tools, the system not only tracks inventory but also offers actionable insights, allowing businesses to dynamically improve their stock management strategies. The implementation of Firebase guarantees scalability while reducing infrastructure costs, distinguishing this system from conventional inventory management platforms.

# **Survey of Literature/ Information Sources**

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

When creating a cloud-based inventory management system for small businesses, important insights from Chukwumuanya et al. (2024) and Shopify's guide on cloud inventory management are very helpful. Chukwumuanya et al. (2024) underline utilising a MySQL-based web platform to store inventory data and employ forecasting models like ARMA for improved inventory management. Shopify stresses the importance of real-time updates, demand forecasting, and multi-location management, all of which are critical for contemporary inventory systems. My solution will merge these approaches to deliver real-time tracking, automated ordering, and forecasting features, aiming to simplify inventory processes, reduce costs, and enhance decision-making for small businesses.  
  
**2.2 Cloud-Based Inventory Systems' Advantages**Cloud-based inventory management systems have transformed the way businesses manage their operations, offering major advantages over traditional desktop systems. Firebase's Cloud Firestore enhances scalability and real-time synchronisation, enabling companies to quickly track inventory changes across multiple devices, ensuring accurate and up-to-date information. This flexibility is critical for addressing fluctuating inventory needs. Unlike desktop solutions that are restricted by physical hardware and manual updates, cloud systems, such as those powered by Firestore, offer smooth integrations, automatic updates, and strong security features like data encryption and disaster recovery (Firebase, n.d.; Linnworks, 2024). The capability to manage inventories remotely, coupled with advanced querying abilities, boosts operational agility and efficiency, particularly for businesses facing variable demand or seasonal fluctuations.

**2.3 Challenges with Current Inventory Management Systems**  
Inventory management encounters issues such as inconsistent tracking, inefficiencies in warehouses, and manual errors, leading to data inaccuracies and operational holdups. Changing demand can result in overstocking or stockouts, which impacts cash flow and customer satisfaction. Fragmented supply chains and ineffective communication further complicate decision-making. These challenges emphasize the necessity for automated solutions that deliver real-time data and enhance efficiency (NetSuite, 2022).

Due to the high expense of specialised software, small businesses frequently resort to error-prone spreadsheet systems. Research indicates that transitioning to user-centred inventory management applications enhances efficiency and decreases errors, emphasising the demand for customised software solutions in small businesses.

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

Handling real-time data and ensuring system scalability are critical factors in designing effective inventory management systems. As Ngcobo et al. (2024) note, Enterprise Data Management (EDM) is essential for organisational performance since it guarantees the efficient management of data from various sources, assisting real-time applications. In inventory management, gaining access to and processing real-time data is essential for maintaining inventory levels, overseeing equipment loans, and minimising stock handling mistakes. The report further emphasises the importance of scalable cloud platforms (such as AWS, Azure, and Google Cloud) for managing increasing data volumes while sustaining system performance. These solutions offer flexibility, allowing businesses to expand their data storage and processing capabilities as necessary, which is critical for supporting the dynamic nature of inventory systems.

## **2.5 Data Visualisation and Reporting**

Data visualisation is essential in enhancing decision-making processes by converting complex data into actionable insights. As indicated by the research on key performance indicators (KPIs) for inventory management, real-time dashboards are essential in supplying stakeholders with timely insights into supply chain performance, assisting knowledgeable decision-making. The study emphasises the adoption of advanced technologies, such as Hadoop and Spark, to process and visualise large datasets, making it simpler for decision-makers to monitor metrics like stock availability and estimated delivery times. By incorporating these KPIs into interactive dashboards, organisations can optimise their inventory management systems and ensure timely decisions, finally resulting in improved business performance and effective resource allocation.

## **2.6 Conclusion**

This literature review emphasises the progress in cloud-based inventory management systems, especially for small businesses. Cloud platforms like Firebase and MySQL enable real-time data synchronisation, scalability, and flexibility, which are essential for addressing changing inventory requirements. The integration of real-time dashboards and forecasting models enhances decision-making, optimises systems, lowers costs, and increases efficiency. However, challenges such as inconsistent tracking and manual processes persist, stressing the necessity for automated solutions. As scalable cloud platforms and real-time data processing continue to develop, they will be important in making sure that businesses remain adaptable. Data visualisation tools also support decision-making, leading to better business results. This review establishes a strong foundation for building a cloud-based inventory system that tackles existing challenges and integrates 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.

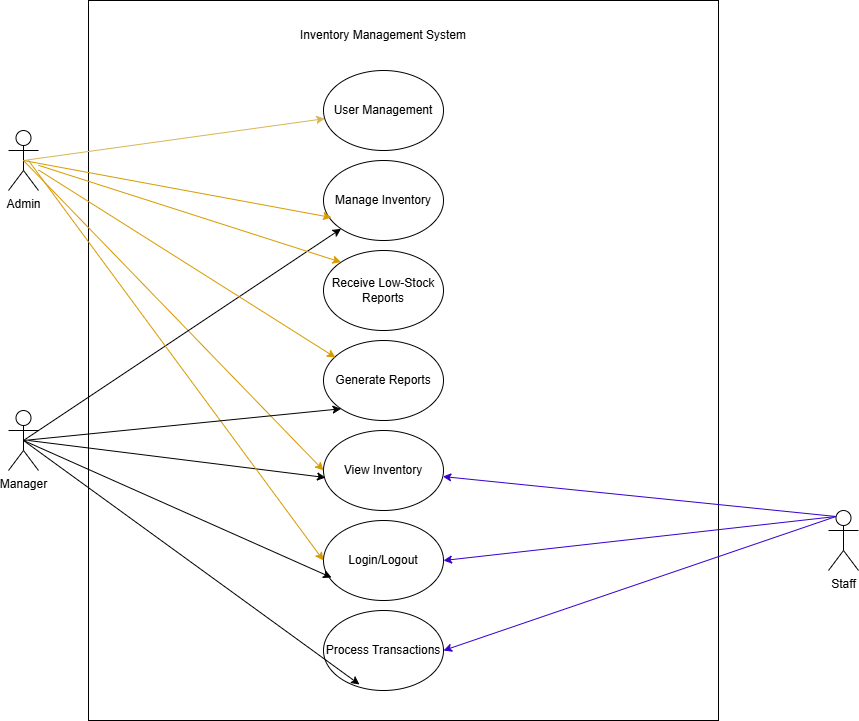


Figure 1 Use Case Diagram illustrating interactions between user roles and the system functionalities.

## **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) 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 (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 architecture of the system is designed in three layers:

* Frontend (React): A user interface that assists inventory management and reporting functionalities
* Backend (Flask): A REST API that manages authentication, data processing, and CRUD operations.
* Database (Firebase Firestore): A cloud-hosted NoSQL database that supports real-time updates.

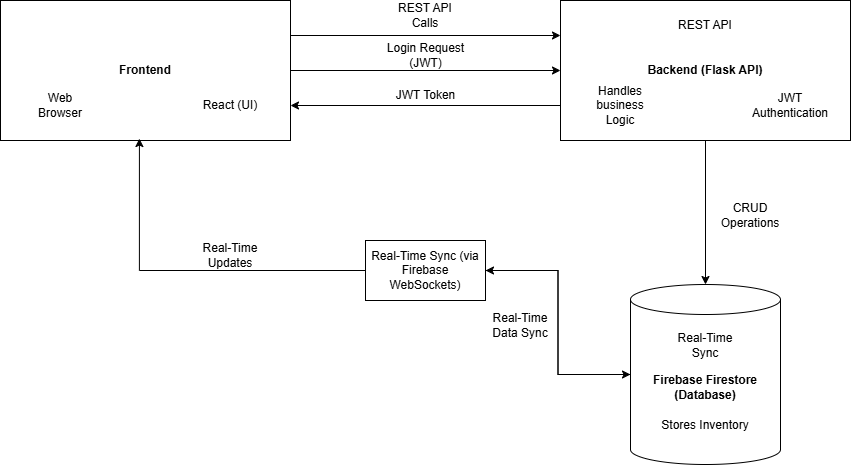


Figure 2 System architecture, showing the interactions between the frontend, backend, database, and real-time sync mechanisms.

The above diagram outlines the various components and the flow of data among the frontend, backend, and database. It demonstrates the interactions between the user interface (React), server-side logic (Flask API with Flask-SocketIO), and the database (Firebase Firestore), which enables real-time updates and secure data management.

### **Frontend**

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

**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**

* 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**

* 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**

* 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**

* 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**

* 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**

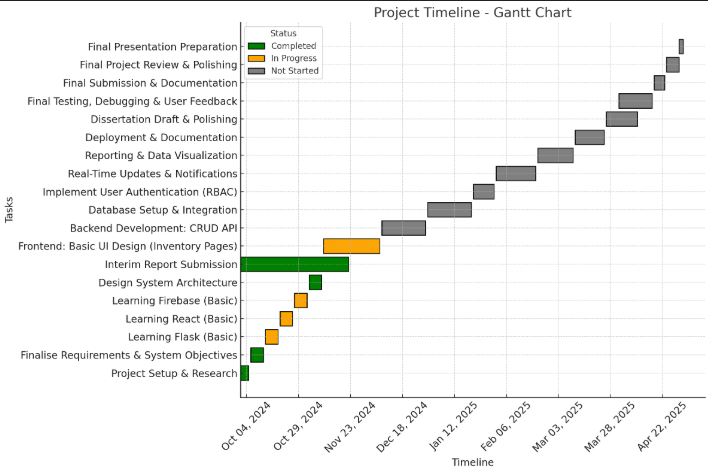
## **Semester 1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |
| Learning Flask (Basic) | October 13, 2024 | October 19, 2024 | 🔄 In Progress | Basic Flask Concepts Grasped |
| Learning React (Basic) | October 20, 2024 | October 26, 2024 | 🔄 In Progress | Basic React Concepts Grasped |
| Learning Firebase (Basic) | October 27, 2024 | November 2, 2024 | 🔄 In Progress | Basic Firebase Concepts Grasped |
| Design System Architecture | November 3, 2024 | November 9, 2024 | ✅ Completed | System Architecture Draft Completed |
| **Interim Report Submission** | October 1, 2024 | November 22, 2024 | ✅ Completed | Interim Report Completed and Submitted by Nov 22 |
| Frontend: Basic UI Design (Inventory Pages) | |  | | --- | | November 10, 2024 |  |  | | --- | |  | | December 7, 2024 | 🔄 In Progress | Core Inventory Functional |
| Backend Development: CRUD API | December 8, 2024 | |  | | --- | | December 29, 2024 |  |  | | --- | |  | | ⬜ Not Started | CRUD Functionality Available |
| Database Setup & Integration | December 30, 2024 | January 20, 2024 | ⬜ Not Started | Database Integrated |
| Implement User Authentication (RBAC) | January 21, 2025 | January 31, 2025 | ⬜ Not Started | Secure Authentication Working |

I plan to allocate 3-4 weeks to learning the fundamentals of Flask, React, and Firebase. Throughout this period, I will concentrate on backend APIs, frontend UI, and database integration, using these skills to design and implement important elements directly related to my project.

## **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>