APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 6 No. 1 (2025) 790-807
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Grocery Inventory Management System for Kedai Buah-buahan Misri Sumiran

Nabila Syafiqah Mohd Hanafi[1], Hannani Aman[1*]

[1] *Faculty of Computer Science and Information Technology*
Universiti Tun Hussein Onn Malaysia, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: hanani@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.01.044

Article Info

Received: 13 June 2024
Accepted: 18 June 2025
Available online: 30 June 2025

Keywords

Grocery, Inventory Management System, Object-Oriented Approach.

Abstract

This project centres on the imperative need for efficient inventory management within the neighborhood grocery store, Kedai Buah-buahan Misri Sumiran. The Grocery Inventory Management System seeks to address these issues of inaccurate records, overlooked expiration dates, and disorganized supplier details by providing a modernized and streamlined solution. Development of this system utilizes the Waterfall Model methodology as the requirements are all stated clearly in the planning phase. The Grocery Inventory Management System for Kedai Buah-buahan Misri Sumiran involves the development of a web-based system with modules for registration, product management, inventory tracking, supplier management, and reminders for impending expirations and low stocks. Low stock are determined by the stock percentage of each prodcut that reach 20% and below. The stock percentage is calculated by the maximum stock level against the current stock. Anticipated outcomes include increased efficiency in stock management, accurate recording of inventory items, and timely reminders for expiration dates and low stocks. The significance of the project lies in enhancing operational efficiency, minimizing waste, and facilitating informed decision-making for both the store owner and employees. Developing the mobile app version of this web-based system would be the future wish of this project.

1. Introduction

For grocery stores to succeed, they must meet customer needs and maintain cost control through effective inventory management. The local grocery store Kedai Buah-buahan Misri Sumiran currently uses manual inventory management, which could lead to problems including inaccurate data, expired goods on the shelf, and issues contacting suppliers. The Grocery Inventory Management System aims to improve overall efficiency and streamline procedures in order to tackle these issues. By reducing the possibility of mistakes and waste, this system makes it easier to quickly and accurately record product information, including important details like expiration dates. The implementation of a reminder feature will help to increase consumer satisfaction by preventing expired goods from being displayed on store shelves. Furthermore, the system would manage supplier contact details, reducing the possibility of data loss and guaranteeing smooth communication for upcoming product orders. The installation of this advanced inventory management system is expected to improve Kedai Buah-buahan Misri Sumiran's operational effectiveness, guaranteeing the store's long-term success and financial gains.

The article is organized into six chapters with each covering different aspect of the project. The first chapter introduces the case study while outlining its scope, expected outcomes and significance. The second chapter

highlights the literature review of the project and reviews three similar systems and their usability assessment techniques. Chapter three details the methodology used in the project. Next, chapter four covers the system requirements, analysis, and design, including UML diagrams. Chapter five focuses on the system's implementation and testing, ensuring that it meets the requirements and user satisfaction. Finally, chapter six evaluates the developed system, offers recommendations, and assesses the research objectives.

2. Literature Review

Inventory in the Oxford Dictionary were defined as "a written list of all objects, furniture, or others in a specific place" [1]. The key to efficient inventory management is understanding what you have, where it is being used, and how much ends up as finished goods. Any business's capability to efficiently manage its inventory is critical to its success, regardless of its size or field. Effective inventory management aims to reduce inventory costs to a minimum while maintaining sufficient inventory on hand to fulfil customer demands [2]. Every inventory management system should be tailored to the company's size and preferences in order to get the right efficiency needed by the company. As stated by Munyaka and Yadavalli [3], maintaining the targeted level of customer satisfaction while minimizing the overall cost of inventory is made possible by effective inventory management.

Management Information Systems (MIS) play a critical role in both public and commercial enterprises, functioning as tools that aid managers in addressing organizational challenges [4]. Their significance extends to improving the effectiveness and efficiency of the public sector, assisting government agencies in delivering services, optimizing resource allocation, and maintaining accountability. Improving the effectiveness and efficiency of inventory management in a grocery shop requires the integration of a Management Information System (MIS) into a Grocery Inventory Management System. The grocery store inventory system includes MIS concepts, which are centred on decision support, effortlessly.

To achieve the goals of this entire project, two technologies will be used which are web-based technology and barcode technology. With the present of these technologies, this project will be developed well as they were the most crucial part of this project. Web-based technologies encompass a spectrum of network applications facilitating internet-based connectivity and interaction. Users engage with diverse platforms, including blogs, discussion boards, and online meeting tools, promoting communication and collaboration. The basic layout is established by HTML, the visual styling is refined by CSS, and dynamic elements and real-time interactions are introduced by JavaScript. These languages work together to form the framework of contemporary web apps, giving developers the ability to create dynamic and interesting user interfaces. Next, barcode in Oxford Dictionary was defined as a machine-readable code printed on a product that is specifically used for stock control. It takes the form of numbers and a pattern of parallel lines with different widths [5]. Barcodes are categorized into two types which are one-dimensional barcodes and two-dimensional barcodes.

Inventory management at Kedai Buah-buahan Misri Sumiran is currently done manually by using only a book and a pen with expiry dates not being recorded and supplier's contact details scattered around. The store owner and employee would record every supply received in a book with the date and day of when the items are received. The expiry dates of delivered goods are not recorded which can be troublesome when customers accidentally bought the expired items as it were not removed from the shelf by the employee. The flow of the current inventory management at the store is as shown in Fig. 1.

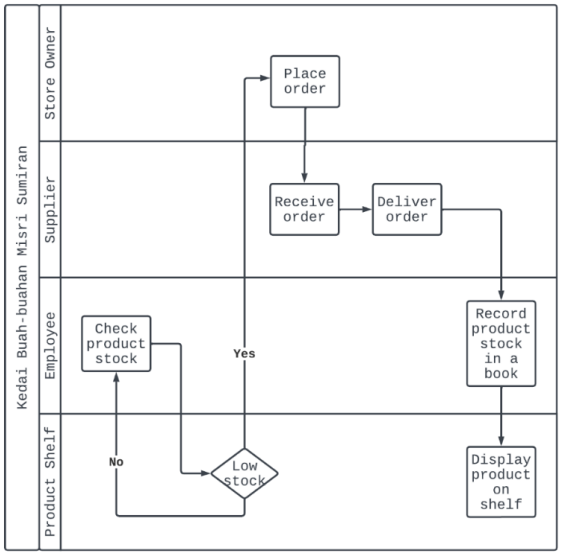

Fig. 1 *Swimlane model of current process at Kedai Buah-buahan Misri Sumiran*

Three similar inventory management systems will be examined and discussed. The following inventory management have similarities to this project's requirement specifications:

- Zoho Inventory Management System [6]
- Sortly Inventory System [7]
- Lightspeed Inventory Management [8]

Based on the comparisons done in Table 1, it can be concluded that there are some similarities and differences in these studied systems. The result that was achieved from this comparison will act as a guide in developing the Grocery Inventory Management System. The system will ease the store owner and employee in making sure the operation at the store run smoothly and flawlessly.

Various features are being compared in Table 1. There are nine features being compared with Grocery Inventory Management System for Kedai Buah-buahan Misri Sumiran from the three similar systems: Zoho Inventory, Sortly Inventory, and Lightspeed Inventory.

Table 1 *Related systems comparison*

No.	Features	Zoho Inventory	Sortly Inventory	Lightspeed Inventory	Grocery Inventory Management System for Kedai Buah-buahan Misri Sumiran
1	Accessible with Internet	Yes	Yes	Yes	Yes
2	Log in and verification	Yes	Yes	Yes	Yes
3	Manage inventory (add, view, edit, delete)	Yes	Yes	Yes	Yes
4	Low stock reminder	Yes	Yes	Yes	Yes
5	Expiry date reminder	No	No	No	Yes
6	Manage supplier's information (add, view, edit, delete)	Yes	No	Yes	Yes
7	Barcode scanning	Yes	Yes	Yes	Yes
8	Filter and search function	Yes	Yes	Yes	Yes
9	Display list of suppliers along with their products	No	No	Yes	Yes

3. Methodology

The waterfall model is a typical, linear approach to software development that moves through defined phases in a single, sequential flow. Each step of the phase must be finished before going on to the next, and it progresses gradually from one to the next. The five stages of the waterfall model are planning, analysis, design, implementation, and testing. Due to the effectiveness of this approach, numerous industrial manufacturers and development firms have made it their main development framework [9].

The tasks and outputs of every phase is shown in Table 2. The system development process begins with the planning phase, where project plans, deliverables, and requirements are documented, and a project schedule is established. For the Grocery Inventory Management System for Kedai Buah-buahan Misri Sumiran, the planning phase included analysing the system, identifying stakeholders (the store owner and employee), and defining project objectives, scope, and problem statements. After gaining approval during the Title Defense, a Gantt chart was created to outline the project's timeline, spanning from October 2023 to July 2024. The analysis phase involved understanding the store's needs through interviews, identifying key requirements such as minimizing data entry errors and tracking product expiry dates, and creating UML diagrams to visualize system requirements.

In the design phase, a detailed design prototype was developed using Draw.IO and XAMPP, encompassing system architecture, database design, and user interface design. This phase ensured that the design met user requirements and laid a solid foundation for the system. The implementation phase involved coding the system using Microsoft Visual Studio Code, XAMPP, PHP, HTML, and SQL, focusing on inventory updates, product tracking, and user-friendliness. Setting up a secure database was also crucial. Finally, the testing phase involved thorough functionality and user acceptance testing to ensure the system operated correctly and met all objectives, with any issues being promptly addressed.

Table 2 *Software development activities and their task*

Phase	Task	Output
Planning	Propose the project along with the title, objectives, problem statements and the scopes.	Project proposal and Gantt chart.
Analysis	Information regarding the company operation and stakeholders are gathered and analysed.	UML diagrams: use case diagram, activity diagrams, class diagram and requirement definition.
Design	Design the system, interface and database that will be used in the system.	System architecture, user interface design and the database design.
Implementation	Develop the system	Program code and final system
Testing	Test the developed system	Test cases

4. Analysis and Design

System requirement analysis will determine that the system will meet the stakeholder's expectations. These requirements will play a crucial role in shaping the overall usefulness and quality of the system. Requirement analysis, functional requirement, non-functional requirement, user requirements, and hardware and software requirements will be discussed further.

4.1 System Requirement Analysis

System requirement analysis will determine that the system will meet the stakeholder's expectations. These requirements will play a crucial role in shaping the overall usefulness and quality of the system. Requirement analysis, functional requirement, non-functional requirement, and user requirements will be discussed in the following subsections.

There are six modules being embedded within the system which are login module, manage suppliers module, manage stock module, manage low stock product reminder module, manage expiry dates module, and generate reports module. Details on all the modules will be as depicted in Table 3.

Table 3 *System's functional requirements*

No.	Module	Functional Requirements
1.	Login	- The system shall allow users to log in using the registered username and password. - The system shall alert users for any invalid input.
2.	Manage suppliers module	- The system shall allow user to create, update, and delete supplier's information. - The system shall store the details of purchase order for every supplier.
3.	Manage stock module	- The system shall allow users to create, update, and delete product details. - The system shall display the registered products. - The system shall allow products to be added using barcode scanner. - The system shall allow user to register in an out of every item at the store. - The system shall allow user to search and filter products based on the expiry dates and suppliers.
4.	Manage low stock product reminder module	- The system shall allow user to set the maximum stock level for every product. - The system shall calculate stocks percentage based on the maximum stock and current stock, then display the stock percentage. - The system shall push reminder on low stock products.
5.	Manage expiry dates module	- The system shall allow user to add expiry date of every product. - The system shall calculate the days taken until the expiry date of every product. - The system shall display and push reminder on products that is nearing the expiry date.
6.	Generate reports module	- The system shall generate reports on the list of suppliers along with the products that they deliver. - The system shall display the list of items that will expire in the next few days, weeks, or months.

As shown in Table 4, non-functional requirements focus on system constraints, ensuring security, performance, and user satisfaction. Both are essential for effective system development and delivery.

Table 4 *System's non-functional requirements*

No.	Aspects	Non-functional Requirements
1.	Reliability	The system shall be available whenever needed by the users.
2.	Usability	The system's user interface should be simple to use and intuitive, requiring little training for new users.
3.	Security	The system shall only allow the registered username and password to have access to the system.
4.	Scalability	The system can easily adapt to meet an increase in the quantity of suppliers and products.
5.	Performance	The system shall respond to user queries quickly.

Table 5 shows the user's requirements for this system.

Table 5 *User's requirements*

No.	User Requirements
1.	Store owner and employee shall be able to log in into their account.
2.	Employee shall be able to create, update, and delete product at the store along with their details.
3.	Employee shall be able to register in and out products using barcode scanner.
4.	Store owner shall be able to set the maximum stock level for every product.
5.	Store owner and employee shall be able to search and filter products based on their ABC category, expiry dates, and the supplier of the products.
6.	Store owner shall be able to create, update, and delete supplier's information.
7.	Store owner shall be able to store details of purchase order.
8.	Store owner shall generate reports of suppliers and their products and products that will expire.

4.2 System Analysis

In this section, the general structure or flow of the system will be discussed in detail. The system's function will be presented in graphical form of use case diagram and class diagram.

There are two actors that will use the system which are the store owner and employee. Store owner associates with login module, manage suppliers module, manage low stock product reminder module, and generate reports module. Next, the employee associates with the login module, manage suppliers module, manage stock module, and manage expiry dates module. The use case diagram of this system is as shown in Fig. 3.

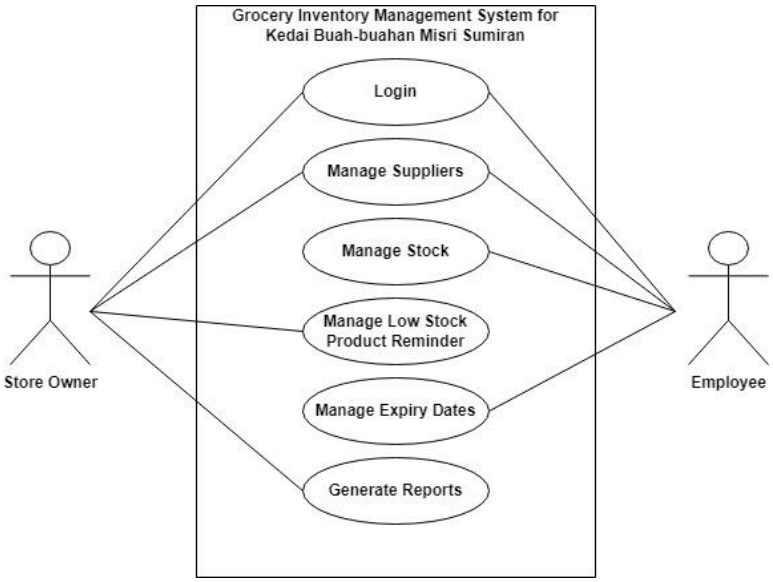

Fig. 3 *Use case diagram*

Use case specifications of login module, manage supplier module, manage stock module, manage expiry dates module, manage low stock product reminder module, and generate reports module are as depicted in Appendix A.

4.3 System Design

This section covers system design, which involves planning the system's overall structure. The architectural design creates diagrams showing connections between components, while the database design organizes data storage for effective management. Interface design focuses on user-friendly interactions. Together, these elements ensure the system is user-centered, effective, and well-organized.

Fig. 4 shows the three-tier system architecture that is being implemented in this system.

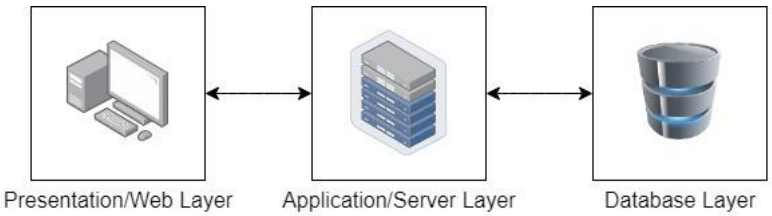

Fig. 4 *Three-tier system architecture*

Database schemas for this project are as follows:

- users (username, password, employment_type, first_name, last_name, contact_no, email)
- products (id, name, price, max_stock, quantity, expiry_date, supplier, updated_by, added_by)
- suppliers (supplier_id, driver_name, driver_contact_no, company_name, company_contact_no, company_address)
- purchaseOrders (order_id, product_id, item_name, number_of_items, date_delivered, day_delivered)

This Grocery Inventory Management System's class diagram shows classes, their characteristics, and the connections between them to visually explain the system's structure. Classes such as "Product", "User", "Stock", "Supplier", and "Purchase Order" are depicted in this diagram, highlighting attributes like product information, user data, stock percentage, supplier information, and purchase order details. Moreover, relationships are also shown in the class diagram. A thorough explanation of the system is given by the class diagram depicted in Fig. 5, which facilitates understanding of the main components and their interactions within the Grocery Inventory Management System.

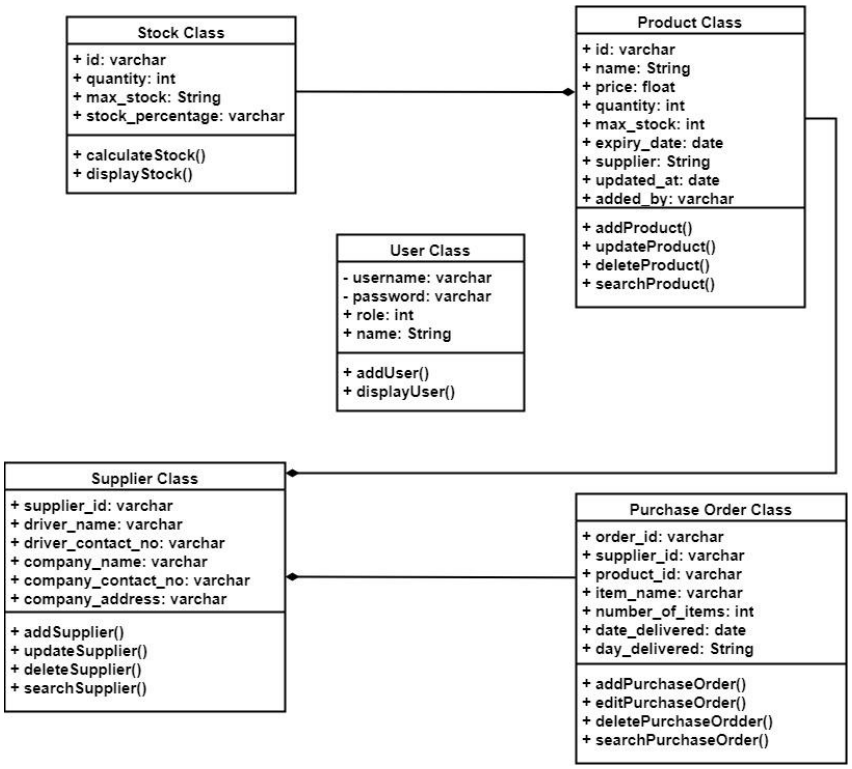

Fig. 5 *Class diagram*

5. Implementation and Testing

The final phase in the software development lifecycle (SDLC) is the implementation and testing phase. This phase is crucial as it ensures that the software is correctly deployed, and functions as intended. Thorough testing is conducted to identify and rectify any issues, guaranteeing that the final product meets all requirements and performs reliably in real-world scenarios.

5.1 Implementation

During implementation, the system was developed according to the designs in the previous phase, analysis and design. For the store owner, the functionality of the system was created based on his roles which include the login module, manage suppliers module, manage low stock product reminder module, and lastly, generate reports module. The store owner's dashboard is as shown in Appendix B. Fig.6 shows the interface for the login module.

Next, the employee will have access to some other functionalities of the system which are the login module, manage suppliers module, manage stock module, and manage expiry dates module. Employee may access to all these modules from their own dashboard which is as shown in Appendix B.

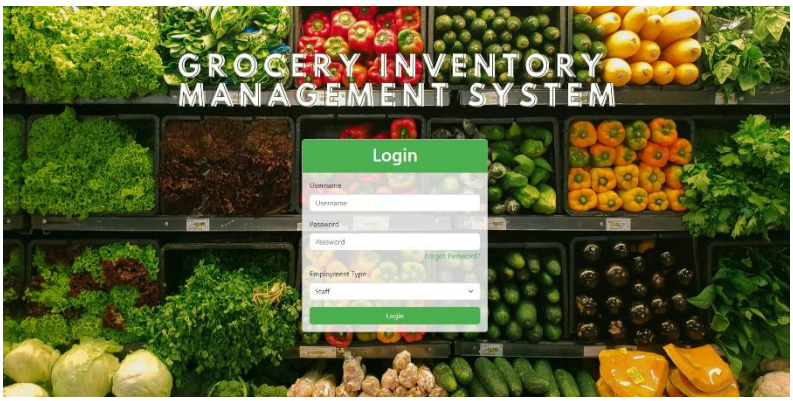

Fig. 6 *Login module interface*

Then, the interface for the manage suppliers module will be as shown in Fig. 7. This module allows the store owner to add, edit and delete supplier. Moreover, employee also can add purchase order details within this module.

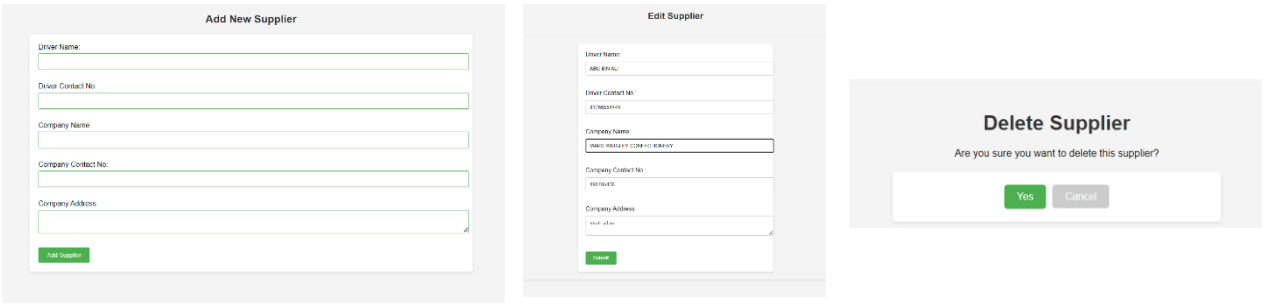

Fig. 7 *Manage suppliers module interfaces*

Fig. 8 shows the interfaces for the manage stock module. In this module, employee able to add, edit, and also delete product. Additionally, to maintain real-time stock, user can register product in and out of the system.

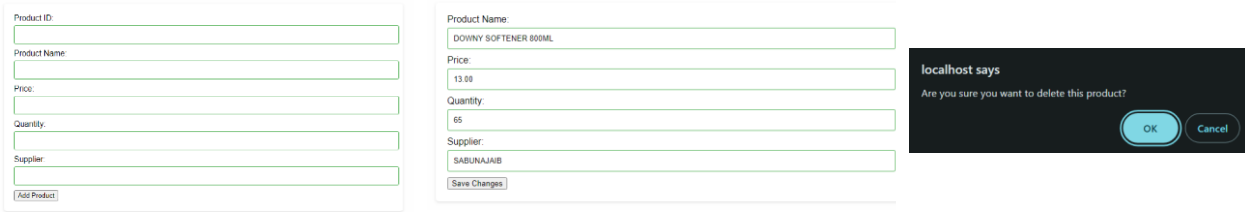

Fig. 8 *Manage stock module interfaces*

The fourth module in this system is manage low stock product reminder module. This module provides functionality to the store owner to set the maximum stock level for every product at the store. Maximum stock level is defined based on the space that can fit for any particular product. Then, the stock percentage of every product is calculated based on the maximum stock level against the current stock. When the stock percentage

reach 20% and below, the system will push a low stock reminder to both the store owner and employee. Fig. 9 and Fig. 10 shows the interfaces and the code segment of manage low stock product reminder module.

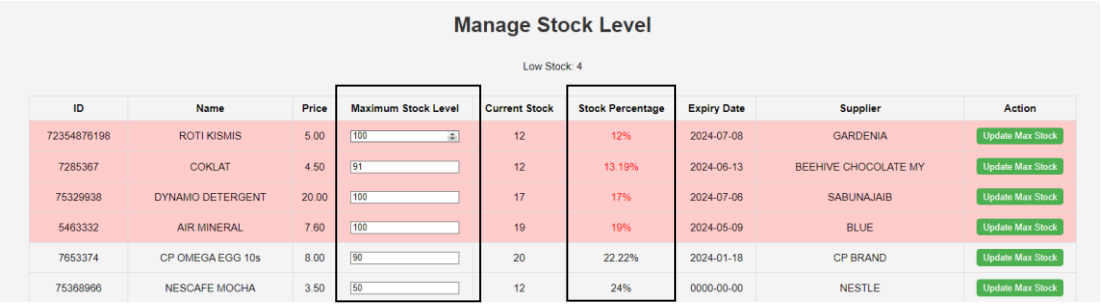

Fig. 9 *Manage low stock product reminder module interface*

```
<script>
    function updateMaxStock(productId, maxStockInput) {
        var newMaxStock = maxStockInput.value;
        var xhr = new XMLHttpRequest();
        xhr.open("POST", "update-stock.php", true);
        xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
        xhr.onreadystatechange = function () {
            if (xhr.readyState == 4 && xhr.status == 200) {
                alert("Maximum stock level updated successfully!");
                location.reload();
            }
        };
        xhr.send("product_id=" + productId + "&max_stock=" + newMaxStock);
    }
</script>
```

Fig. 10 *Manage low stock product module code segment*

The manage expiry dates module provides functionality to the employee to add and edit expiry dates. Additionally, this module will push reminder on nearing expiry dates within the next 7 days, 14 days, and 30 days. Fig. 11 and Fig. 12 shows the interface and code segment of this module. The expiry dates reminder functionality calculates the current date and then determines which expiry dates fall within the upcoming 7, 14, and 30-day periods. By comparing these calculated dates with the stored expiry dates, the system can effectively trigger reminders for the user.

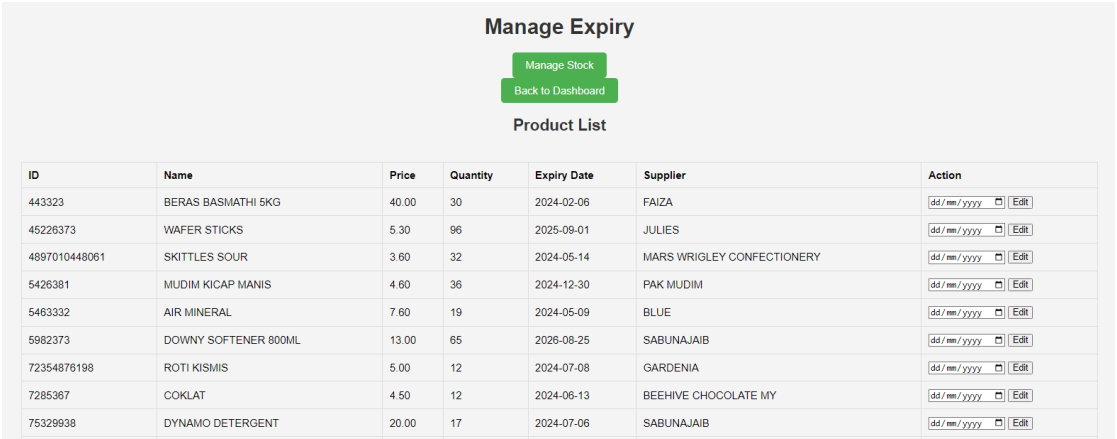

Fig. 11 *Manage expiry dates module interface*

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $expiryDate = $_POST["expiryDate"];
    $productId = $_POST["productId"];

    if (strtotime($expiryDate) < strtotime('today')) {
        $feedbackMessage = "Error: Please choose a valid date.";

        echo '<script>alert("' . $feedbackMessage . '");</script>';
    } else {
        $updateSql = "UPDATE products SET expiry_date = ? WHERE id = ?";
        $stmt = $conn->prepare($updateSql);
        $stmt->bind_param("si", $expiryDate, $productId);
        $stmt->execute();
        $stmt->close();

        header("Location: manage-expiry.php");
        exit();
    }
}
```

Fig. 12 *Manage expiry dates module code segment*

The last module of this grocery inventory management system is the generate reports module. This module allows the store owner to generate reports on suppliers and the products that they deliver to the store, and reports on products expiring within a date range set. Fig. 13 shows the module's interface.

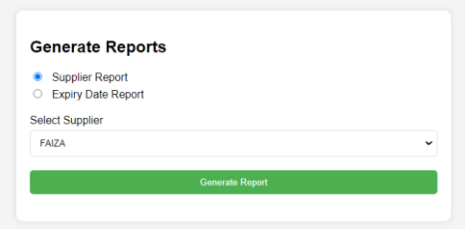

Fig. 13 *Generate reports module interface*

5.2 Testing

The testing phase begins once the system prototype has been completed. The type of testing that needs to be done is the User Acceptance Testing (UAT). UAT is employed to verify that the system meets the needs and requirements of the end users. The UAT questionnaire was given to both the store owner and employee to gain their feedback on the functionality of this system. This final phase ensures that the software is user-friendly, functions as intended, and performs reliably in real-world scenarios. The test cases of the system will be as depicted in Table 6. Table 7 shows the overall result of the test cases carried out. There are a total of 27 test cases with all of them pass at 100% and no fail test case being recorded.

Table 6 *List of test cases*

No.	Test Cases	Description
TEST_100 (Login Module)		
1.	TEST_100_001	Verify if store owner and employee will be able to login with valid username and password.
2.	TEST_100_002	Verify if store owner and employee unable to login with empty username or password's text field.
3.	TEST_100_003	Verify if store owner and employee unable to login with incorrect password.
4.	TEST_100_004	Verify if the system alerts users upon invalid username or password.
5.	TEST_100_005	The system shall display the dashboard upon successful login.
TEST_200 (Manage Suppliers Module)		
1.	TEST_200_001	Verify if store owner can add a new supplier.
2.	TEST_200_002	Verify if store owner can edit existing supplier.
3.	TEST_200_003	Verify if store owner can delete existing supplier.
4.	TEST_200_004	Verify if store owner and employee can view the list of suppliers added.
5.	TEST_200_005	Verify if employee can add purchase order details.
TEST_300 (Manage Stock Module)		
1.	TEST_300_001	Verify if employee can add product.
2.	TEST_300_002	Verify if employee can edit the existing products.

3.	TEST_300_003	Verify if employee can delete the existing products.
4.	TEST_300_004	Verify if store owner and employee can search product using product ID.
5.	TEST_300_005	Verify if store owner and employee can filter search products.
6.	TEST_300_006	Verify if employee can register product in and out.
TEST_400 (Manage Low Stock Product Reminder)		
1.	TEST_400_001	Verify if store owner can set maximum stock level for every product.
2.	TEST_400_002	Verify if the system calculates stock percentage based on maximum stock level and current stock level.
3.	TEST_400_003	Verify if the system displays the stock percentage.
4.	TEST_400_004	Verify if the system push reminder for low stock product based on its percentage.
TEST_500 (Manage Expiry Dates Module)		
1.	TEST_500_001	Verify if employee able to add expiry date.
2.	TEST_500_002	Verify if employee able to edit expiry date.
3.	TEST_500_003	Verify if store owner and employee can view expiry dates.
4.	TEST_500_004	Verify if store owner and employee receive reminder on nearing expiry items.
TEST_600 (Generate Reports Module)		
1.	TEST_600_001	Verify if store owner can view report of suppliers.
2.	TEST_600_002	Verify if store owner can view report of expiring products based on the parameters (date ranges)
3.	TEST_600_003	Verify if the store owner can download or print the report.

Table 7 *Overall test results*

No.	Test case ID	Number of test cases	Number of passed test cases	Pass (%)
1.	TEST_100	5	5	100
2.	TEST_200	5	5	100
3.	TEST_300	6	6	100
4.	TEST_400	4	4	100
5.	TEST_500	4	4	100
6.	TEST_600	3	3	100
	Total	27	27	100

6. Conclusion

This final section outlines the advantages, disadvantages, and suggestions for improving the inventory management system created for Kedai Buah-buahan Misri Sumiran. The system has been tested thoroughly by both users and developers. It effectively meets the company's requirements and ensures efficient functionality through rigorous user acceptance testing.

The system provides several benefits: it replaces manual inventory processes with digital solutions, notifies staff about products nearing expiry within 7, 14, and 30 days, and alerts the store owner about low stock levels to facilitate timely reordering. However, there are some downsides, including the ability to generate only basic reports, unsuitability for large businesses, absence of two-factor authentication for password recovery, and lack of email or SMS alerts for reminders.

To further enhance the system, it is recommended to add customizable reporting tools, implement two-factor authentication, enable email and SMS alerts for reminders, and develop a mobile app for inventory management on the go. In summary, this inventory management system successfully improves daily operations at Kedai Buah-buahan Misri Sumiran and meets the initial project objectives.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

References

[1] M. Waite, Pocket Oxford English Dictionary. Oxford University Press, 2013. [Online]. Available: http://books.google.ie/books?id=xqKcAQAAQBAJ&printsec=frontcover&dq=ISBN-13.+978-0199640942&hl=&cd=8&source=gbs_api

[2] H. Sabah Salih, M. Ghazi, and M. Aljanabi, "Implementing an Automated Inventory Management System for Small and Medium-sized Enterprises," Iraqi Journal for Computer Science and Mathematics, pp. 238–244, May 2023, doi: 10.52866/ijcsm.2023.02.02.021.

[3] J.-C. B. Munyaka and S. V. Yadavalli, "INVENTORY MANAGEMENT CONCEPTS AND IMPLEMENTATIONS: A SYSTEMATIC REVIEW," South African Journal of Industrial Engineering, vol. 32, no. 2, 2022, doi: 10.7166/33-2-2527.

[4] Meiryani, "DECISION MAKING AND MANAGEMENT INFORMATION SYSTEMS," Journal of Critical Reviews, vol. 7, no. 07, Apr. 2020, doi: 10.31838/jcr.07.07.52.

[5] A. S. Hornby, Oxford Advanced Learner'"s Dictionary of Current English. 2015. [Online]. Available: http://books.google.ie/books?id=ezFprgEACAAJ&dq=ISBN-13.+978-0194798792&hl=&cd=1&source=gbs_api

[6] Zoho, "Zoho | Cloud Software Suite for Businesses," Zoho. https://zoho.com/. Accessed: Nov. 16, 2024.

[7] Sortly, "Sortly: Inventory Simplified," Sortly. https://www.sortly.com/. Accessed: Nov. 16, 2024.

[8] Lightspeed, "Lightspeed," Lightspeed. https://lightspeedhq.com/. Accessed: Nov. 16, 2024.

[9] H. K. Aroral, "Waterfall Process Operations in the Fast-paced World: Project Management Exploratory Analysis," International Journal of Applied Business and Management Studies, vol. 6, no. 1, Apr. 2021.

Appendix A

Use Case Specifications – Login

Table 8 shows the login use case specification.

Table 8 *Use Case Specifications – Login*

Use Case Name	Login
Use Case ID	UC-1
Actors	Store owner and employee
Description	User needs to log in to have access to the system.
Preconditions	1. User must register first to create an account. 2. User login using their registered username and password.
Postconditions	The system displays the dashboard.
Normal Flow	**1.0 Login** 1. User inserts username. 2. User inserts password. 3. User clicks login button. 4. System validates the user. See E.1. 5. System redirects to the home page.
Exceptions	**E.1 Fail to login** 1. System will display 'Invalid username or password. Please try again.' 2. The username and password text field will reset.

Use Case Specifications – Manage Suppliers

Table 9 shows the manage suppliers use case specification.

Table 9 *Use Case Specifications – Manage Suppliers*

Use Case Name	Manage Suppliers
Use Case ID	UC-2
Actors	Store owner and employee
Description	To manage suppliers of the products at the store.
Preconditions	User successfully login and have access to the system.
Postconditions	Suppliers' information is all kept in the system.
Normal Flow	**1.0 Manage Suppliers** 1. Store owner selects 'Manage Supplier' option. 2. Store owner selects 'Add Supplier' option to add new supplier. See A.1., E.1. 3. Store owner selects 'Update Supplier' option to update supplier information. See A.2., E.1. 4. Store owner selects 'Delete Supplier' option to delete supplier information. See A.3. 5. Store owner selects 'View Suppliers' to view list of suppliers. 6. Store owner selects 'Add PO' to enter purchase order (PO) information. See A.4., E.1.
Alternative Flow	**A.1 Add new supplier.** 1. Store owner selects 'Add Supplier'. 2. Store owner enters supplier information: company name, company address, contact information, and products supplied. 3. Store owner clicks 'Add' to confirm. **4.** System updates with newly added supplier.
	A.2 Update existing supplier. 1. Store owner selects 'Update Supplier'. 2. System display list of suppliers. 3. Store owner selects one supplier to be updated its information. 4. Store owner fills in information to be updated: company name, company address, contact information, purchase order number, and products supplied.

	5. Store owner clicks 'Update' to confirm. **6.** System updates with updated supplier.
	A.3 Remove existing supplier. 1. Store owner selects 'Delete Supplier'. 2. System display list of suppliers. 3. Store owner selects one supplier to be deleted. 4. Store owner clicks 'Delete' to confirm deletion. 5. System updates with new list of suppliers.
	A.4 Add purchase order information. 1. Employee selects 'Add PO'. 2. Employee enters purchase order information: supplier information, PO information, products delivered, and delivery terms. 3. Employee selects 'Add' to confirm. 4. System updates with newly added purchase order information.
Exceptions	**E.1 Invalid form entry.** 1. System will display 'Invalid information. Please try again.' 2. The text field will reset.

Use Case Specification – Manage Stock

Table 10 shows the manage stock use case specification.

Table 10 Use Case Specification – Manage Stock

Use Case Name	Manage Stock
Use Case ID	UC-3
Actors	Employee
Description	To manage stock at the store and register in and out product at the store.
Preconditions	User successfully login and have access to the system.
Postconditions	The stock is updated according to actions performed by user.
Normal Flow	**1.0 Manage Stock** 1. Store owner and employee selects the 'View Stock' to view list of products available at the store. 2. Employee selects 'Add Product' option to add new product. See A.1., E.1. 3. Employee selects 'Update Product' option to update details of existing product. See A.2., E.1. 4. Employee selects 'Remove Product' option. See A.3. 5. Store owner and employee select 'Barcode Scanning' option to retrieve and display product information. See E.2. 6. Employee selects 'Register In/Out' option to register product in and out of the store. See A.4., E.1. 7. Store owner and employee filter products based on the expiry dates and supplier.
Alternative Flow	**A.1 Add new product.** 1. Employee clicks 'Add Product' button. 2. Employee enters product details: product name, product quantity, the supplier, and purchase order number. 3. Employee clicks 'Add' button to confirm. 4. System updates the inventory with newly added product.
	A.2 Update existing product. 1. Employee clicks 'Update Product' button. 2. System displays list of existing products. 3. Employee selects one product to be updated. 4. Employee update relevant information: product name, product quantity, and supplier. 5. Employee clicks 'Update' button to confirm. 6. System updates the inventory with new info.
	A.3 Remove existing product. 1. Employee clicks 'Remove Product' button. 2. System displays list of existing products.

	3. Employee selects one product to be deleted. 4. Employee clicks 'Delete' to confirm product to be deleted. 5. System updates the inventory with new list of existing products
	A.4 Register product in and out. 1. Employee clicks 'Register In/Out' button. 2. System prompts employee to choose 'Register In' or 'Register Out'. 3. Employee enters relevant details: product and quantity. 4. Employee clicks 'Confirm' button to confirm. 5. System updates and displays the newly updated inventory. 6. System updates with newly added purchase order information.
Exceptions	**E.1 Invalid form entry.** 1. System will display error message for invalid form entry. 2. System will prompt user to fill in form again.
	E.2 Invalid barcode. 1. System will display error message for invalid barcode and asks user to scan barcode again.

Use Case Specification – Manage Low Stock Product Reminder

Table 11 shows the manage low stock product reminder use case specification.

Table 11 *Use Case Specification – Manage Low Stock Product Reminder*

Use Case Name	Manage Low Stock Product Reminder
Use Case ID	UC-4
Actors	Store Owner
Description	To set maximum stock level for every product and calculate stock percentage then, push reminder for low stock item.
Preconditions	Products are added to the system.
Postconditions	The inventory is updated according to actions performed by user.
Normal Flow	**1.0 Manage Low Stock Product Reminder** 1. Store owner selects 'Manage Stock Level' option. 2. System displays list of existing products. 3. Store owner selects one item to enter its maximum stock level. 4. Store owner clicks 'Confirm' to confirm the maximum stock level of the product. See E.1. 5. System updates the product information with its maximum stock level. 6. System calculates stock percentages for every product based on its maximum stock level and current stock. 7. System display stock percentage of every product. 8. System push reminder for product with low percentage of stock.
Exceptions	**E.1 Fail to apply maximum stock level** 1. System will display 'Invalid input. Please enter valid input.' 2. The text field will reset.

Use Case Specification – Manage Expiry Dates

Table 12 shows the manage expiry dates use case specification.

Table 12 *Use Case Specification – Manage Expiry Dates*

Use Case Name	Manage Expiry Dates
Use Case ID	UC-5
Actors	Employee
Description	To manage expiry dates of every product.
Preconditions	Products are added to the system.
Postconditions	Expiry dates for products are entered and reminders are pushed for products nearing expiry date.
Normal Flow	**1.0 Manage Expiry Dates** 1. Employee selects the 'Manage Expiry Dates'. 2. System displays list of existing products at the store.

	3. Employee selects one product to manage its expiry date. 4. Employee selects 'Enter Expiry Date' option to enter the expiry date of the product. See A.1., E.1. 5. Employee selects 'Edit Expiry Date' option to edit the entered expiry date. See A.2., E.1. 6. Employee selects 'View Expiry Date' option to view the product's expiry dates by applying the filter. 9. System push reminder for products nearing its expiry date. See A.3.
Alternative Flow	**A.1 Enter Expiry Date.** 1. System displays list of existing products. 2. Employee selects one product. 3. Employee selects 'Enter Expiry Date'. 4. Employee enters expiry date of the product. 5. Employee clicks 'Confirm' button to confirm. 6. System updates the product with its expiry date.
	A.2 Edit Expiry Date. 1. System displays list of existing products. 2. Employee selects one product. 3. Employee selects 'Edit Expiry Date'. 4. Employee enters the correct expiry date. 5. Employee clicks 'Confirm' button to confirm. 6. System updates the product with its edited expiry date.
	A.3 Push Reminder 1. System checks for products nearing its expiry dates. 2. System display products nearing expiry date on dashboard. 3. System push reminder for products nearing its expiry date.
Exceptions	**E.1 Invalid Expiry Date.** 1. System will display 'Invalid date. Please enter the valid date.' 2. The text field will reset.

Use Case Specification – Generate Reports

Table 13 shows the generate reports use case specification.

Table 13 *Use Case Specification - Generate Reports*

Use Case Name	Generate Reports
Use Case ID	UC-6
Actors	Store owner
Description	To generate reports of suppliers, monthly stock pattern, and expiring products.
Preconditions	User successfully login and have access to the system.
Postconditions	Reports generated for list of suppliers, monthly stock pattern, and expiring products.
Normal Flow	**1.0 Generate Reports** 1. Store owner selects 'Generate Reports' option. 2. The system displays three options: 'Supplier Report', 'Monthly Stock Pattern', and 'Expiring Products'. 3. Store owner selects 'Supplier Report' to view report about suppliers and their product. See A.1., E.1. 4. Store owner selects 'Expiring Products' to view the list of expiring products in the next few days, weeks, or months. See A.2., E.1. 5. The system prompts user to download report in pdf or print.
Alternative Flow	**A.1 Generate Supplier Report.** 1. Store owner selects 'Supplier Report'. 2. Store owner selects which supplier details to be displayed. 3. The system displays the supplier along with list of products they delivery to the store.
	A.2 Generate Expiring Products Report. 1. Store owner selects 'Expiring Products'. 2. Store owner selects in the next how many days, weeks, or months.

	3. The system displays list of expiring products for the next days, weeks, or months.
Exceptions	**E.1 Fail to generate report** 1. System will display 'Unable to display report. Please try again.' 2. The text field will reset.

Appendix B

Fig. 14 shows the functionality that be achieved from the store owner's dashboard and Fig. 15 shows the functionality that be achieved from the employee's dashboard.

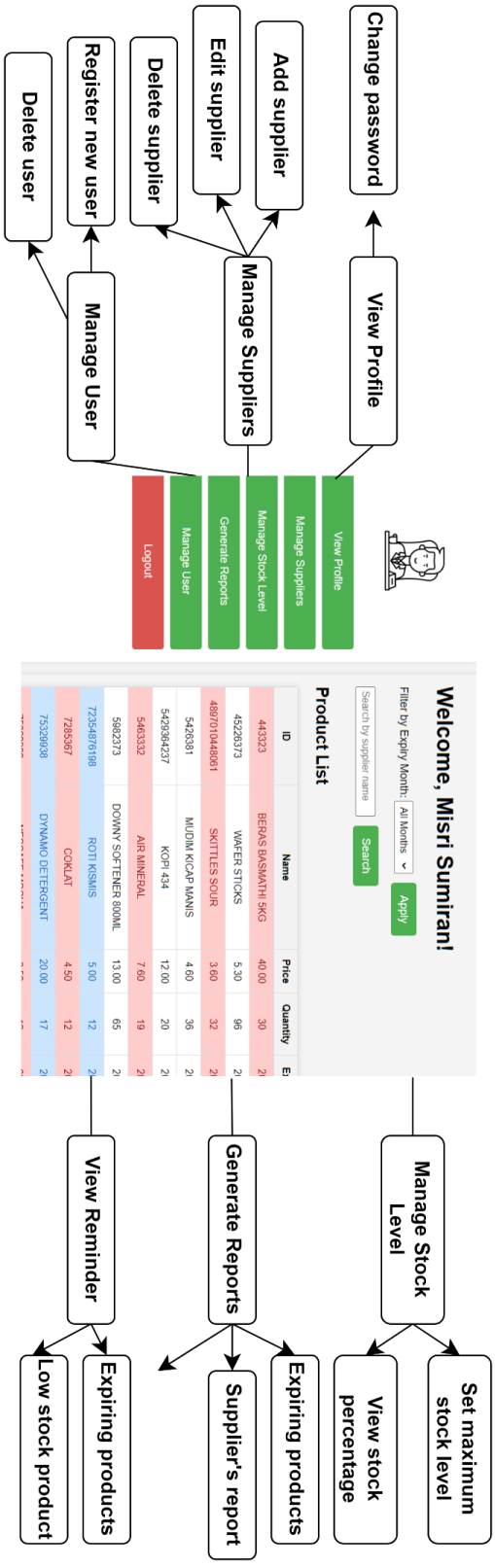

Fig. 14 *Store owner's dashboard*

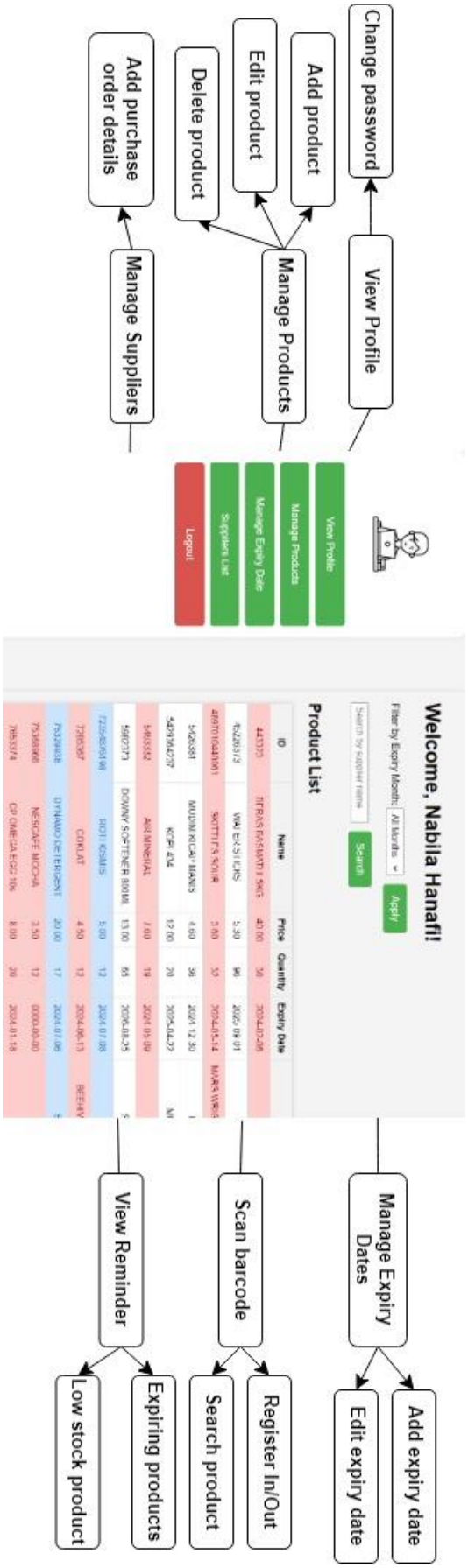

Fig. 15 *Employee's dashboard*