202

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

AITCS

Vol. 6 No. 1 (2025) 1145-1164
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

MCM Inventory Management System (MCM – IMS)

Asweni Thavamani[1], Rabatul Aduni Sulaiman[1*]

[1] *Fakulti Sains Komputer dan Teknologi Maklumat,*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: rabatul@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.01.063

Article Info

Received: 7 February 2025
Accepted: 18 June 2025
Available online: 30 June 2025

Keywords

Inventory management, web-based system, waterfall methodology, Inventory processes, automated notification, project phases.

Abstract

The MCM Inventory Management System (MCM-IMS) is a web-based solution designed to revolutionize inventory management in mini-markets. The system aims to address issues such as time-consuming manual order processing, inefficient stock tracking, and systemic delays. The system includes features like real-time visibility of stock levels, automated low-inventory notifications, seamless integration with point-of-sale terminals, and comprehensive reporting tools. The goal is to enhance operational efficiency, accuracy, and customer satisfaction while easing current inventory management challenges. The project utilizes advanced technology to address problems associated with traditional inventory procedures, employing a methodical approach with object-oriented design and exact user acceptance testing to ensure a reliable and error-free system. MCM-IMS is designed to accommodate the unique needs of various retail sectors through scalability and customization, promoting improved operational agility and performance. The project is expected to change Inventory management, bringing about a new era of accessible, accurate, and efficient inventory data management.

1. Introduction

Pasar Mini MCM Utara Trading, a retail business, has implemented the MCM Inventory Management System (MCM-IMS) to optimize and streamline inventory management processes. Previously, suppliers manually verified stock levels and sought manager approval before restocking products, which was time-consuming and prone to errors. This manual approach led to inefficiencies in stock replenishment and tracking. MCM-IMS has transformed the store's operations by reducing manual labor and costs, improving accuracy, and enhancing overall operational efficiency. The web-based system simplifies inventory management, allowing for real-time tracking of stock levels, automated restocking requests, and comprehensive reporting on sales trends, product performance, and stock turnover [1]. This automation minimizes delays and errors, ensuring timely and accurate inventory updates. With its user-friendly interface, the system requires minimal training for suppliers, minimizing disruptions to daily operations. As a result, MCM-IMS has become an integral part of Pasar Mini MCM Utara Trading's operations, enabling the store to provide essential products to the community efficiently and effectively. Overall, the MCM Inventory Management System transforms inventory management procedures, making them more efficient and reliable. By reducing manual labor and costs, enhancing accuracy, and improving operational efficiency, the system supports Pasar Mini MCM Utara Trading in effectively serving the community's needs with a diverse range of products

1.1 Problem Statement

MCM Inventory Management System faces challenges in the inventory management processes. To improve efficiency and operational effectiveness, the organization should address several critical concerns. The lengthy procedure of manual order processing by suppliers is one of the main problems. There are major delays because each order needs to be manually checked, verified, and approved order detail. This process also requires extensive human intervention, increasing the risk of errors and inefficiencies, and consuming valuable supplier time that could be better utilized elsewhere. Furthermore, without a centralized system, it becomes difficult to monitor stock levels and identify products with high or low demand for it. The system mostly depends on irregular manual stock checks, which are labor-intensive and prone to errors, in the absence of real-time inventory tracking. It is difficult to maintain ideal stock levels because of the frequent manual interventions required due to this lack of visibility into inventory levels, which puts an additional burden on time and resources. Often managers experience delays while they await confirmation and order insertion due to suppliers' manual stock inspections. In addition to slowing down order processing overall, this makes systemic inefficiencies more difficult to resolve because managers are unable to act quickly or make judgements based on accurate stock information. The human checks create an interruption impact that slows down the supply process as a whole and lowers the inventory management system's overall responsiveness. To address these challenges, an Inventory Management System should be developed to facilitate effective data access, ensure data correctness and provide real time data access.

1.2 Objective

The objectives of developing the MCM Inventory Management System are as follows:

i. To design the MCM Inventory Management System (MCM-IMS) using the object-oriented approach.
ii. To develop a web-based management system
iii. To test MCM Inventory Management System using user acceptance testing.

1.3 Scope Project

An extensive web-based system created to optimize and simplify an organization's inventory management processesis called the MCM Inventory Management System. This project encompasses the following key features and functionalities. The system includes two user roles, which are manager and supplier. In the other hand manager is responsible for managing the entire system while supplier involves day-to-day operation related to inventory management. Table 1 shows the modules of the system.

Table 1 *Modules system of MCM Inventory Management System*

Modules	Description
Login	Users can enter the system by entering username and password
Manage Inventory	Add, edit or delete inventory
Manage Supplier	Add, edit or delete supplier
Manage Profile	View and update profile information
Order Request	Add order for restocking inventory items and track order
Manage Purchase Order	Supplier will accept or reject incoming request
Payment	Allow managers to make payment once an order has been accepted by supplier
Manage Category	Add, edit or delete categories
Sales List	Allow managers to process transaction
Notification	Receive notifications for low stock levels, pending orders alerts by notify thru system
Generate Report	View sales report, order list report and revenue by day, week, year and month

2. Related Work

This section discusses the current understanding of a computerized system and compares it with the proposed system through a literature review. The review aims to gather data and identify issues, while also serving as the project's source material. The literature review helps understand linked initiatives and solve important issues. Odoo Inventory, Zoho Inventory & Trade Gecko Inventory, and MCM Inventory Management System are the systems that are compared in this section.

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 1145-1164

2.1 Odoo Inventory

The literature highlights the essential features of Odoo Inventory storage, optimization, replenishment automation, traceability, and reporting for efficient inventory management. These complement the privilege of the suggested Advanced Inventory Management System nicely, improving storage effectiveness, simplifying processes, and facilitating well-informed decision-making. Leveraging Odoo's integrated platform and automation, our system benefits from reduced manual effort and improved scalability [2]. As shown in Fig. 1, the Homepage for Odoo Inventory provides an intuitive and user-friendly interface that supports these functionalities effectively.

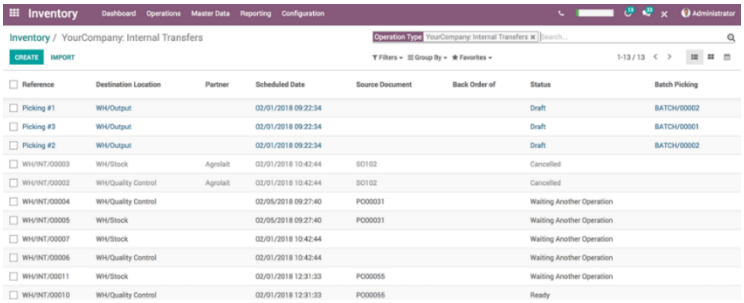


Fig. 1 *Homepage for Odoo Inventory*

2.2 Zoho Inventory

Zoho Inventory offers an array of features tailored for small to medium-sized enterprises, simplifying inventory management [3]. Features include robust inventory monitoring with batch and serial number tracking, buy order and backorder management. These features support our goals for the proposed Zoho Inventory System, which is to improve the accuracy and efficiency of inventory management. By making use of Zoho Inventory's capabilities, our system improves productivity, lowers error rates, and provides deeper insights into sales success. Finance Suite's integration with Zoho guarantees smooth coordination with accounting and bookkeeping responsibilities, enabling users to maximize inventory management and maintain exact client information. As shown in Fig. 2, the Homepage for Zoho Inventory provides a comprehensive and user-friendly interface that enhances these functionalities.

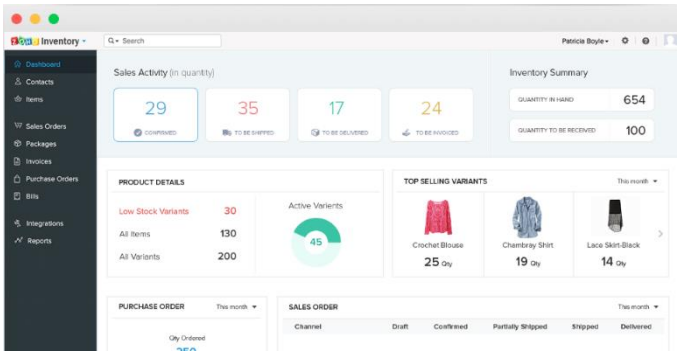


Fig. 2 *Homepage for Zoho Inventory*

2.3 Trade Gecko Inventory

Trade Gecko as shown in Fig. 3 was a cloud-based inventory and order management tool for small and medium-sized businesses founded in Singapore [4]. It optimized inventory, orders, and sales across platforms like Shopify and Amazon, and included features like multi-location tracking, real-time stock updates, and external interfaces. It was acquired by Intuit in 2020 and called QuickBooks Commerce before being deactivated in 2022. Users are now considering options such as Ordoro, Zoho Inventory, and Cin7.

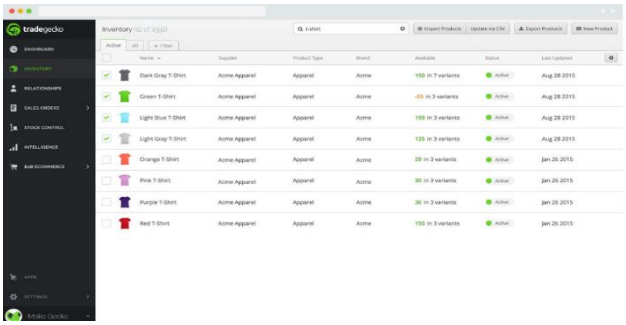


Fig. 3 *Homepage for Trade Gecko Inventory*

2.4 Comparison between related systems

The MCM Inventory Management System (MCM-IMS) stands out from competitors like Odoo Inventory, Zoho Inventory, and Trade Gecko Inventory with its extensive features for inventory management. Unlike its competitors, MCM-IMS includes order requests, request management, stock control, supplier management, sales monitoring, notifications, and report generation. Its user-friendly, web-based platform is accessible from anywhere. Utilizing PHP, HTML, CSS, JavaScript, and SQL, MCM-IMS ensures reliable performance and scalability. Among these, Odoo Inventory is the most similar but lacks key features like payment handling, sales tracking, and notifications, which MCM-IMS provides for better efficiency. Table 2 compares MCM-IMS with existing systems.

Table 2 *Comparison between the proposed system and the existing systems*

Comparison/ System	Odoo Inventory	Zoho Inventory	Trade Gecko Inventory	MCM-IMS
Modules:				
Login	Yes	Yes	Yes	Yes
Manage Inventory	Yes	Yes	No	Yes
Manage Supplier	Yes	No	Yes	Yes
Manage Profile	Yes	Yes	Yes	Yes
Order Request	Yes	Yes	Yes	Yes
Manage Purchase Order	No	Yes	No	Yes
Payment	No	No	No	Yes
Manage Category	Yes	Yes	Yes	Yes
Sales List	No	No	No	Yes
Notification	No	No	Yes	Yes
Generate Report	Yes	Yes	Yes	Yes

3. Methodology

Table 3 shows that the tasks completed in each step utilizing the waterfall model methodologies are explained. The requirement and analysis phase, design phase, implementation phase, testing phase, and maintenance phase are the five stages of this process. A Gantt chart that carefully details the sequential tasks and their related timeframes inside each step of the software development process is an essential visual tool that we have included in the Appendix A, Fig A.1, the extensive documentation that has been supplied. This diagram functions as an essential road map, showing the intended flow of tasks from requirements study to maintenance. Its appendix placement helps team members and stakeholders comprehend the organized nature of the project's execution by providing a clear, concise summary of task dependencies, durations, and the overall project timeline.

Table 3 *Software development activities task*

Phases	Task/Activities	Deliverables	Tools
Requirement & Analysis	- Identify system scope and requirement - Analyze requirement: Data collection – interview client to identify system requirement -Project timeline –Gantt Chart	i. Proposal ii. Gantt Chart	i. MS Word ii. TeamGantt.com

Table 3: *(continued)*

Design	- Design user interface as perrequirement - Design database as perrequirement - Design use case, sequence diagram, activity diagram, and class diagram	i. Use Case Diagram ii. Sequence Diagram iii. Activity Diagram iv. Class Diagram v. Schema table vi. Interface	i. Draw.io ii. PowerPoint
Implementation	The designated design will beconverted to actual code	To execute the system	i. Visual Studio Code ii. XAMPP Control Panel
System Testing	User acceptance testing	Test the functionalities and acceptance test	i. Visual Studio Code ii. XAMPP Control Panel iii. Google Form
Maintenance	Involve correcting errors and upgrading due to current trends and needs	Make sure the system is ready to use by the stakeholder and ensure the quality of use which provides effectiveness, efficiency, satisfaction, and reliability. and secure	i. Visual Studio Code ii. XAMPP Control Panel

4. Analysis and Design

This section will discuss the analysis and design phases of the system, covering system requirement analysis, which includes both functional and non-functional requirements, database design, and implementation. The design phase involves detailing the components and interfaces to address these requirements and constraints.

4.1 System Requirement Analysis

The process of identifying the needs that a constructed system needs to meet and the expectations of users for the suggested system is known as requirement analysis. The system requirements include user requirements, system requirements, and functional and non-functional requirements. The functional needs and the proposed system's description are shown in Table 4, while the non-functional requirements and their descriptions are shown in Table 5.

Table 4 *Functional Requirements*

Modules	Description
Login	• The system should allow users to log in with username and password • The system should verify user credentials against stored database data. • The system should show alerts for incorrect login attempts. • The system should direct users to appropriate pages upon successful login based on their roles (Manager or Supplier).
Manage Inventory	• The system should allow users to add new inventory items, including product name, description, quantity, expiration date, and price. • The system should allow users to edit details of existing inventory items. • The system should allow users to delete inventory items that are no longer available. • The system should monitor stock levels in real-time and send automatic notifications when stock levels fall below a predefined threshold

Table 4: *(continued)*

Manage Supplier	• The system should allow users to add and maintain a database of suppliers, including company name, contact person, address, phone number, email, and contract details. • The system should allow users to edit supplier details to keep information up to date. • The system should allow users to delete suppliers that are no longer in use. • The system should categorize suppliers based on products or services they provide. • The system should manage supplier contracts and renewal dates.
Manage Profile	• The system should allow users to view personal profile information including name, contact details, and role. • The system should allow users to update profile details such as email, phone number, and address. • The system should provide a change password feature for security purposes. • The system should allow users to upload and update profile pictures.
Order Request	• The system should allow users to place orders for restocking inventory items, specifying product details, and quantities. • The system should track the status of each pending order. • The system should generate purchase orders and send them to suppliers. • The system should maintain a history of all orders placed for future reference and analysis.
Manage Purchase Order	• The system should allow suppliers to receive incoming requests from managers for order approvals or specific action. • The system should allow suppliers to accept or reject requests • The system should notify managers of the request status (pending, accepted, rejected or order shipped). • The system should track the history and status of all requests. • The system should allow managers to view the status and details of their submitted requests in real time.
Payment	• The system should allow the manager to initiate payment after the supplier has accepted the order. • The system should prompt the manager to input card details (credit/debit card number, expiration date, CVV) or integrate with a payment gateway for secure card transactions. • The manager should be able to track the status of the COD payment in the system • The system should ensure that payment information, especially card details, is securely
Manage Category	• The system should allow the manager to create new categories. • The manager should be able to update the category name
Sales List	• The system should provide a user-friendly interface for managers to add item • The system should allow managers to process transactions efficiently, including calculating totals of sales. • The system should track daily sales and transaction details. • The system should integrate with inventory management to automatically update stock levels after each sale.
Notification	• The system should send automated notifications for low stock levels and pending order alerts via email.
Generate Report	• The system should allow users to access and generate detailed reports on sales and revenue by day, week, month, and year. • The system should allow users to export reports in PDF format.

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 1145-1164

Table 5 *Non-Functional Requirements*

Modules	Description
Usability	• The system's interface should be intuitive and user-friendly to reduce the learning curve for new users. • Provide clear error messages, tooltips, and prompts to help users.
Performance	• Standard queries should result in reports that are created and displayed within 5 seconds. • The system should respond to user actions within 2 seconds for most operations.
Security	• Encrypt user passwords for safe authentication.

4.2 System Design

The system's architecture, interface, and database designs are all covered in the specification. While the design of the system and a schema table comprise the interface and database designs, respectively, thearchitectural design is represented by an architecture diagram.

4.3 UML Diagram

UML is a powerful tool for visually representing software and system structures, aiding engineers in understanding complex designs and implementations [5]. This section will explain the system design and analysis for the proposed system, which adopts an object-oriented approach. System analysis involves meticulous examination and development to determine functionality. Throughout the process, system analysis is initially generated and then translated using specialized diagrams like Use Case and class diagrams. Fig. 4 illustrates the Use Case diagram for the proposed system.

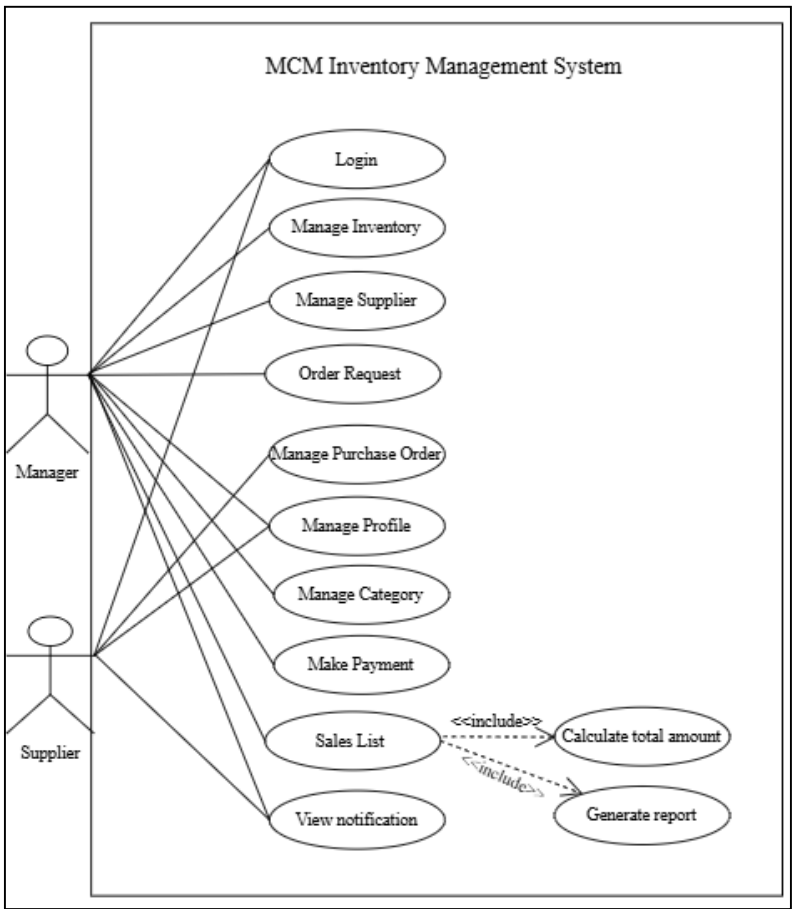


Fig. 4 *Use Case Diagram*

4.4 Class Diagram

Class diagram is used to explain the static view of a system (McGill, 2001; Rumbaugh, 2005). The class diagram for the MCM IMS shows a system intended for effective management of suppliers, orders, sales, notifications, and reports in addition to inventory [8]. In Fig. 5 depicts the organization of a database for handling items, categories, suppliers, inventory, purchase orders, and sales. It contains several critical tables, including category_list, item_list, supplier_list, purchase_order_list, po_items, stock_list, sales_list, and users. Each table contains attributes such as IDs, names, and other information, along with relationships between them.

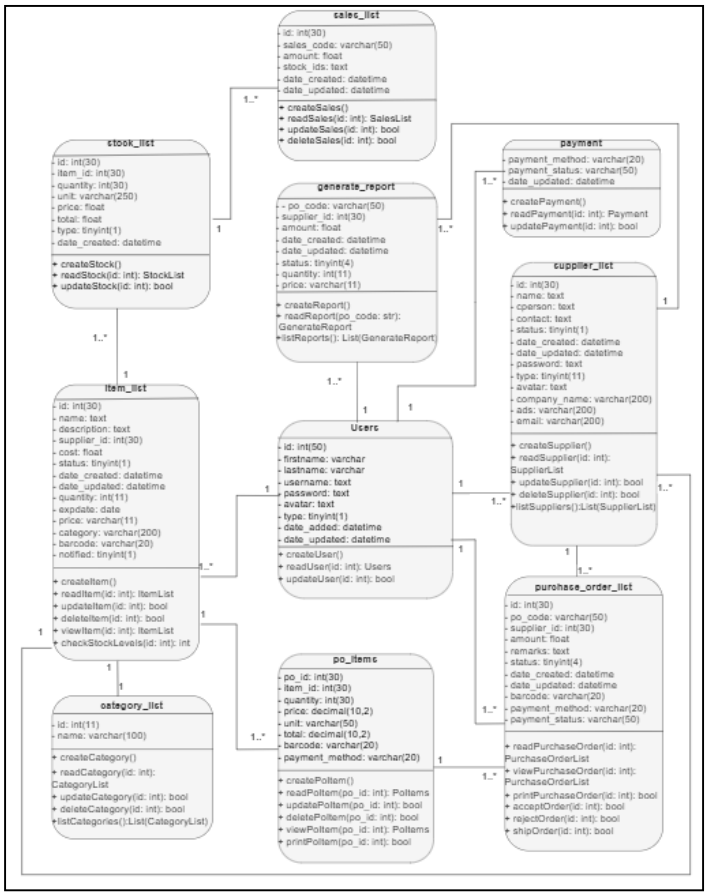


Fig. 5 *Class Diagram*

4.5 Use Case Specification

A use case specification is a document that describes the interactions between users and a computer system, with the goal of achieving a specific goal or set of goals. It typically includes a description of the actors, the goals or objectives of the interactions, the steps that must be taken to achieve those goals, and any constraints or preconditions that must be met [7]. For further visualization of the interaction flow, reference can be made to Appendix B, which contains the sequence diagram and activity diagram illustrating the sequential steps and activities involved in the use case scenario.

Use Case Specification for Login: The use case allows managers and supplier to log in to the system using their username and password, verify their credentials, handle incorrect login attempts, direct users to the appropriate pages based on their roles, maintain user login sessions, offer password reset mechanisms, and securely save profile changes. For a visual representation of the login process, please refer to Appendix B Fig B.1 for the sequence diagram illustrating the sequential steps involved in the login process, and Appendix B Fig B.2 for the activity diagram showcasing the activities performed during the login scenario [10].

Use Case Specification for Manage Supplier: The use case allows managers to add, edit, and delete suppliers, maintain supplier details, categorize suppliers based on products or services they provide, and manage supplier contracts. For a visual representation of the supplier management process, please refer to Appendix B Fig B.3 for the sequence diagram illustrating the sequential steps involved in managing suppliers, and Appendix B Fig B.4 for the activity diagram of the manage supplier.

Use Case Specification for Order Request: The use case allows managers to place orders for restocking inventory items, track the status of pending orders, receive notifications for order status updates, generate purchase orders for suppliers, and maintain a history of all orders placed for future reference and analysis. For a visual representation of the order request process, please refer to Appendix B Fig B.5 for the sequence diagram illustrating the sequential steps involved in the order request process, and Appendix B Fig B.6 for the activity diagram showcasing the activities performed during the order request.

4.6 Interface Design

Interface Design involves creating user-friendly and visually appealing digital platforms, such as software, websites, or apps. It entails crafting interfaces that are easy to navigate, visually appealing, and seamlessly functional [6]. This ensures users have a pleasant and intuitive experience while interacting with these digital products. Below are some Graphical User Interface (GUI) prototypes from the MCM Inventory Management System.

Login Interfaces: In this process, the user needs to log into the system by providing the correct username and password by using the login interface in Fig. 6. The system will then validate the username and password. If both username and password are correct, the system will redirect Admins or Supplier to the dashboard. If one of the usernamesor passwords is incorrect, the system will automatically display an error message. After that, the system will request the user to re-enter the correct username and password**.**

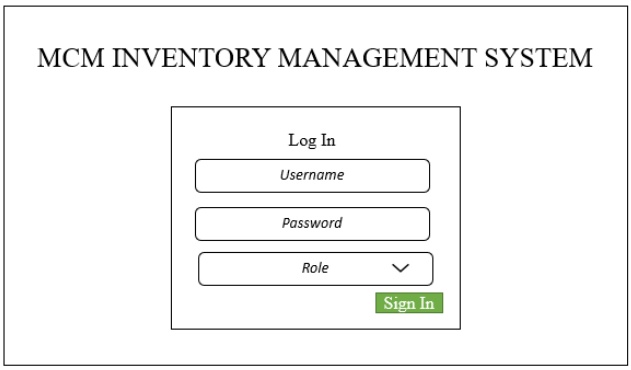


Fig. 6 *Login interface*

Manage Inventory Interfaces*:* Fig. 7(a) is an interface where manager can add inventory items within the system by inputting item details. Fig. 7(b) displays a list of stocks and at the same time manager can update inventory items.

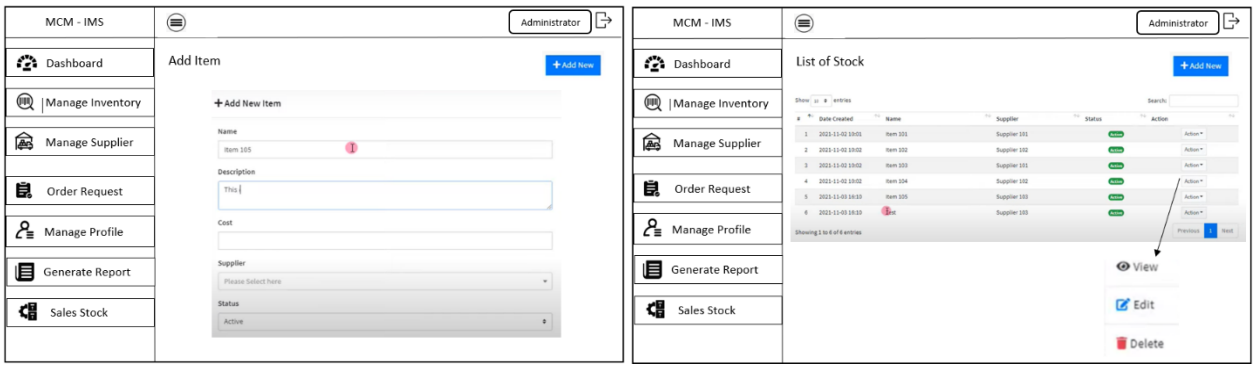


Fig. 7 *(a)Add inventory item Interface; (b) List of stock Interface*

Manage Supplier Interfaces*:* The interface below shows the interface where managers can create new suppliers by entering new supplier information such as name, contact information, and address. Fig. 8(a) depicts the Add supplier interface, where managers can input the necessary details. Fig. 8(b) illustrates the List of supplier interface, displaying the existing suppliers within the system.

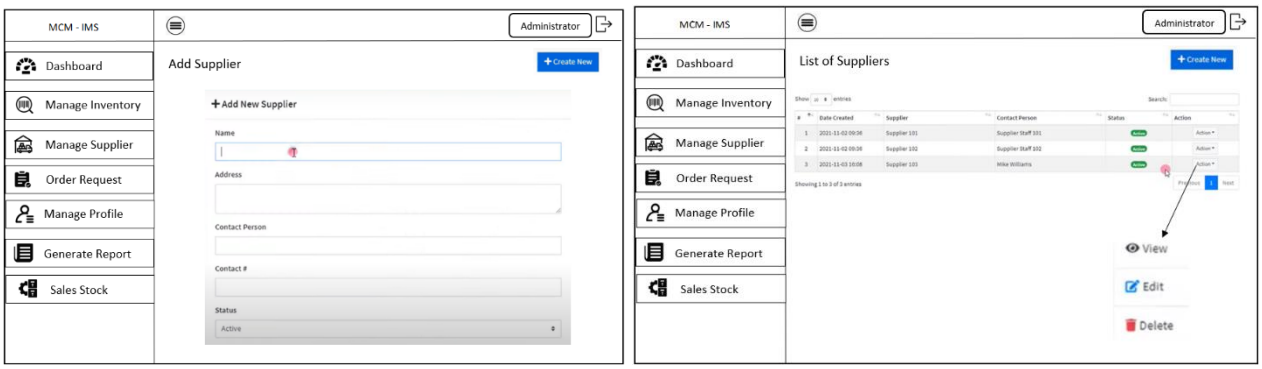


Fig. 8 *(a)Add supplier interface; (b)List of supplier interface*

5. Results and Discussion

The results and discussion section presents the findings of the study or project, interpreting them within the context of the research questions or objectives. It discusses the significance of these results, and their implications for the field, and suggests potential avenues for further research or application.

5.1 Implementation

The system uses HTML, CSS, JavaScript, Bootstrap, and jQuery for the front end, ensuring responsive and user-friendly interfaces. Bootstrap simplifies development with pre-built CSS classes. The back end is powered by PHP for server-side functionality, with CSV for Excel document generation and Apex Charts for interactive visualizations. This combination ensures efficient development and maintainable code.

5.1.1 User Login Interface

The login interface for manager and supplier is shown in Fig. 9. Users need to enter a username and password into the input fields and select the roles. If the username, password and role are valid, the users will be allowed to log in to the system. If they are not valid, the system will display a message indicating that the username, password or role are incorrect.

Fig. 9 *Login interface*

The PHP script for handling a login form submission is shown in Fig. 10(a). It sanitizes and hashes the provided username, password and roles compares them to a MySQL database, and, if a match is discovered, it initiates a session. If no match is found, it displays an error message indicating an incorrect username or password. The HTML code for styling the login form is shown in Fig. 10(b) It includes input fields for a username and password and dropdown to select type of roles and a submit button labeled "Sign In," and a prompt with a link to those who have forgotten their password

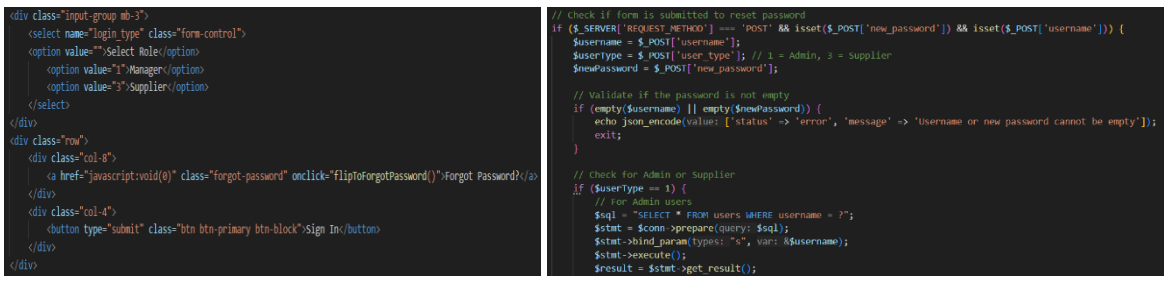


Fig. 10 *(a)Front end for user login; (b)Back end for user login*

5.1.2 Manage Inventory Interface

The interface in Fig. 11 shows a straightforward inventory management dashboard that presents a list of objects in tabular style. It contains critical information such as the Product ID, Product Name, Category, Quantity, and an Action column for tasks such as updating or deleting things. This user-friendly design ensures that inventory data is managed efficiently, giving users the tools they need to effortlessly track, search, and alter inventory records.

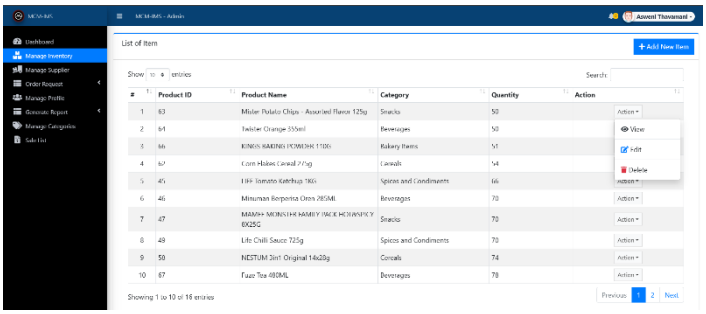


Fig. 11 *Manage Inventory interface*

For implementation in Fig. 12(a) front-end code involves developing the user interface with HTML, CSS, and JavaScript. This includes building the table to display inventory items, providing a search box for filtering, and implementing pagination to move between pages. In Fig. 12(b) back-end code uses PHP to perform server-side logic, such as obtaining, updating, deleting, and saving inventory data in a MySQL database, guaranteeing that the interface functions properly with real-time changes.

```
<div class="card-body">
    <div class="container-fluid">
    <div class="container-fluid">
        <table class="table table-bordered table-striped">
            <colgroup>
                <col width="5%">
                <col width="15%">
                <col width="25%">
                <col width="20%">
                <col width="15%">
                <col width="20%">
            </colgroup>
            <thead>
<tr>
    <th>#</th>
    <th>Product ID</th>
    <th>Product Name</th>
    <th>Category</th>
    <th>Quantity</th>
    <th>Action</th>
</tr>
```

```
<?php
$i = 1;
$qry = $conn->query("SELECT i.*, s.name as supplier FROM `item_list` i INNER JOIN supplier_list s ON i.supplier_id = s.id
ORDER BY i.quantity < 50 DESC, i.quantity ASC, i.id ASC, i.name ASC, s.name ASC");
while($row = $qry->fetch_assoc()):
    // Check if quantity is below 50
    $row_class = ($row['quantity'] < 50) ? ($i % 2 == 1 ? 'low-quantity-odd' : 'low-quantity-even') : '';
?>
<tr class="<?php echo $row_class; ?>" data-quantity="<?php echo $row['quantity']; ?>">
    <td class="text-center"><?php echo $i++; ?></td>
    <td><?php echo $row['id'] ?></td>
    <td><?php echo $row['name'] ?></td>
    <td><?php echo $row['category'] ?></td>
    <td><?php echo $row['quantity'] ?></td>
    <td align="center">
        <button type="button" class="btn btn-flat btn-default btn-sm dropdown-toggle dropdown-icon" data-toggle="dropdown">
            Action
            <span class="sr-only">Toggle Dropdown</span>
        </button>
```

Fig. 12 *(a)Front end for manage inventory; (b)Back end for manage inventory*

5.1.3 Order Request Interface

Fig. 13 shows the order request interface, which allows for more efficient purchase order request. It enables users to conveniently manage and process orders by including a status column that displays the status of each order, such as "Pending Order," "Accepted by Supplier," "Rejected by Supplier," "Order Shipped," and "Order Received." This ensures efficient and structured order management throughout the process.

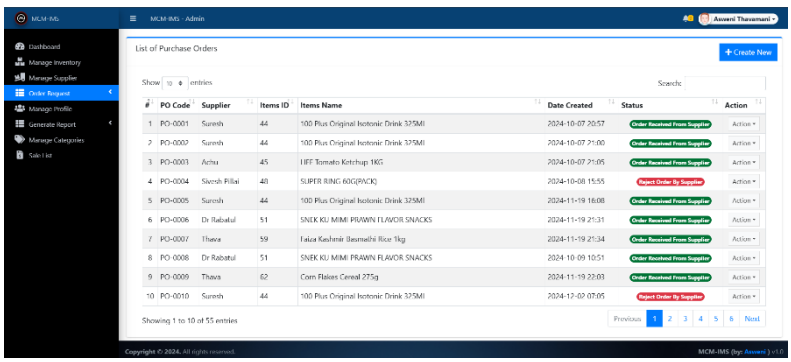


Fig. 13 *Order Request interface*

The front-end code in Fig. 14(a) generates the user interface, which includes order data, a search capability, and a form for placing new orders. The backend code in Fig. 14(b), built in PHP, processes order data, updates order

statuses, and connects to the database to store and retrieve data. They work together to guarantee that the user interface and database communicate smoothly, allowing for real-time purchase order tracking.

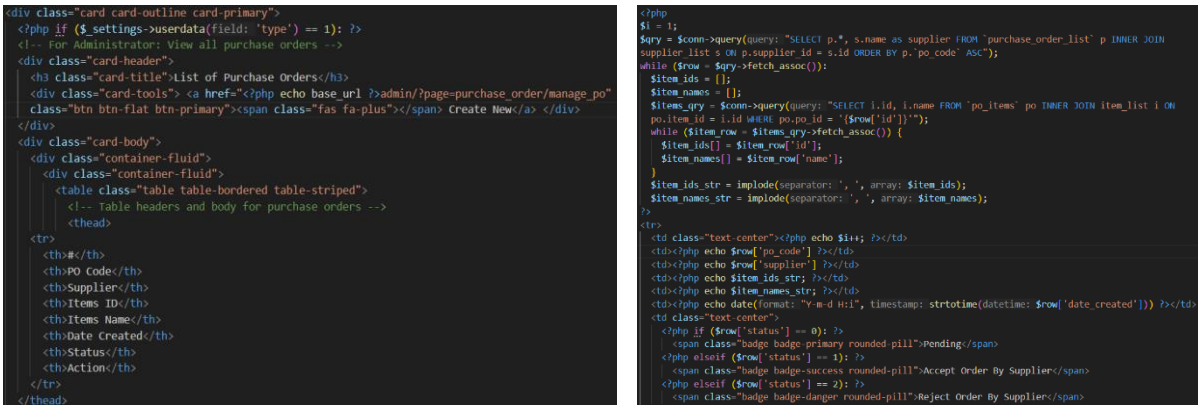


Fig. 14 *(a)Front end for order request; (b)Back end for order request*

5.1.4 Manage Purchase Order Interface

Fig. 15 shows the Manage Purchase Order Interface, which is intended to improve communication between suppliers and management about purchase orders. Suppliers can examine incoming order requests from managers and accept decline, or ship orders. The interface also monitors the status and history of each order, offering a comprehensive picture of the full order lifecycle. It ensures that both suppliers and management can effectively monitor and handle orders.

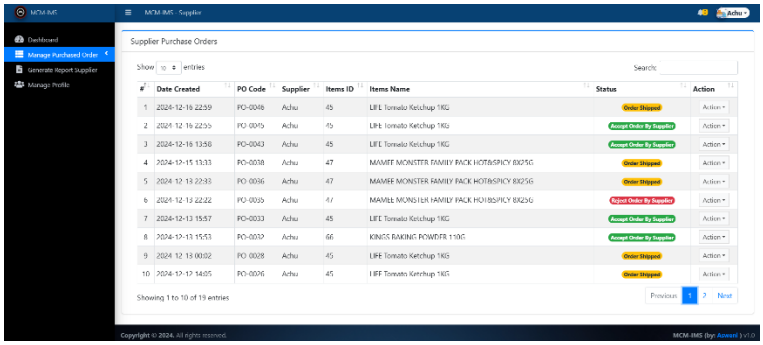


Fig. 15 *Manage Purchase Order interface*

Fig. 16(a) shows the front-end code for purchase orders provides a visual interface for suppliers to interact with, presenting order requests and allowing them to accept, reject, or dispatch orders. It also let's check order history and status, and it uses HTML, CSS, and JavaScript to manage tables and communicate with users. Fig. 16(b) shows the purchase order back-end code processes order data using server-side languages such as PHP, updating the database when orders are accepted, denied, or sent, and keeping track of order history.

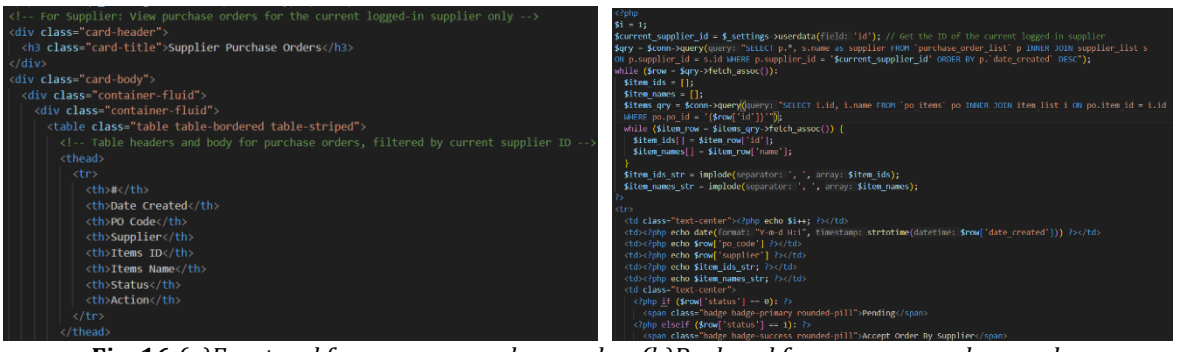


Fig. 16 *(a)Front end for manage purchase order; (b)Back end for manage purchase order*

5.1.5 Payment Interface

Fig. 17 shows the payment options available once a provider accepts an order. It offers a list of pending payments and allows the manager to pay with cash or a card. When the payment is complete, the manager can obtain the receipt in Excel format.

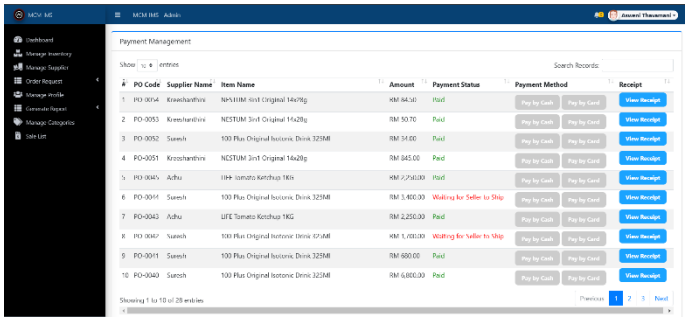


Fig. 17 *Payment interface*

The front-end code for payment in Fig. 18(a) generates a user interface for managing payments, displaying payment options (cash or card), and including a button to download the receipt. The back-end code for payment in Fig. 18 (b) processes the payment, updates the database, and generates the receipt through Excel.

```
<div class="card card-outline card-primary">
    <div class="card-header">
        <h3 class="card-title">Payment Record</h3>
    </div>
    <div class="card-body">
        <div class="container-fluid">
            <table class="table table-bordered table-striped">
                <thead>
                    <tr>
                        <th>#</th>
                        <th>PO Code</th>
                        <th>Payment Date</th>
                        <th>Amount</th>
                        <th>Payment Method</th>
                        <th>Payment Status</th>
                        <th>Action</th>
                    </tr>
                </thead>
                <tbody>
```

```
<?php
require __DIR__ . "/vendor/autoload.php";

$stripe_secret_key = getenv(name: 'STRIPE_SECRET_KEY') ?:
"sk_test_51QMRCxFAx7eOE6VIeX9qYdQNYYagR4wXhsbJexdfzxWMvMlNTDdz0lrRvaSImhdtetnkbaFr2HGnCcUqOVYhYF7r00FGNKG7vW";
\Stripe\Stripe::setApiKey($stripe_secret_key);

if (isset($_GET['po_id']) && is_numeric(value: $_GET['po_id'])) {
    $po_id = (int) $_GET['po_id'];

    $conn = new mysqli(hostname: "localhost", username: "root", password: "", database: "mcm_db");
    if ($conn->connect_error) {
        die("Database connection failed: " . $conn->connect_error);
    }

    $qry = $conn->prepare(query: "
        SELECT p.amount, s.name AS supplier_name, p.po_code
        FROM purchase_order_list p
        INNER JOIN supplier_list s ON p.supplier_id = s.id
        WHERE p.id = ?
    ");
    $qry->bind_param(types: "i", var: &$po_id);
    $qry->execute();
    $result = $qry->get_result();
```

Fig. 18 *(a)Front end for payment; (b)Back end for payment*

5.1.6 Sales List Interface

Fig. 19(a) displays the sales list interface presenting a detailed list of all sales transactions, including sales id, products sold, and total amount. This enables users to conveniently examine and manage sales records. In Fig. 19(b), the code segment of sales list refers to the underlying code that powers the interface and oversees retrieving sales data from the database and showing it on the user interface. The code contains both front-end elements (HTML, CSS, JavaScript) for displaying data and back-end logic (PHP or equivalent) for processing and retrieving sales data.

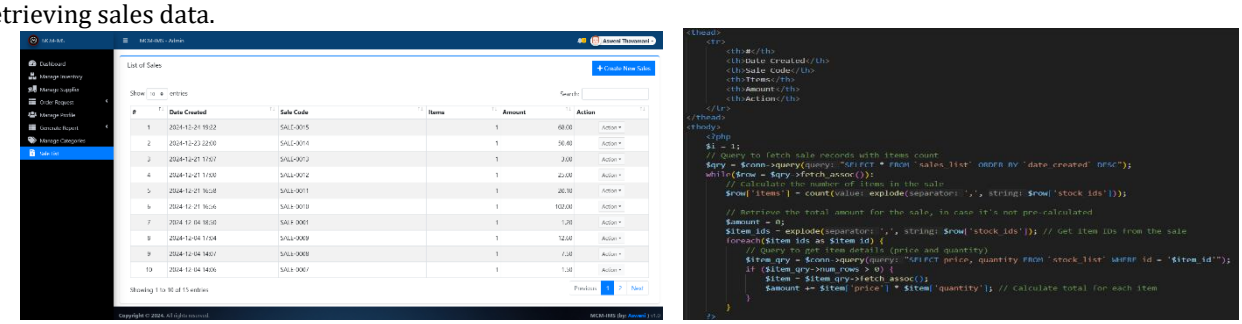


Fig. 19 *(a)Sales List interface; (b)Code segment of Sales List*

5.1.7 Generate Report Order List Interface

The generated report for order list in Fig. 20(a) enables users to create detailed order reports, displaying order IDs, supplier details, items, quantities, and totals for specific time periods. Fig. 20(b) shows the code that processes data from the database and transforms it into Excel reports, including both front-end and back-end components to facilitate report generation and download.

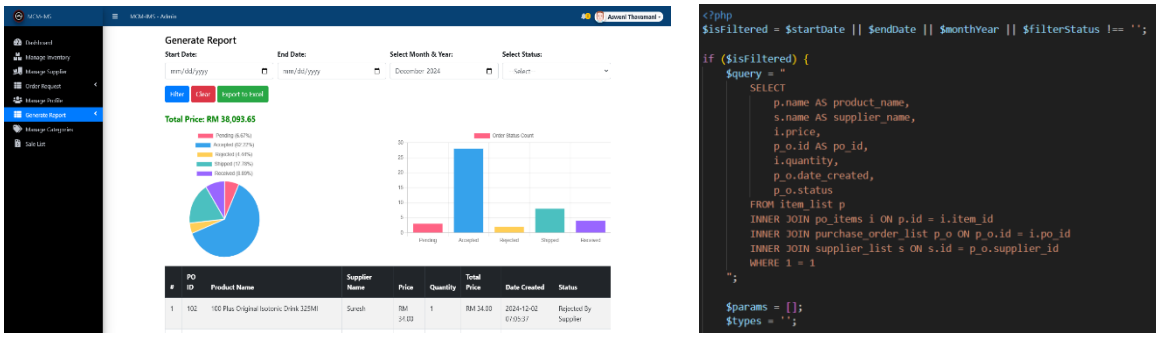


Fig. 20 *(a)Generate Report Order List Interface; (b)Code segment of Generate Report*

5.2 Testing

The testing phase is an important process for evaluating the performance and quality of the system. It involves functional testing and user acceptance testing to provide a comprehensive evaluation (Jindal, 2016). The main objective of system testing is to improve the reliability and usability of the system while minimizing risks and meeting all user needs. Functional testing refers to the process of testing the functionality of a system to ensure it works as expected and meets user requirements. It focuses on testing each module and function within the system. Functional testing uses test cases to evaluate the system based on functional requirements. It aims to identify errors in functional requirements, such as incorrect calculations, incomplete system features, or inconsistent user interfaces [9]. Table 5 shows test cases for all modules.

Table 5 *List of Test Cases*

Test Cases	Description	Test Result (Pass/Fail)
TEST_100_001	Verify that the manager can log in successfully.	Pass
TEST_100_002	Verify that a supplier can log in successfully.	Pass
TEST_100_003	Verify that the system displays an error message for a non-existent account.	Pass
TEST_100_004	Verify that the system displays an error message for an incorrect password.	Pass
TEST_100_005	Verify that the system displays a pop-up message in case of a system error during login	Pass
TEST_100_006	Verify that the login page is accessible.	Pass
TEST_200_001	Verify that a new item can be added with all required details	Pass
TEST_200_002	Verify that mandatory fields are validated when adding new item information.	Pass
TEST_200_003	Verify that existing item information can be updated.	Pass
TEST_200_004	Verify that existing item information can be deleted.	Pass
TEST_200_005	Verify that item information can be searched	Pass
TEST_200_006	Verify the system behavior when adding new item information fails.	Pass
TEST_200_007	Verify the system behavior when updating item information fails	Pass
TEST_200_008	Verify the system behavior when deleting item information fails.	Pass
TEST_200_009	Verify the system behavior when no matching item information is found during a search.	Pass
TEST_300_001	Verify that manager can navigate to the Manage Supplier module.	Pass
TEST_300_002	Verify that the manager can view a list of current suppliers.	Pass
TEST_300_003	Verify that a new supplier can be added with all required details.	Pass
TEST_300_004	Verify that mandatory fields are validated when adding new supplier information.	Pass
TEST_300_005	Verify that existing supplier information can be updated	Pass
TEST_300_006	Verify that existing supplier information can be deleted.	Pass
TEST_300_007	Verify that supplier information can be searched	Pass
TEST_300_008	Verify the system behavior when adding new supplier using same email	Pass
TEST_300_009	Verify the system behavior when adding new supplier information fails	Pass
TEST_300_010	Verify the system behavior when updating supplier information fails.	Pass
TEST_300_011	Verify the system behavior when deleting supplier list fails.	Pass

Table 5 *(continued)*

TEST_300_012	Verify the system behavior when no matching supplier information is found during a search.	Pass
TEST_400_001	Verify that manager and supplier can navigate to the manage profile	Pass
TEST_400_002	Verify that manager and supplier can modify profile details.	Pass
TEST_400_003	Verify that manger can add new admin and manage admin profile	Pass
TEST_400_004	Verify the system behavior when adding new admin information fails.	Pass
TEST_400_005	Verify the system behavior when updating profile information fails.	Pass
TEST_400_006	Verify that only authorized manager or supplier can access the manage profile	Pass
TEST_500_001	Verify that manager can navigate to the order request module	Pass
TEST_500_002	Verify that manager can view a list of current order request	Pass
TEST_500_003	Verify that manager can make an order request	Pass
TEST_500_004	Verify that manager can modify order request details.	Pass
TEST_500_005	Verify that staff can delete an order request.	Pass
TEST_500_006	Verify that order request information can be search	Pass
TEST_500_007	Verify the system behavior when updating order request information fails.	Pass
TEST_600_001	Verify that supplier can navigate to the manage purchase order module	Pass
TEST_600_002	Verify that supplier can view a list of current bookings	Pass
TEST_600_003	Verify that supplier receives alert about pending orders	Pass
TEST_600_004	Verify that supplier can accept or reject the order.	Pass
TEST_600_005	Verify that the manager is notified when the order accepts or reject	Pass
TEST_600_006	Verify the system behavior when updating order status	Pass
TEST_700_001	Verify that the manager can access the payment section.	Pass
TEST_700_002	Verify that the manager can select the cash payment method	Pass
TEST_700_003	Verify that the manager can select the card payment method.	Pass
TEST_700_004	Verify that the system allows the manager to input valid card details	Pass
TEST_700_005	Verify that the system rejects invalid card details.	Pass
TEST_700_006	Verify system behavior when the payment by card transaction fails	Pass
TEST_700_007	Verify that the system provides a payment success message after card payment.	Pass
TEST_800_001	Verify that the system displays the sales list page correctly.	Pass
TEST_800_002	Verify that the system shows all sales records accurately.	Pass
TEST_800_003	Verify that the supplier can search for a specific sales record.	Pass
TEST_800_004	Verify the system behavior when there are no sales records available.	Pass
TEST_800_005	Verify that clicking on a sales record shows detailed information about it.	Pass
TEST_900_001	Verify the manager or supplier can access the report generation module.	Pass
TEST_900_002	Verify the system handles errors gracefully during report generation.	Pass
TEST_900_003	Verify the staff can use various filtering options to customize the report.	Pass
TEST_1000_001	Verify that the system sends a notification when a new order is placed.	Pass
TEST_1000_001	Verify that the system sends a notification when an order has been accepted or rejected by supplier.	Pass
TEST_1000_001	Verify that the system sends an email containing the supplier's username and password upon account creation.	Pass

The login feedback in Fig. 21(a) offers smooth and fast user experience, enabling effortless login with a username and password while promptly alerting users to incorrect credentials and providing a simple password reset process. The page is quick to load, accessible, and delivers a seamless experience, however, two evaluators rated its usability and performance 67% as "Strongly Agree". Similarly, manage inventory in Fig. 21(b) display feedback by allowing manager to easily add, update, and search for items efficiently, enhancing productivity. While most users rated its performance 56% as "Strongly Agree," a few rated it 44% as "Agree" for ease of use and efficiency. APPENDIX C.1(a) and APPENDIX C.1(b) display the sample questions used during the evaluation process to gather feedback on system performance and usability.

Fig. 21 *(a)Feedback for Login module; (b)Feedback for Manage Inventory module*

The manage supplier feedback in Fig. 22(a) provides an efficient way to organize supplier information, allowing users to sort suppliers by products, and edit details quickly. It includes a confirmation prompt before deleting a supplier, preventing accidental removals, and receiving positive feedback, with users rating it "Strongly Agree" as 67% for ease of use and functionality. Similarly, the manage profile feedback in Fig. 22(b) offers a user-friendly interface for viewing and updating profile information. Its simplicity and functionality were well received, with most users rating it 78% as "Strongly Agree." APPENDIX C.2(a) and APPENDIX C.2(b) display the sample questions used during the evaluation process to gather feedback on system performance and usability.

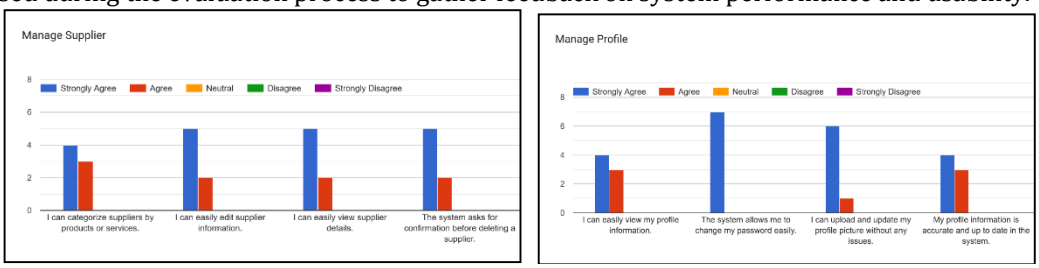


Fig. 22 *(a)Feedback for Manage Supplier module; (b)Feedback for Manage Profile module*

The order request feedback in Fig. 23(a) simplifies the process of placing orders to replenish inventory, providing users with clear confirmation after submission to keep them informed of their request's status. It is user-friendly and received positive feedback for its effectiveness and efficiency. Similarly, the manage purchase order feedback in Fig. 23(b) enhances supplier efficiency by maintaining a complete record of purchase orders, including their status, ensuring transparency and up-to-date information. Users praised its simplicity and functionality, with most rating it "Strongly Agree," while a few rated it as "Agree" for dependability and overall performance. APPENDIX C.3(a) and APPENDIX C.3(b) display the sample questions used during the evaluation process to gather feedback on system performance and usability.

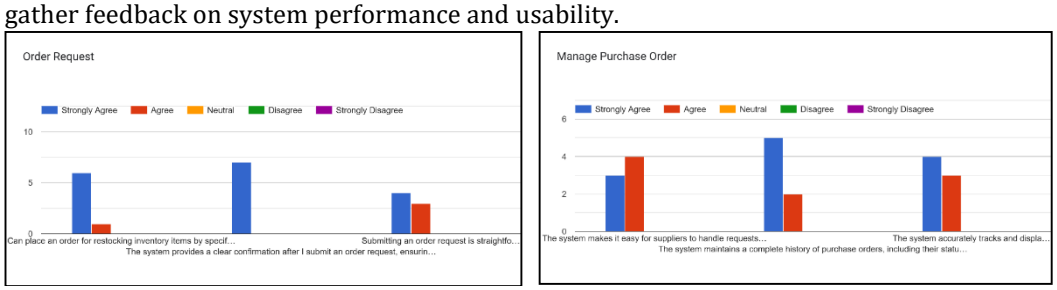


Fig. 23 *(a)Feedback for Order Request module; (b)Feedback for Manage Purchase Order module*

The payment module Fig. 24(a) allows managers to easily access the payment area and switch between various payment options. It ensures a smooth and efficient process, displaying accurate error messages when payments fail, keeping users informed. Similarly, the generate report module Fig. 24(b) provides quick access to modify and view reports with accurate and useful data. Both modules are user-friendly and highly functional, earning a "Strongly Agree" and 68% rating from users for their convenience and usability.

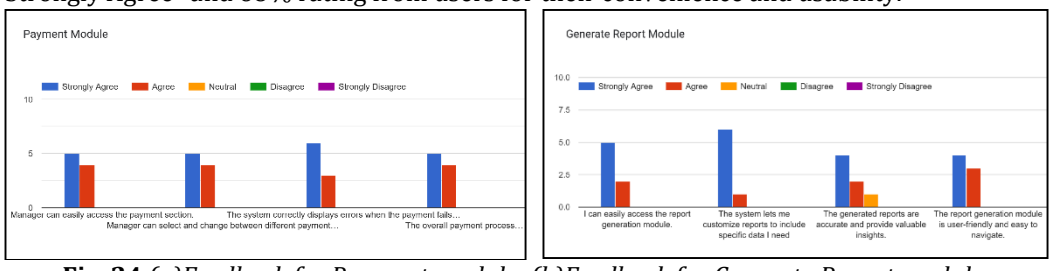


Fig. 24 *(a)Feedback for Payment module; (b)Feedback for Generate Report module*

6. Conclusion

The MCM Inventory Management System successfully met its overall objectives. Firstly, an object-oriented approach was used to ensure modularity, scalability, and maintainability of the system. Secondly, front-end and back-end technologies were used in the development of the web application, creating a responsive and user-friendly interface. Lastly, the system underwent user acceptance testing to ensure that it satisfies user requirements and performs reliably under various conditions. These achievements highlight the system's dependability, usability, and user-centric design.

The MCM Inventory Management System enhances efficiency by automating tasks like tracking supplies and managing orders, reducing errors and improving decision-making. It provides a user-friendly platform for managers and suppliers and includes detailed reporting modules for data-driven decisions. However, it has some drawbacks, such as security risks if users fail to log out, vulnerability to cyber-attacks, and occasional inaccuracies in real-time tracking. For future enhancements, integrating a barcode reader can automate stock updates, reducing errors. Adding RFID technology enables real-time tracking, while connecting to a POS system streamlines stock management.

In summary, the MCM Inventory Management System highlights the project's achievements in automating processes and centralizing data to improve inventory management and user experience. It addresses challenges encountered, particularly in overcoming the inefficiencies of manual processes, and reviews the system's strengths in enhancing efficiency alongside its weaknesses. User acceptance testing results are analyzed, offering insights that guide recommendations for future improvements.

Acknowledgment

The authors express their gratitude to the Faculty of Computer Science and Information Technology at University Tun Hussein Onn Malaysia for their invaluable assistance and motivation during the study's conduct.

Conflict of Interest

Authors declare that there is no conflict of interest regarding the publication of the paper.

Author Contribution

This journal requires that all authors take public responsibility for the content of the work submitted for review. The contributions of all authors must be described in the following manner:

The authors confirm contribution to the paper as follows: **study conception and design**: Asweni A/P Thavamani, Rabatul Aduni Binti Sulaiman; **data collection**: Asweni A/P Thavamani, Rabatul Aduni Binti Sulaiman; **analysis and interpretation of results**: Asweni A/P Thavamani, Rabatul Aduni Binti Sulaiman; **draft manuscript preparation**: Asweni A/P Thavamani, Rabatul Aduni Binti Sulaiman. All authors reviewed the results and approved the final version of the manuscript.

References

[1] Inventory Management Software – Stock control. (n.d.). Xero. https://www.xero.com/my/accounting-software/manage-inventory/
[2] The #1 open-source inventory Management | ODOO. (n.d.). Odoo. https://www.odoo.com/app/inventory
[3] Zoho Corporation Pvt Ltd. (n.d.). Online inventory management software | Zoho Inventory. https://www.zoho.com/inventory/
[4] TradeGecko Inventory Management Software - Chrome Web Store. (n.d.). https://chromewebstore.google.com/detail/tradegecko-inventory-mana/gecdhnolllllammeoogdgaaebfmcfloo?hl=en
[5] Brush, K. (2022, November 28). use case. Software Quality. https://www.techtarget.com/searchsoftwarequality/definition/use-case
[6] Aryal, S. (2019). Bootstrap: a front-end framework for responsive web design.
[7] Jacobson, I., Christerson M., Jonsson P., Övergaard G., (1992). Object-Oriented Software Engineering - A Use Case Driven Approach, Addison-Wesley.
[8] Gemino, A., Parker, D. (2009) "Use case diagrams in support of use case modeling: Deriving understanding from the picture", Journal of Database Management, 20(1), 1-24.
[9] Arrelid, D., & Backman, S. (2012). How to manage and improve inventory control
[10] Inventory Management System Sequence UML Diagram | Academic Projects. (2018, January 31). https://www.freeprojectz.com/uml-diagram/inventory-management-system-sequence-diagram

Appendix A

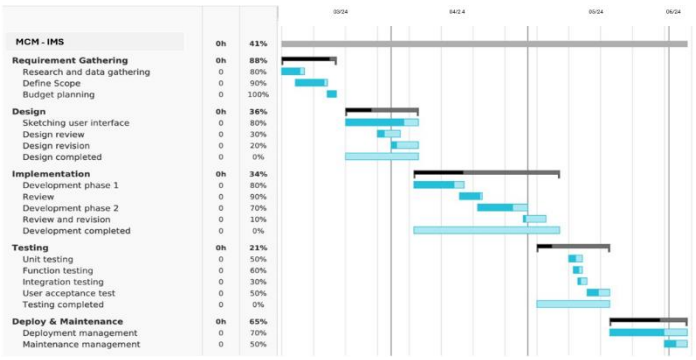


Fig. A.1 *Gantt Chart*

Appendix B

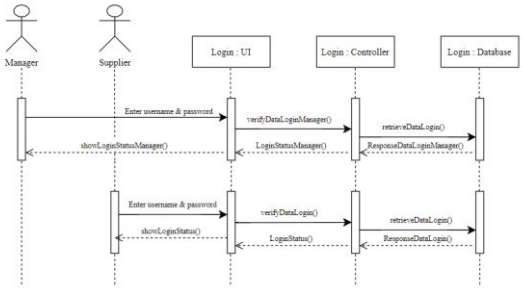


Fig B.1 *Sequence Diagram For Login*

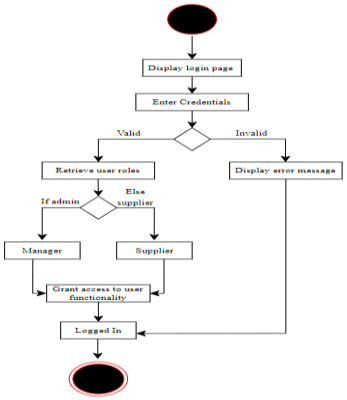


Fig B.2 *Activity Diagram for Login*

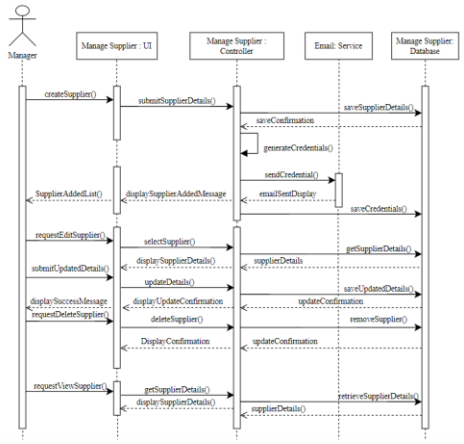


Fig B.3 *Sequence Diagram for Manage Supplier*

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 1145-1164

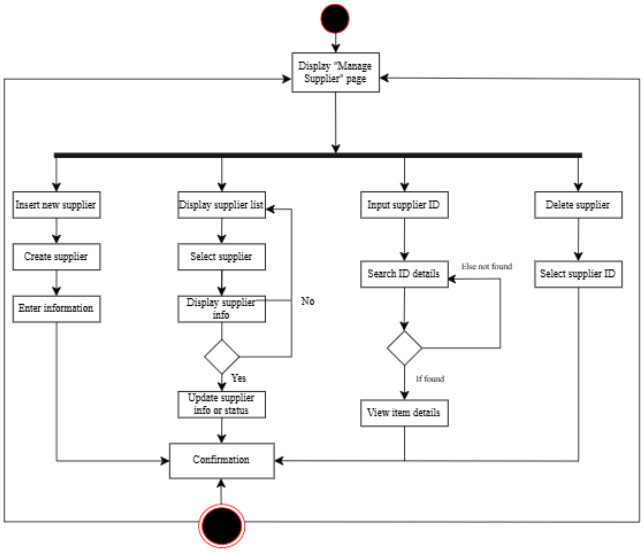


Fig B.4 *Activity Diagram for Manage Supplier*

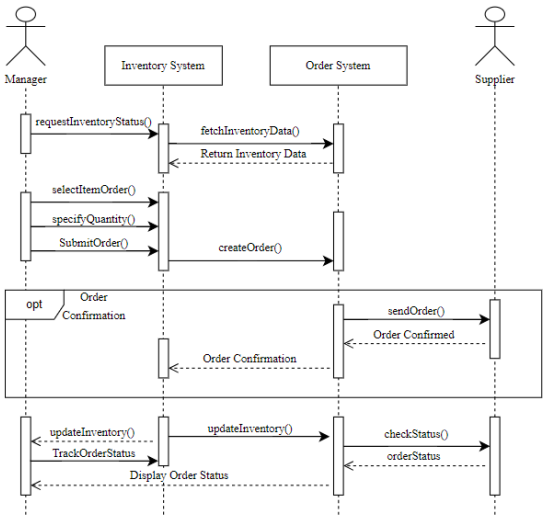


Fig B.5 *Sequence Diagram for Order Request*

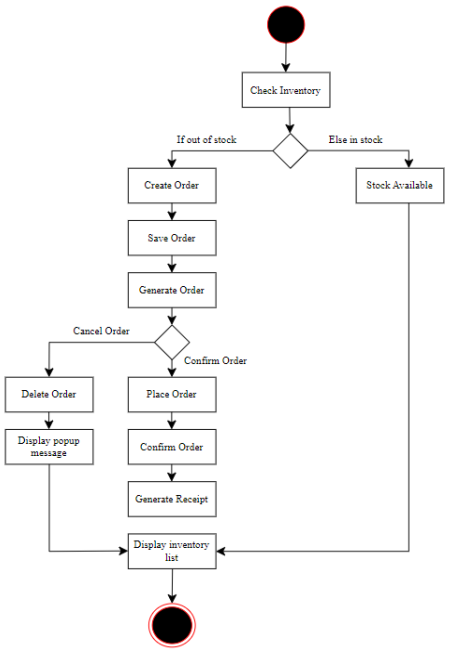
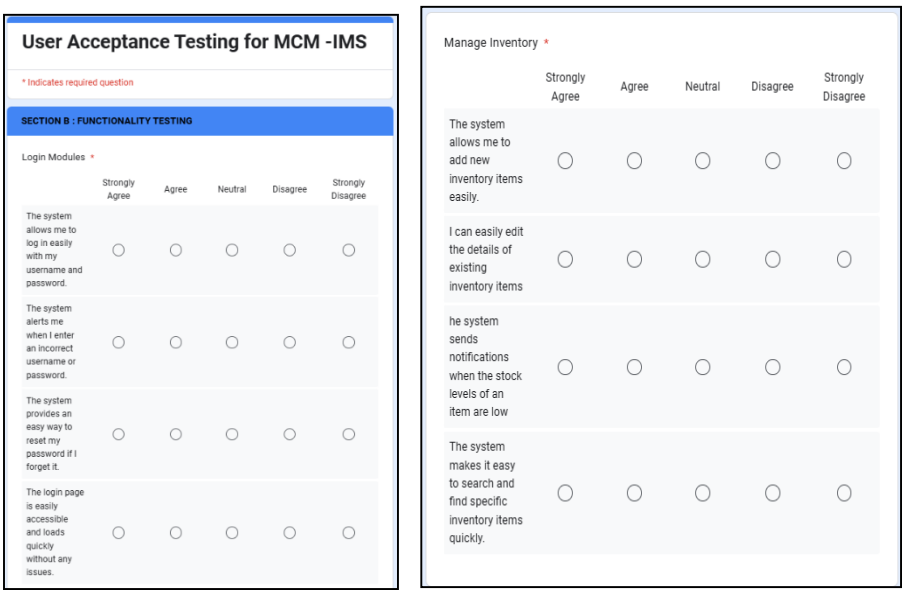


Fig B.6 *Activity Diagram for Order Request*

Appendix C



User Acceptance Testing for MCM -IMS

* Indicates required question

SECTION B : FUNCTIONALITY TESTING

Login Modules *

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
The system allows me to log in easily with my username and password.					
The system alerts me when I enter an incorrect username or password.					
The system provides an easy way to reset my password if I forget it.					
The login page is easily accessible and loads quickly without any issues.					

Manage Inventory *

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
The system allows me to add new inventory items easily.					
I can easily edit the details of existing inventory items					
he system sends notifications when the stock levels of an item are low					
The system makes it easy to search and find specific inventory items quickly.					

Fig C.1 (a) *Sample Google Form for Login Modules; (b) Sample Google Form for Manage Inventory*

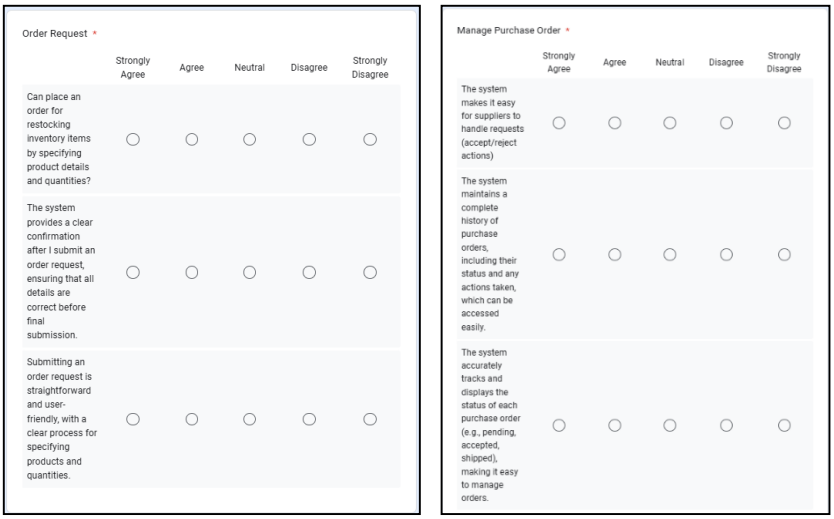
Manage Supplier *

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
I can categorize suppliers by products or services.					
I can easily edit supplier information.					
I can easily view supplier details.					
The system asks for confirmation before deleting a supplier.					

Manage Profile *

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
I can easily view my profile information.					
The system allows me to change my password easily.					
I can upload and update my profile picture without any issues.					
My profile information is accurate and up to date in the system.					

Fig C.2 (a) *Sample Google Form for Manage Supplier; (b) Sample Google Form for Manage Profile*



Order Request *

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Can place an order for restocking inventory items by specifying product details and quantities?					
The system provides a clear confirmation after I submit an order request, ensuring that all details are correct before final submission.					
Submitting an order request is straightforward and user-friendly, with a clear process for specifying products and quantities.					

Manage Purchase Order *

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
The system makes it easy for suppliers to handle requests (accept/reject actions)					
The system maintains a complete history of purchase orders, including their status and any actions taken, which can be accessed easily.					
The system accurately tracks and displays the status of each purchase order (e.g., pending, accepted, shipped), making it easy to manage orders.					

Fig C.3 (a) *Sample Google Form for Order Request; (b) Sample Google Form for Manage Purchase Order*