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

AITCS

Vol. 6 No. 1 (2025) 576-592
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Tailoring Order Management System with Data Retention and Role-Based Access System

Ku Nur Hanis Ku Azman[1], Nordiana Rahim[1]*

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

*Corresponding Author: *nordiana@uthm.edu.my*
DOI: https://doi.org/10.30880/aitcs.2025.06.01.032

Article Info

Received: 9 May 2025
Accepted: 19 June 2025
Available online: 30 June 2025

Keywords

Data Retention, Role-Based Access Control, RBAC, Order Management, Laravel, Agile

Abstract

The Tailoring Order Management System with Data Retention and Role-Based Access Control (RBAC) was developed to help small tailoring businesses address issues with manual order management. This project aimed to digitize processes, improve data organization, and secure customer information for Nadimah Tailor that based in Kuala Lipis, Pahang. The Agile development methodology was used to create a web-based system with features like secure login, customer and staff management, order tracking, and automated data retention policies. System testing showed that the modules performed well, ensuring secure access, efficient order tracking, and automatic data removal after the retention period. These features enhanced productivity and minimized risks of data mismanagement. The system successfully streamlined operations and provided a user-friendly interface for staff and customers. Future work could explore integrating advanced analytics and expanding functionalities to further optimize tailoring operations.

1. Introduction

The tailoring industry in Malaysia, especially for small businesses like Nadimah Tailor in Kuala Lipis, Pahang, plays an important role in making traditional clothes like *baju kurung*. Nadimah Tailor started in 2012, offering sewing services mostly to family members, especially during festive seasons like Eid. The business grew after receiving a sewing machine from a government workshop in 2022, allowing them to handle more orders. However, despite its growth, the business still uses manual methods like WhatsApp for communication and paper records for keeping customer details and orders.

This manual way of working causes many problems. Customer records, which include measurements and design details, are often unorganized or misplaced. This leads to confusion when handling repeat orders and takes up valuable space in the tailor's work area. Managing orders through WhatsApp makes it difficult to track and match specific customer orders. Manual record-keeping methods can be inefficient and lead to issues like data loss and disorganization [1].

To solve these problems, this study aims to develop a Tailoring Order Management System with Data Retention and Role-Based Access Control (RBAC). The system will help digitize the business operations of Nadimah Tailor and improve the way customer data is managed. The main objectives of this project are:

- To design a system that includes Data Retention policies and Role-Based Access Control for Nadimah Tailor.
- To develop a web-based system that makes customer and order management more efficient while improving security through features like RBAC and Data Retention.

- To test the system's performance to ensure it works smoothly and meets the business' needs.

The system's scope includes key modules to handle customer and order management effectively. Customers' measurements and order details will be stored digitally, making it easier to organize and access. Staff and admin users will have specific roles and access based on their responsibilities, ensuring secure management of data. The system will also include a Data Retention feature to automatically delete customer data after 400 days, which suits the tailoring cycle as customers often return annually during Eid.

By replacing manual processes with this digital solution, Nadimah Tailor can better manage its growing customer base, reduce errors, and improve efficiency. The system also helps protect customer data and ensures secure access for authorized users. This project aims to make daily operations easier for the business and provide better service to customers. In the future, the system could be expanded with features like analytics or mobile app integration to support further business growth.

2. Literature Review

The Tailoring Management System is specifically designed to streamline operations in the tailoring business, covering essential tasks such as managing customer orders, measurements, design selections, and order statuses. It includes various modules such as order management, customer management, inventory management, and reporting. These modules ensure smooth operations, minimize errors, and improve customer satisfaction. The integration of an efficient tailoring management system can significantly enhance service quality and operational efficiency [1]. Furthermore, it offers features to help tailors manage extensive customer data and monitor the status of orders effectively.

An extension of the Tailoring Management System includes an online platform, which provides convenience for both customers and tailors. An online system allows customers to place orders, provide measurements, and track order statuses remotely [2]. By automating much of the manual work, this platform helps reduce errors, improves productivity, and provides valuable business insights such as customer preferences and sales trends.

The Order Management System integrates several modules like order entry, inventory management, order processing, and customer service to ensure that orders are processed efficiently. The Order Management System provides a seamless and responsive order management process, enhancing operational efficiency and improving customer experience [3]. Real-time inventory tracking, automated order processing, and efficient customer service support contribute to better order fulfillment and higher customer satisfaction.

Security is a crucial aspect of digital systems, particularly those handling sensitive customer data [1]. The security element modules within the system implement robust authentication mechanisms and data protection strategies to prevent unauthorized access and data breaches. Ensuring data security is key to building customer trust and maintaining privacy standards.

2.1 Data Retention

Data retention policies play an essential role in managing the lifecycle of sensitive information. These policies define the duration for which data is retained, how it is protected, and when and how it should be disposed of. Implementing clear data retention strategies helps organizations comply with legal and regulatory standards, manage data efficiently, and reduce risks associated with storing sensitive information [4]. Different types of data retention policies exist, including regulatory, operational, historical, and backup retention policies. For example, financial data must often be kept for several years to meet regulatory requirements, while backup data ensures business continuity during unexpected incidents like data loss or cyberattacks.

Data retention policies should be tailored to ensure that only necessary data is stored and that its handling complies with privacy laws [5]. The management of personal data has also become a key focus, with regulations such as the General Data Protection Regulation (GDPR) emphasizing the need to retain user data for a limited period. Businesses nowadays can implement historical data retention for long-term analysis, such as tracking customer trends, and backup retention to ensure that organizations are prepared for unforeseen events [6] [7].

Additionally, organizations must ensure that their data retention practices adhere to legal compliance and security standards. For instance, financial institutions must maintain detailed transaction records for auditing purposes for many years. Organizations need to balance regulatory compliance with minimizing risks associated with data exposure [8], while secure data disposal after the retention period ends is important to avoid data breaches and unauthorized access [9].

2.2 Role-Based Access Control

Role-Based Access Control (RBAC) is a popular security model used to regulate access to system resources based on users' roles within an organization. RBAC simplifies the process of managing user access by assigning permissions to roles, rather than to individual users, which improves operational efficiency and enhances security [10]. Users are assigned specific roles based on their job functions, ensuring that they have access only

to the resources needed for their responsibilities. This system follows the principle of least privilege, limiting access to sensitive data and reducing the risk of data breaches.

RBAC is particularly useful in cloud environments where decentralization enhances security and trust [11]. RBAC models provide flexibility by adapting to different organizational structures, and they help ensure compliance with regulatory standards. A dynamic RBAC model for decentralized applications, which allows for more adaptable and secure access control in distributed environments [12]. The flexibility and security provided by RBAC make it an essential tool in modern information security systems.

However, implementing RBAC can be challenging. The system requires a thorough understanding of organizational roles and regular updates to keep up with changing responsibilities [10]. Nonetheless, the ongoing development of dynamic RBAC models continues to enhance the scalability and effectiveness of access control systems, making it a widely adopted and efficient approach for managing security in contemporary IT systems. Continuous innovation and the adaptation of RBAC in dynamic environments are critical to maintaining robust and scalable access control systems [13].

2.3 Comparison of System

This section looks at three tailoring management systems; SmartMaster, Indochino, and The Cayman Tailor to understand their features, user interfaces, and security measures. It also includes a comparison with the Tailoring Order Management System for Nadimah Tailor, which focuses on data retention and role-based access control. The analysis examines key aspects such as functionality, performance, user-friendliness, and security to see how each system meets the needs of a tailoring business.

By comparing these systems, this discussion highlights useful practices and unique features that can improve the proposed system. These findings are essential for creating a reliable, easy-to-use, and secure tailoring management system for Nadimah Tailor. Table 1 shows the comparison between existing system and Tailoring Order Management System with Data Retention and RBAC for Nadimah Tailor.

Table 1 *Comparison between existing system and Nadimah Tailor*

Features	SmartMaster	Indochino	The Cayman Tailor	Nadimah Tailor
Functionality	Basic tailoring service management with limited features for customers and order tracking.	Comprehensive tailoring service management with integration of online and offline orders, and personalized services.	Advanced customer and tailoring management, emphasizing personalized service and detailed client records.	Enables staff and admins to manage customer data, body measurements, and orders. Provides unique order links for customer access without login. Tracks order status and history. Includes data retention policies for customer records based on link activation.
Performance	Simple system with moderate speed for managing orders and users.	Works smoothly and can handle many customers and services.	Optimized for high-end tailoring services with robust tracking and reporting features.	Efficient for managing customer and order details. Customers use links, making it fast and simple to access orders.
User Interface (UI)	Basic design, easy to use but not much customization.	Clean and professional design, easy for both staff and customers.	Premium design for smooth customer experience.	Simple dashboards for staff and admin to manage everything. Customers access their orders directly using unique links.
Security	Standard authentication mechanisms (user ID and password). Limited security layers.	Includes basic security measures with email authentication for staff and customers.	Use basic security, making it easier for users to access the service but with fewer advanced protections in place.	Role-Based Access Control: Admins have full system of access, staff manage customers and orders, and customers access specific orders via unique links. Data retention policies are

	implemented for customer records.

In conclusion, the comparison of tailoring management systems highlights their unique strengths and weaknesses in functionality, performance, user interface, and security. SmartMaster is suitable for small businesses but lacks advanced tools, while Indochino and The Cayman Tailor cater to larger operations with more comprehensive features. The Tailoring Order Management System with Data Retention and RBAC stands out with its focus on customer convenience, data retention, and enhanced security through role-based access control. By automatically removing customer data after the retention period, it ensures compliance with privacy laws, reduces risks of data breaches, and maintains a streamlined database. This system combines functionality, efficiency, and security, offering a reliable solution tailored to meet the needs of modern tailoring businesses.

3. Methodology

Agile model is the method that was picked to create the Tailoring Order Management System with data retention for Tukang Jahit Nadimah. The agile method focuses on iterative development. It was picked for this project because it is flexible enough to keep getting better and adapting to changes as the project grows. The Agile model works best for projects that need to be updated and improved on a regular basis to make sure the end product meets user needs. Additionally, incorporating secure development practices within the Agile framework ensures that security measures are integrated throughout the development process, enhancing system safety.

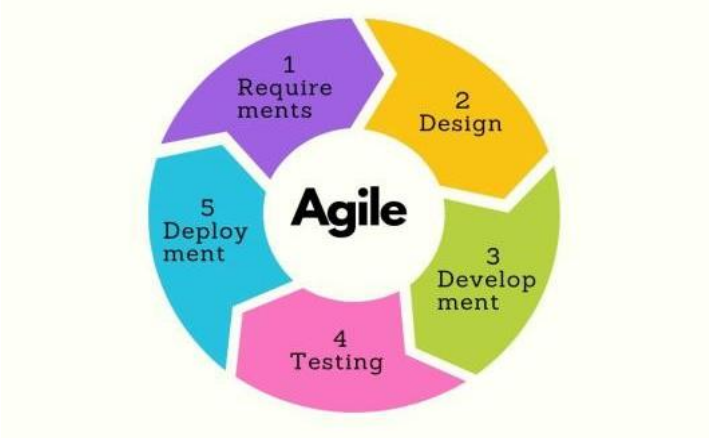


Fig. 1 *Agile Model*

3.1 Requirements Phase

The Requirements phase in Agile is essential for defining the project scope and ensuring all stakeholders align on the goals and deliverables. For Nadimah Tailor's Tailoring Order Management System with Data Retention and RBAC, this phase focuses on understanding the business needs, identifying the required functionalities, and setting a development timeline. Meetings with stakeholders, phone interviews, and brainstorming with the owner are key techniques used to gather user requirements and prioritize features. This process helps identify business challenges and improvements needed to ensure smoother operations after implementing the system.

Table 2 *Functional requirements for Tailoring Order Management System*

No	Function	Functionality
1	Registration & Log in	• Admin registers accounts for staff and customers. • Staff log in using a username and password, with first-time login requiring a password change. • Staff can reset their password if forgotten. • Customers do not need log in; they are provided with a unique order link to access their customer page.
2	Customer Management	• Admin and staff can add new customers, including their body measurements. • Admin and staff can view customer listings and track order histories.

Table 2 *(cont.)*

No	Function	Functionality
3	Order Management	• Admin and staff can create and manage orders, including setting descriptions, measurements, and charges. • Admin and staff can manage order statuses (e.g., Paid/To Collect) and generate unique order links. • Customer orders are moved to retention and order history after they are marked as paid. • Admin and staff can view order history.
4	View Order Details	• Customers can view their order details and status through unique order links. • Admin and staff can view all customer details, including measurements and order history.
5	Security measure	• The system implements data retention policies to manage customer data, automatically removing records after the retention period based on link activity. • Role-Based Access Control (RBAC) ensures: • Admin: Full access to manage staff, customers, orders, data retention period and the system. • Staff: Access to manage customers and orders. • Customer: Limited access to specific order details via unique order links

3.2 Design Phase

In this design phase, the purpose is to create detailed design of the Tailoring Order Management System with Data Retention for Nadimah Tailor. This phase translates the information from Requirements phase into a blueprint for development. It involves the creation of various Unified Modelling Language (UML) diagrams to provide visual representation of the system. The first diagrams include use case diagrams, which visually represent the interactions between users and the system. Next is the class diagram, which defines the system's structure by showing the classes, their attributes, methods, and relationships. Lastly, there is the flowchart, which outlines the step-by-step processes or workflows within the system to ensure clarity in the operational sequence.

3.2.1 Use case diagram

The use case diagram for the Tailoring Order Management System with Data Retention for Nadimah Tailor illustrates the interactions between various actors which are Admin, Staff and Customer also the system's functionalities. Each actor has specific roles and interactions. Fig 2 shows the use case diagram.

From Fig 2, it shows that the system's access is managed through Role-Based Access Control (RBAC), with Admins having the highest level of control. Admins log in with a username and password to manage security features, set data retention policies, and handle staff and customer accounts. They create staff accounts with details like username, email, and salary, as well as customer accounts with personal details and body measurements. Admins manage orders, including creating and updating them, tracking histories, and generating unique order links for customers to view order details without logging in. Staff members, logging in with credentials, focus on managing customer details, orders, and updating statuses. They assist in creating orders and generating unique links for customers. Customers access their order details via these links, ensuring ease of use. The system's design emphasizes security, with RBAC ensuring only authorized access to sensitive data and functionality.

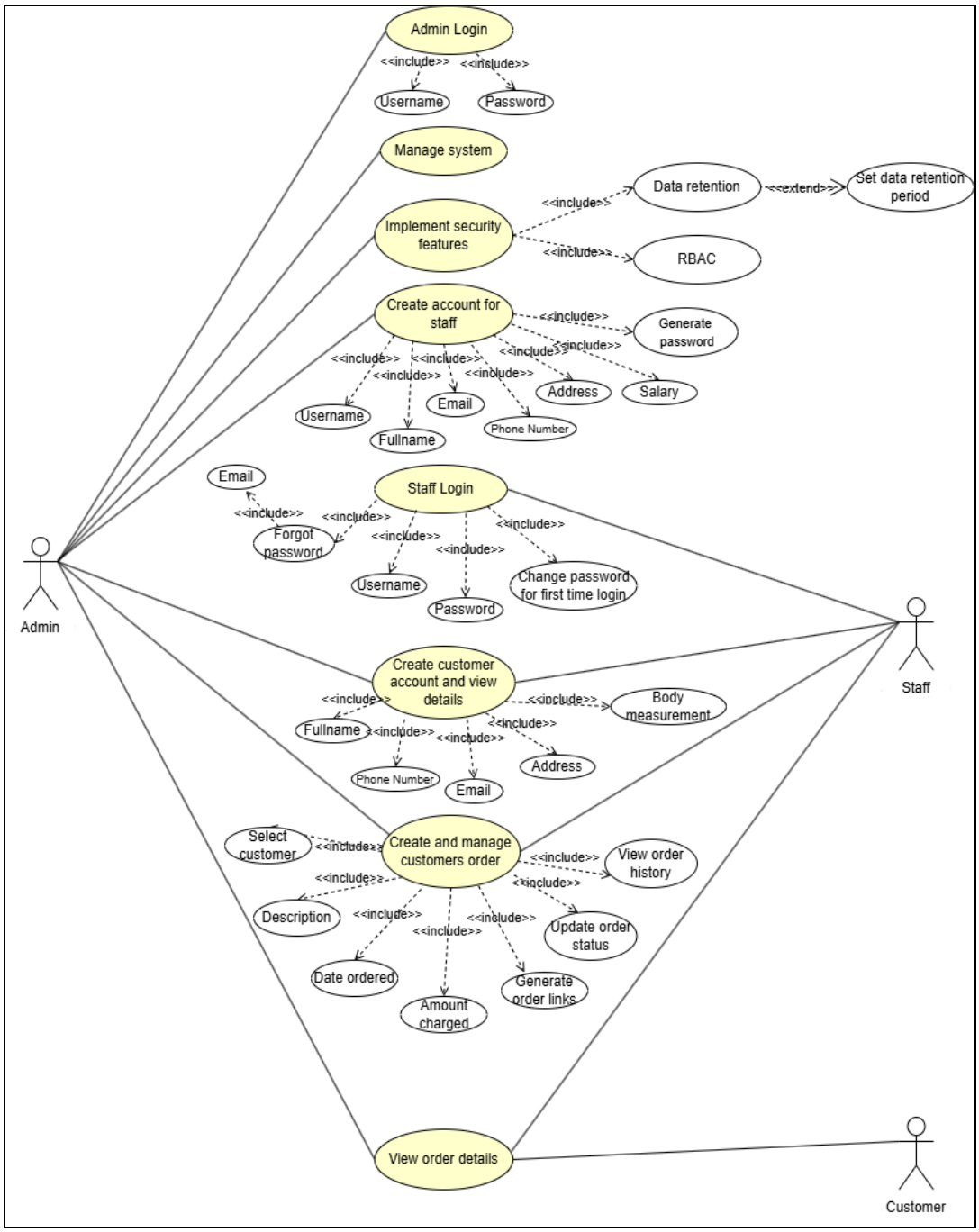


Fig. 2 *Use Case Diagram*

3.2.2 Class diagram

The class diagram for the Tailoring Order Management System with Data Retention for Nadimah Tailor illustrates the main entities and their interactions within the system. The primary entities include Admin, Staff, Customer and Order each with specific attributes and methods that define their roles and responsibilities. Fig. 3 shows the class diagram for the system. It shows that the class diagram of the Tailoring Order Management System outlines four main classes: Admin, Staff, Customer, and Order, showcasing their roles and relationships. The Admin class handles core system functions, such as managing staff, customers, orders, and implementing Role-Based Access Control (RBAC) and data retention policies. The Staff class represents employees who assist with customer and order management, including updating accounts and generating order links. The Customer class stores customer details like contact information and measurements, allowing customers to view their orders via unique order links. The Order class tracks all order-related details, such as status, measurements, and links, while managing the order lifecycle. The system demonstrates a clear hierarchy, with Admin overseeing Staff and Customers, and Customers linked to multiple Orders. This design ensures secure, efficient management of tailoring operations while supporting scalability for Nadimah Tailor's business.

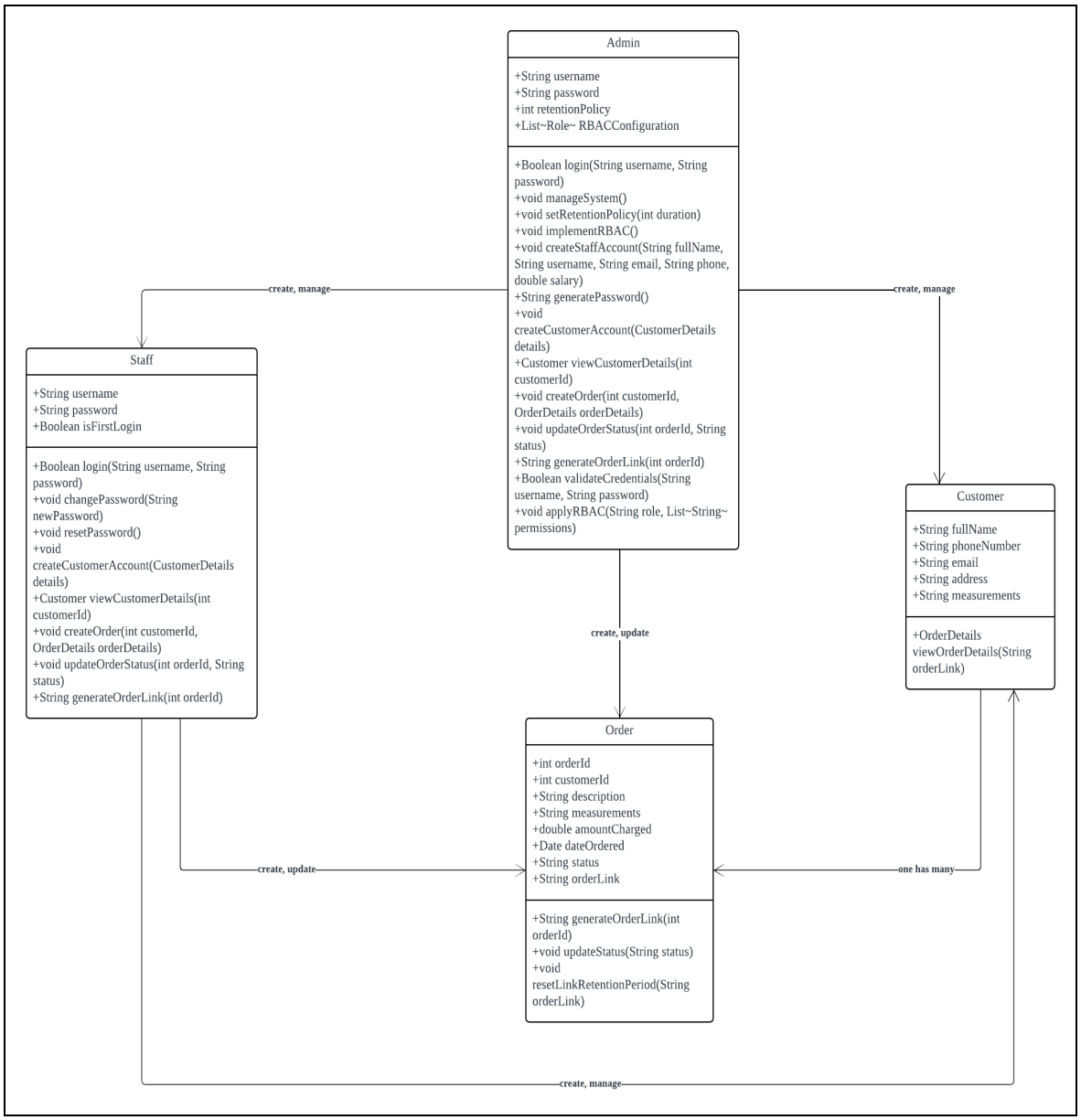


Fig. 3 *Class Diagram*

3.2.3 Flowchart

Flowchart explains the steps involved for both the admin or staff and the customers to manage and view tailoring orders effectively. This flowchart highlights key processes such as login validation, customer and order management, and the handling of unique order links for customer access. It is designed to provide a clear and structured view of the interactions and transitions between different actions in the system. The flowchart in Fig. 4 outlines the workflow of the Tailoring Order Management System, detailing processes for admins, staff, and customers. Admins and staff log in with credentials, and first-time staff are prompted to change their passwords. Once logged in, they can manage customer profiles, create orders by adding details like measurements and descriptions, and generate unique order links for customers. Customers use these links to view order details, with statuses updated by staff (e.g., "In Progress" or "To Collect"). The system also includes a retention policy: if the retention period expires, customer details are deleted, and a 404-error page is shown. Paid orders are stored in the order history for record-keeping, ensuring smooth operations and customer satisfaction.

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 576-592

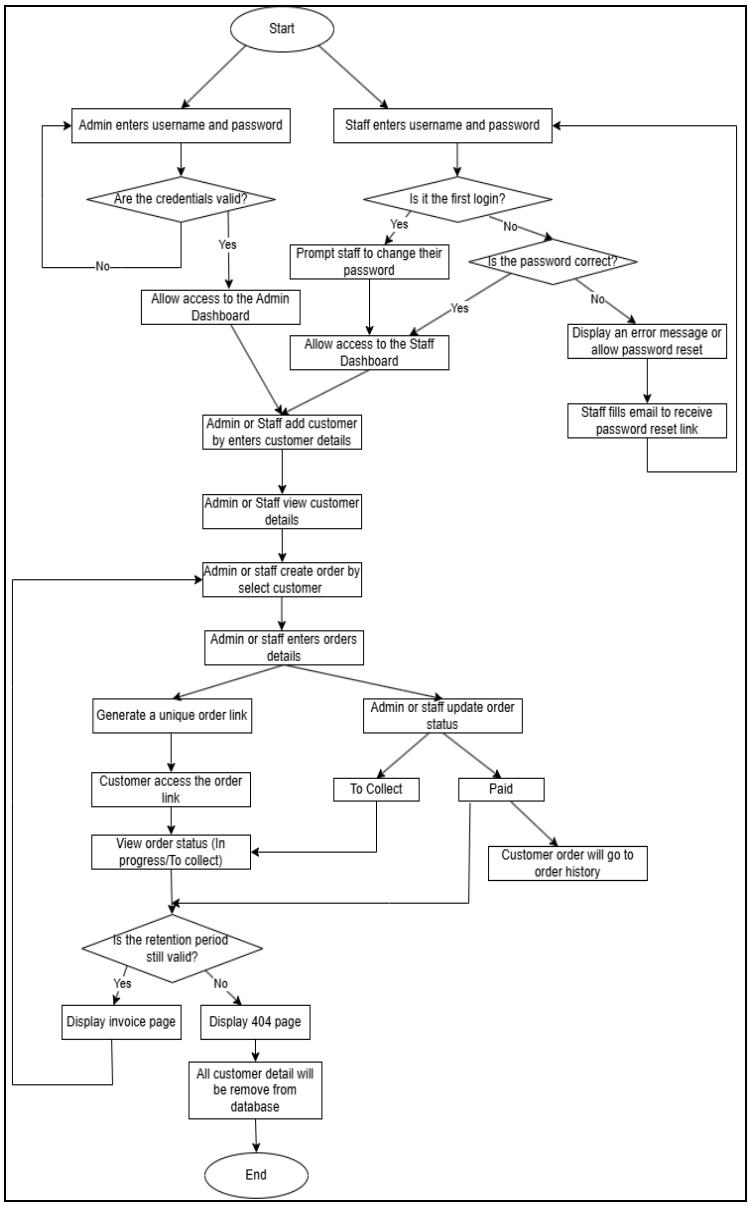


Fig. 4 *Flowchart*

3.3 Development Phase

Development Phase is the most complex and important phase for this project as it converts the detailed design into a functional system through coding and implementation. For Nadimah Tailor, this involves building the Tailoring Order Management System with Data Retention according to the specifications laid out in the design phase. The activities for this phase include setting up development environments, writing code, and integrating different component. Laravel is chosen as the development framework because it offers a robust, secure, and scalable structure with route management and the Model-View-Controller (MVC) architecture, which streamline the development process and ensure that the tailoring order management system is both structured and maintainable.

3.4 Testing Phase

In this Testing Phase, it is crucial to make sure that the system is free from bugs in other words is no error that can be found in the system to make sure that the system is established well. To make sure that the system is free form bugs, the code will be debugged to ensure it is completely clean. Any errors encountered during this process lead to immediate modifications and retesting.

3.5 Deployment Phase

Deployment Phase purpose is to make sure the Tailoring Order Management System with Data Retention and Role-based Access Control available for use by Nadimah Tailor and its customer. It involves deploying the system to a live environment and to make sure it is fully operational. Other than that, this phase also includes setting up the production environment and deploying the system.

4. Result and Discussion

This section provides a detailed analysis of the Tailoring Order Management System, focusing on the key modules and functionalities that ensure efficient operations and robust data handling. It highlights the system's ability to manage customer orders, enforce security measures, and comply with data retention policies. Additionally, it evaluates the implementation of Role-Based Access Control (RBAC) and presents testing results that validate the system's reliability, security, and user satisfaction. The insights offered aim to demonstrate the system's effectiveness in delivering seamless and secure user experience for admins, staff, and customers.

4.1 Order Management Module

The Order Management Module is central to the Tailoring Order Management System, enabling the creation, update, and detailed management of customer orders. Key functionalities include adding new orders with validation and secure image handling, updating order statuses while maintaining a comprehensive transaction history, and providing customers with secure access to order details through unique tokens. These features ensure data integrity, streamline operations for staff, and improve the overall customer experience. Through robust validation, logging, and transaction handling, the module supports an efficient workflow, enhancing communication and accuracy in order management.

4.1.1 Add Order

The code provided in Fig 5 is part of a Laravel-based system for managing customer orders within the Tailoring Order Management System. The store function handles the creation of new orders by first validating the incoming request to ensure all required data is present and meets specific criteria. It checks fields such as customer information, order description, the date the order was received, the charged amount, and optionally, an uploaded image. The image is validated for format and size before being stored in a designated directory, and the system logs all incoming data for debugging or auditing purposes.

```php
public function store(Request $request)
{
    try {
        // Validate request
        $request->validate([
            'customer' => 'required|exists:customers,id',
            'description' => 'required',
            'received_on' => 'required|date',
            'amount_charged' => 'required|numeric',
            'image' => 'nullable|image|mimes:jpeg,png,jpg,gif|max:2048'
        ]);

        \Log::info('Attempting to create order with data:', $request->all());

        // Start transaction
        return DB::transaction(function() use ($request) {
            // Handle image upload
            $imagePath = null;
            if ($request->hasFile('image')) {
                $image = $request->file('image');
                $imagePath = $image->store('order-images', 'public');
            }

            // First, check if customer has any previous orders (including deleted ones)
            $previousOrder = Orders::withTrashed()
                ->where('customer_id', $request->customer)
                ->whereNotNull('access_token')
                ->latest()
                ->first();

            // If previous order exists, use its token, otherwise generate new one
            $token = $previousOrder ? $previousOrder->access_token : (string) Str::uuid();
```

Fig. 5 *Add Order Code*

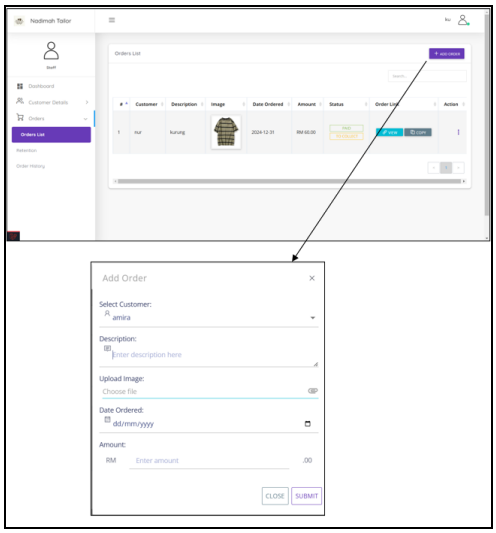


Fig. 6 *Add Order Interface*

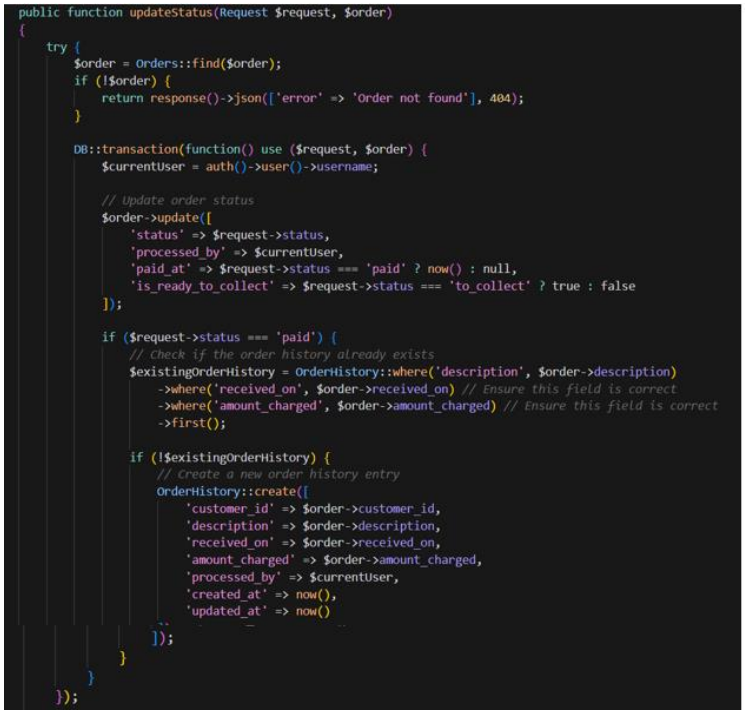
The function in Fig 5 ensures data consistency by using Laravel's transaction mechanism, which guarantees that all operations related to order creation are executed atomically. If an image is uploaded, it is saved to the order-images directory, and its path is stored for further processing. Additionally, the code checks if the customer has any previous orders, including soft-deleted ones, to determine whether an existing access_token can be reused. If no previous order exists, a new unique token is generated, ensuring that customers have secure access to their order details through a unique link.

This functionality is an integral part of the system's order management process, which streamlines operations for admins and staff while enhancing customer experience. By validating input, securely handling data, and efficiently managing orders, the system ensures a reliable and user-friendly workflow. Features like

order tracking, secure access tokens, and the ability to upload and attach images to orders help maintain accuracy and improve communication between staff and customers, making the system both robust and efficient.

4.1.2 Update Status

The code provided in Fig 7 is part of a Laravel-based system for managing and updating order statuses within an application. The updateStatus function is designed to handle updates to an order's status while ensuring data integrity and maintaining a detailed order history. It begins by attempting to retrieve the specified order using the Order::find method. If the order does not exist, the function returns a 404-error response, signaling that the requested order was not found. Once the order is validated, the function proceeds with the status update using Laravel's database transaction mechanism. This ensures that all operations are performed atomically, preventing partial updates in case of errors.



```
public function updateStatus(Request $request, $order)
{
    try {
        $order = Orders::find($order);
        if (!$order) {
            return response()->json(['error' => 'Order not found'], 404);
        }

        DB::transaction(function() use ($request, $order) {
            $currentUser = auth()->user()->username;

            // Update order status
            $order->update([
                'status' => $request->status,
                'processed_by' => $currentUser,
                'paid_at' => $request->status === 'paid' ? now() : null,
                'is_ready_to_collect' => $request->status === 'to_collect' ? true : false
            ]);

            if ($request->status === 'paid') {
                // Check if the order history already exists
                $existingOrderHistory = OrderHistory::where('description', $order->description)
                    ->where('received_on', $order->received_on) // Ensure this field is correct
                    ->where('amount_charged', $order->amount_charged) // Ensure this field is correct
                    ->first();

                if (!$existingOrderHistory) {
                    // Create a new order history entry
                    OrderHistory::create([
                        'customer_id' => $order->customer_id,
                        'description' => $order->description,
                        'received_on' => $order->received_on,
                        'amount_charged' => $order->amount_charged,
                        'processed_by' => $currentUser,
                        'created_at' => now(),
                        'updated_at' => now()
                    ]);
                }
            }
        });
```

Fig. 7 *Update Status*

Within the transaction, the function updates the order's status based on the request data. It also records the current user, identified through the auth system, as the one who processed the update. Additionally, if the status is set to paid, the paid_at timestamp is updated with the current time. For orders marked as "to collect," the function flags them as ready for collection. To further enhance accountability, the function checks if the updated status is paid and verifies whether an existing order history record already matches the current order details. This verification involves checking fields such as the order's description, received date, and charged amount. If no matching history record is found, a new entry is created in the OrderHistory table, capturing details such as the customer ID, description, received date, amount charged, and the user who processed the update.

This function plays a critical role in the system by maintaining a comprehensive history of order transactions while ensuring data consistency and accuracy. Its use of conditional logic and transaction handling prevents duplicate history entries and facilitates auditing and customer support. Overall, the updateStatus function contributes to a robust and efficient order management workflow, allowing seamless status updates and preserving detailed records for better traceability and operational reliability.

4.1.3 View Order Details

The code provided in Fig 8 is part of a Laravel-based system that handles customer order details. The view function is designed to manage customer access to order details via a unique access token. This function ensures security and data integrity while taking actions for invalid or expired access links.

```php
public function view($token)
{
    try {
        $order = Orders::where('access_token', $token)
                    ->whereNull('deleted_at')
                    ->first();

        if (!$order) {
            // Try to find order with this token to get customer ID
            $order = Orders::where('access_token', $token)
                        ->withTrashed()
                        ->first();

            if ($order) {
                try {
                    DB::beginTransaction();

                    // Disable foreign key checks
                    DB::statement('SET FOREIGN_KEY_CHECKS=0');

                    // Delete measurements
                    BodyMeasurement::where('customer_id', $order->customer_id)->delete();

                    // Delete customer
                    Customer::where('id', $order->customer_id)->delete();

                    // Re-enable foreign key checks
                    DB::statement('SET FOREIGN_KEY_CHECKS=1');

                    DB::commit();

                    Log::info('Customer data deleted after invalid link access. Order ID: ' . $order->id);
                } catch (\Exception $e) {
                    Log::error('Error deleting customer data on invalid link: ' . $e->getMessage());
                }
            }

            abort(404, 'Order not found');
        }

        // Only check expiry for paid orders
        if ($order->paid_at && $order->isLinkExpired()) {
            try {
                DB::beginTransaction();

                // Disable foreign key checks
                DB::statement('SET FOREIGN_KEY_CHECKS=0');

                // Delete measurements
                BodyMeasurement::where('customer_id', $order->customer_id)->delete();

                // Delete customer
                Customer::where('id', $order->customer_id)->delete();

                // Re-enable foreign key checks
                DB::statement('SET FOREIGN_KEY_CHECKS=1');

                DB::commit();

                Log::info('Customer data deleted after expired link access. Order ID: ' . $order->id);
            } catch (\Exception $e) {
                DB::rollBack();
                Log::error('Error deleting customer data on expired link: ' . $e->getMessage());
            }

            abort(404, 'Order link has expired');
        }
```

Fig. 8 *View Order Details code*

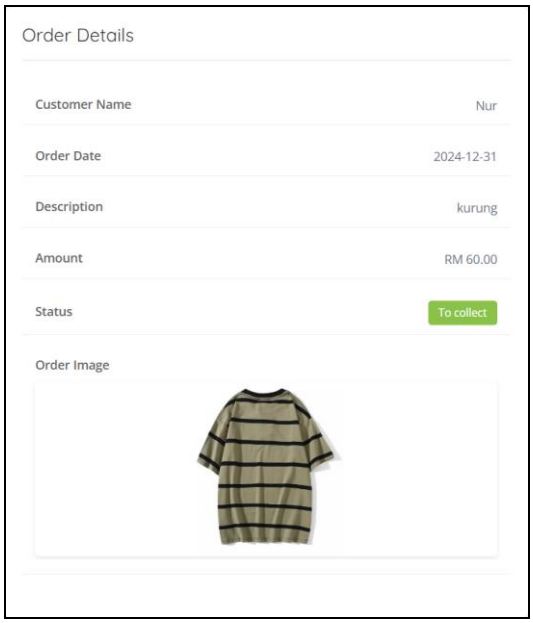


Fig. 9 *View Order Details code*

Fig 8 shows the function begins by attempting to retrieve the order associated with the provided access token using the Orders model. If no matching order is found or the order has been soft-deleted, the function returns a 404 error response, indicating that the requested order could not be located. In cases where an order exists, the system verifies the customer's existence and proceeds to validate the token's legitimacy.

If the token is deemed invalid, the function initiates a database transaction to delete all data associated with the customer, including their measurements and customer record. To ensure the deletion process completes without constraints, foreign key checks are temporarily disabled using database statements. Once the data is removed, foreign key checks are re-enabled, and a log entry is created to record the deletion event for auditing purposes. In case of errors during this process, the function captures the exception, logs the error, and aborts the request with a 404 response.

Additionally, the function handles scenarios where the link is expired for paid orders. If the order is marked as paid and the link is expired, the function begins another transaction to delete all related customer data, following the same process of disabling and re-enabling foreign key checks. A detailed log is recorded to document the data deletion resulting from the expired link. If any issues arise, the function rolls back the transaction, logs the error, and aborts the process with an appropriate error message.

This function ensures a secure and efficient workflow for managing customer order details while enforcing data protection policies. By invalidating unauthorized or expired access links and securely removing associated data, the system maintains both privacy and compliance. The use of logging mechanisms enhances traceability and accountability, making the function a critical component of the system's customer data management process.

4.2 Data Retention Module

The code provided in Fig 10 implements a query in a Laravel-based application to retrieve "Retention Orders" by filtering data according to specific conditions. It begins by defining a $title variable with the value "Retention Orders," which is passed to the view for display purposes. The query focuses on fetching the latest "paid" orders for customers while excluding those with active orders. To achieve this, it joins the orders table with the customers table using the customer_id column, establishing a relationship between orders and their respective customers.

```php
public function retention()
{
    $title = "Retention Orders";

    // Get latest paid orders for each customer
    $orders = Orders::join('customers', 'orders.customer_id', '=', 'customers.id')
        ->whereExists(function($q) {
            // Has orders in history
            $q->select('customer_id')
                ->from('order_histories')
                ->whereColumn('orders.customer_id', 'order_histories.customer_id');
        })
        ->whereNotExists(function($q) {
            // Does NOT have any active orders
            $q->select('customer_id')
                ->from('orders as o')
                ->whereColumn('orders.customer_id', 'o.customer_id')
                ->whereNull('o.deleted_at')
                ->where('o.status', '!=', 'paid');
        })
        ->where('orders.link_status', 'active')
        ->whereNull('orders.deleted_at')
        ->where('orders.status', 'paid')
        ->whereIn('orders.id', function($query) {
            // Subquery to get only the latest order ID for each customer
            $query->select(DB::raw('MAX(id)'))
                ->from('orders')
                ->where('status', 'paid')
                ->whereNull('deleted_at')
                ->groupBy('customer_id');
        })
        ->select('orders.*')
        ->get();

    return view('retention', compact('title', 'orders'));
}
```

Fig. 10 *Data Retention*

Fig 10 shows the query uses the whereExists method to ensure only orders associated with customers who have prior transaction history in the order_histories table are included. Simultaneously, the whereNotExists method excludes customers with active orders, defined as orders with no deletion date (deleted_at) and a status of "paid." Additional filters are applied to include only orders with an active link_status, exclude soft-deleted orders, and ensure the status is "paid." A subquery within the whereIn clause selects the latest order ID for each customer by retrieving the maximum id value (DB::raw("MAX(id)")) and grouping by customer_id. This guarantees that only the most recent paid order for each customer is considered.

Finally, the query retrieves all columns from the orders table and executes the query using the get() method. The results, along with the title, are passed to the retention view for rendering. This implementation is tailored to generate a refined list of retention orders by applying robust filtering and subquery logic, ensuring that the data aligns with data retention setting.

4.2.1 Cleanup Expired Data

The code in Fig 11 implements a cleanup function in a Laravel application to handle expired data efficiently by removing outdated orders and their associated records. The process begins by fetching configuration values for the retention period (retention_period) and the time unit (retention_unit), which determine the expiration duration. The Carbon::now() function is used to calculate the expiration date by subtracting the configured period from the current time. A switch statement handles different time units (minutes, hours, days) to dynamically adjust the expiration logic.

```
public function cleanupExpiredData()
{
    $period = config('retention.period', 400);
    $unit = config('retention.unit', 'days');

    $expiryDate = Carbon::now();

    switch($unit) {
        case 'minutes':
            $expiryDate = $expiryDate->subMinutes($period);
            break;
        case 'hours':
            $expiryDate = $expiryDate->subHours($period);
            break;
        case 'days':
            $expiryDate = $expiryDate->subDays($period);
            break;
    }

    // Get expired orders
    $expiredOrders = Orders::with('customer')
                        ->whereNotNull('paid_at')
                        ->where('paid_at', '<=', $expiryDate)
                        ->where('link_status', '!=', 'revoked')  // Only process non-
                        ->get();

    foreach($expiredOrders as $order) {
        try {
            DB::beginTransaction();

            // Store customer name in order_histories before deletion
            if ($order->customer) {
                // Get all order histories for this customer and update them
                DB::table('order_histories')
                    ->where('customer_id', $order->customer_id)
                    ->update(['customer_name' => $order->customer->fullname]);
            }

            // First update the order status
            $order->update([
                'access_token' => null,
                'link_status' => 'revoked'
            ]);

            // Use the same deletion logic as the delete button
            try {
                DB::statement('SET FOREIGN_KEY_CHECKS=0;'); // Temporarily disable foreign key checks

                // Delete from body_measurements first
                BodyMeasurement::where('customer_id', $order->customer_id)->delete();

                // Then delete from customers
                Customer::where('id', $order->customer_id)->delete();

                DB::statement('SET FOREIGN_KEY_CHECKS=1;'); // Re-enable foreign key checks
            } catch (\Exception $e) {
                Log::error('Error during customer deletion: ' . $e->getMessage());
                throw $e;
            }

            DB::commit();
            Log::info('Successfully processed order #' . $order->id);
        } catch (\Exception $e) {
            DB::rollBack();
            Log::error('Error in cleanup for order #' . $order->id . ': ' . $e->getMessage());
        }
    }

    return response()->json(['message' => 'Cleanup completed']);
}
```

Fig. 11 *Cleanup Expired Data*

Fig 11 shows the system retrieves expired orders by querying the orders table for records where the paid_at timestamp is older than the calculated expiration date and the link_status is not marked as "revoked." This query ensures that only non-revoked orders meeting the expiration criteria are selected. The orders are fetched with their associated customer data for further processing.

The cleanup process iterates over the expired orders using a foreach loop. Within each iteration, a database transaction is initiated to ensure atomicity. If the order is associated with a customer, the customer's order histories are first moved to the order_histories table for record-keeping, preserving important data before deletion. Following this, the order's attributes, such as access_token, are reset, and the deletion process is carried out. Temporary foreign key checks are disabled during the deletion process to ensure referential integrity and avoid constraints-related errors.

The code handles errors during the cleanup process by wrapping operations in a try-catch block. If an error occurs, it logs the exception details and continues with the next iteration, ensuring that other orders are processed even if one fails. After all expired orders are handled, the function returns a JSON response confirming the cleanup's completion. This implementation demonstrates a robust approach to managing expired data, ensuring data integrity, preventing orphaned records, and maintaining application performance.

4.3 Role-Based Access Control

The code provided in Fig 12 and Fig 13 implements role-based access control (RBAC) in a Laravel-based application by conditionally rendering elements of the user interface based on the user's role and permissions. This ensures that only authorized users can access or interact with specific functionalities.

```
@if(auth()->user()->staff && auth()->user()->staff->role !== 'staff')
    <li class="{{ Request::routeIs('users') ? 'active' : '' }}">
        <a href="{{route('users')}}">
            <i class="material-icons">people_outline</i>
            <span class="menu-title" data-i18n="Users">Staff Management</span>
        </a>
    </li>
@endif
```

Fig. 12 *RBAC for Staff Management*

The code in Fig 12 checks if the currently authenticated user has a staff relationship and if their role is not equal to staff. If these conditions are met, the system dynamically generates a navigation menu item for "Staff Management." The logic uses Laravel's auth() helper to retrieve the currently logged-in user and evaluates their role. If the conditions are satisfied, an HTML list item (<li>) is rendered with a link to the "Staff Management" section of the application. The link includes dynamic attributes for highlighting the menu item as active when the user navigates to the corresponding route.

```
@if(auth()->user()->staff->role === 'admin')
    <button class="btn btn-primary btn-sm" data-toggle="modal" data-target="#retention-settings">
        <i class="la la-cog"></i> Retention Settings
    </button>
@endif
```

Fig. 13 *RBAC for Retention Setting*

The code in Fig 13 performs a stricter check for users with the admin role. If the authenticated user's role is explicitly set to admin, the system renders a button labeled "Retention Settings." This button triggers a modal window for managing data retention settings. By using conditional logic, the button ensures that only users with administrative privileges can access and manage sensitive application settings. The button's attributes and design follow standard conventions for user interaction, making it both functional and visually consistent with the rest of the interface. These snippets demonstrate the implementation of RBAC principles by dynamically controlling access to features and UI components based on user roles. This approach enhances security by ensuring that unauthorized users cannot access restricted areas or perform privileged actions. Furthermore, it improves user experience by presenting only relevant options to users, depending on their roles, reducing clutter and potential confusion. This implementation aligns with best practices in secure and efficient application design, making the system both robust and user-friendly.

4.4 Testing Result

The testing is crucial to ensure that the system operates as intended, providing seamless and reliable experience for all user roles, including admins, staff, and customers. Each test case was designed to evaluate specific functionalities and security features, ensuring that the system meets its defined objectives. Functionality testing focuses on verifying that the system's core operations, such as login processes, order management, and data handling, work as expected without errors. Security testing emphasizes the system's ability to protect sensitive information, enforce access controls, and handle potential vulnerabilities effectively. Together, these tests ensure the system's performance, reliability, and compliance with security standards. The detailed results are summarized in Table 3 for functionality testing and Table 4 for security testing.

Table 3 *Functionality Testing*

No	Test case	Expected Result	Actual Result
1.	Admin login with correct credentials	Admin dashboard is accessible	Pass
2.	Staff login with correct credentials	Staff dashboard is accessible	Pass
3.	Customer accesses unique order link	Order details page is displayed	Pass
4.	Admin creates a new customer profile with measurements	Customer profile is created and saved	Pass
5	Staff updates customer order status to "Paid"	Status updated successfully	Pass
6.	Admin sets a data retention period	Retention period is saved and applied	Pass
7.	Admin and Staff views order history	Order history is displayed accurately	Pass
8.	Customers view their order status via the unique link after it is marked "To Collect"	Order status displayed correctly	Pass
9.	Staff changes the password during the first login	Password updated successfully	Pass
10.	Staff logs out after completing tasks	Staff are logged out, and the session ends securely	Pass

Table 3 shows the functionality testing verifies the core operations of the system for all user roles, including admin, staff, and customers. It confirms that the admin can successfully log in and access the dashboard, create new customer profiles with measurements, set data retention periods, and view order histories. Staff members can also log in securely, manage customer orders, update order statuses, change their passwords during first login, and log out after completing tasks. Additionally, customers can view their order details via a unique order link, ensuring accessibility and user-friendliness. All test cases passed, confirming the system's effectiveness in delivering its intended functionalities.

Table 4 *Security Testing*

No	Test case	Expected Result	Actual Result
1.	Admin or staff attempts login with incorrect credentials	Error message is displayed, login fails	Pass
2.	Customer accesses order link after it expires	Access is denied	Pass
3.	Attempt to access admin functionalities without proper role	Access is denied	Pass
4.	Input less than 8 characters for a new password	Validation errors are displayed, password update fails	Pass
5	Admin updates data retention settings	Retention policies are applied securely without errors	Pass
6.	Ensure expired customer data is securely deleted after the retention period	Data is removed from the database without leaving residual information	Pass

Table 4 shows the security testing evaluates the system's ability to handle unauthorized access and ensure data protection. It verifies that login attempts with incorrect credentials are denied, admin functionalities are inaccessible to unauthorized roles, and expired order links cannot be accessed by customers. Password validation is enforced, requiring a minimum of eight characters, and retention policies are securely updated by the admin. The testing also confirms that customer data is securely deleted from the database after the retention period, leaving no residual information. All test cases passed, demonstrating the system's robust security measures and compliance with data protection standards.

During testing, a few notable challenges came up that affected how the system performed. One of the main issues happened after deployment, when the database was unexpectedly lost due to a server configuration problem. Thankfully, a recent backup was available, which helped restore the system quickly without major data loss. Another issue involved the data retention feature. The system didn't automatically block access to the customer's order link once the retention period ended. Instead, the order page including the invoice and thank-you message remained visible until the page was manually refreshed. To fix this, the Laravel command `php artisan schedule:work` was used to run background tasks every minute, allowing the system to delete expired data automatically, even without a page reload.

4.4.1 User Acceptance Form

The feedback gathered through the User Acceptance Form to evaluate the usability and efficiency of the Tailoring Order Management System. The evaluation focused on two key aspects: the user interface (UI) and user experience (UX). Feedback was collected from 15 participants, and the results are presented in the charts below in Fig 14 and Fig 15. The responses provide detailed insights into how effectively the system supports navigation, organization, and simplicity in performing key tasks, reflecting user satisfaction with the system design and functionality.

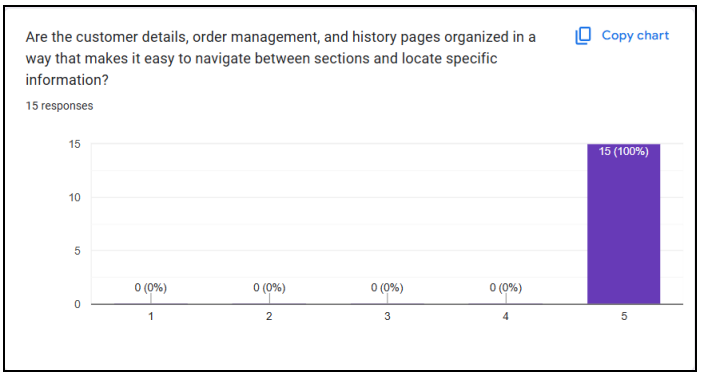


Fig. 14 *User Interface (UI) form*

Fig 14 shows that all 15 participants (100%) rated the organization and navigation of the system's interface with a perfect score of 5. This feedback reflects that the customer details, order management, and history pages are well-structured and designed for easy access. Users found it effortless to navigate between different sections and locate specific information without confusion. The high rating suggests that the interface layout, menu structures, and visual design elements contribute to an intuitive and seamless user experience. This level of satisfaction implies that users can complete their tasks efficiently and effectively, without unnecessary delays or frustration. The feedback validates that the system's user interface aligns with the needs and expectations of its target users.

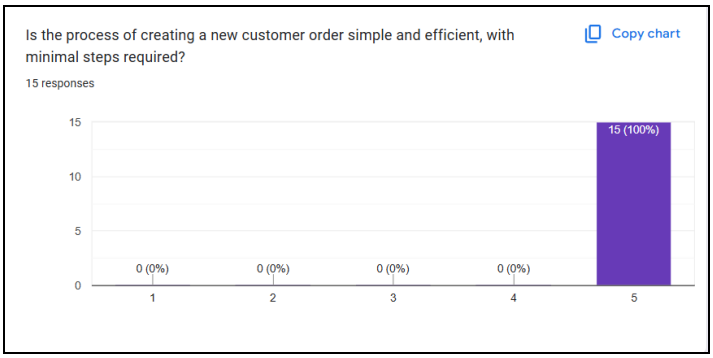


Fig. 15 *User Experience (UX) form*

Fig 15 shows that similarly, all 15 participants (100%) rated the user experience (UX) of creating a new customer order with a perfect score of 5. This result highlights the simplicity and efficiency of the order creation process. Participants appreciated that the system requires minimal steps to complete a task, making it quick and hassle-free to generate new orders. The streamlined workflow reduces the time and effort needed to perform critical functions, ensuring a smooth experience for staff handling customer orders. The perfect score also suggests that users did not encounter any unnecessary complexity or technical barriers, further validating the effectiveness of the system's design. This feedback underscores the system's ability to meet the functional needs of its users while enhancing productivity and ease of use.

5. Conclusion

The Tailoring Order Management System for Nadimah Tailor successfully achieved its objectives by providing efficient order management, secure role-based access control, and data retention features to optimize operations and protect customer data. Despite challenges such as database management issues and retention feature limitations, alternative solutions ensured the system's functionality. This project demonstrates the practical value of adopting structured digital solutions in tailoring businesses, enhancing both operational effectiveness and customer satisfaction [14]. To further improve the system, future updates such as OTP authentication for staff logins and automated customer notifications are recommended. OTP can serve as a second layer of verification, particularly useful in situations where passwords are weak, shared, or forgotten. Additionally, introducing automated customer notifications would further enhance engagement and improve the overall service experience. These upgrades would not only strengthen system security and usability but also align the platform with modern expectations, encouraging continued innovation in the tailoring services domain [15].

Acknowledgement

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

Conflict of Interest

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

Author Contribution

The authors confirm contribution to the paper as follows: **study conception and design:** *K. N. H Ku Azman, N. Rahim*; **data collection:** *K. N. H Ku Azman, N. Rahim*; **analysis and interpretation of results:** *K. N. H Ku Azman, N. Rahim*; **draft manuscript preparation:** *K. N. H Ku Azman, N. Rahim*. All authors reviewed the results and approved the final version of the manuscript.

References

[1] A. V. Omopariola, R. O. Enihe, C. N. Ogbonna, U. Felix, M. C. Ezeocha, and M. J. Abdullahi. (2023)."Design and Implementation of a Tailoring Management System (Virlor)," *Open Journal for Information Technology*, vol. 6, no. 2, pp. 67.

[2] Academic Project Guidance Provider. *Online tailoring management system*. Academic Project Guidance Provider. Retrieved from https://www.graciousnaija.com/2021/11/online-tailoring-management-system.html

[3] N. Chilate and A. Bhandarkar, "Order management systems in e-commerce: Enhancing customer satisfaction," *Journal of Retail Management*, vol. 8, no. 4, pp. 45–52, 2022.

[4] CyberEdge Group, "Effective data retention strategies in modern enterprises," *Journal of Cybersecurity*, vol. 12, no. 1, pp. 56–72, 2024.

[5] O'Hagan Meyer, "Best practices for data retention and disposal," *Data Management Insights*, vol. 9, no. 3, pp. 35–42, 2024.

[6] Levi Strauss & Co., "Data retention for business continuity: Lessons from healthcare," *Enterprise Systems Management*, vol. 14, no. 2, pp. 23–37, 2024.

[7] Commonwealth of Virginia, "Best practices for data retention in public institutions," *Data Security Framework*, 2024.

[8] Cybereason, "Protecting data through effective retention policies," *Cybersecurity for Enterprises*, 2021.

[9] O'Hagan Meyer, "Data retention and legal compliance," *Journal of Information Security*, vol. 10, no. 2, pp. 55–60, 2024.

[10] A. Rahim, "Role-based access control in enterprise systems: Challenges and strategies," *Journal of Information Systems and Security*, vol. 18, no. 1, pp. 23–36, 2023.

[11] C. Ri, J. Kim, and S. Park, "Blockchain-enhanced RBAC for cloud environments," *Journal of Blockchain Technology*, vol. 11, no. 2, pp. 89–97, 2022.

[12] S. Chatterjee, A. S. Kumar, and M. Y. Lim, "Dynamic role-based access control in decentralized applications," in *Proceedings of the International Conference on Cybersecurity*, 2020, pp. 129–135.

[13] J. Shi, R. K. Gupta, and W. Yang, "Flexible and scalable role-based access control systems," *International Journal of Cybersecurity*, vol. 17, no. 4, pp. 67–76, 2023.

[14] E. Oye, A. Ibrahim, and C. Osueke, "Role-Based Access Control (RBAC) Enhancements for Big Data," *ResearchGate*, 2023. [Online]. Available: https://www.researchgate.net/publication/387707684_ROLE-BASED_ACCESS_CONTROL_RBAC_ENHANCEMENTS_FOR_BIG_DATA

[15] National Institute of Standards and Technology (NIST), "The Economic Impact of Role-Based Access Control (RBAC)," NIST Planning Report 02-1, 2002. [Online]. Available: https://www.nist.gov/system/files/documents/2017/05/09/report02-1.pdf