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

AITCS

Vol. 6 No. 1 (2025) 1065-1084
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Navinsilk Tailoring Management System

Nanthini Rajagobal[1], Noraini Ibrahim[1]*

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

*Corresponding Author: noraini@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.01.059

Article Info

Received: 7 January 2025
Accepted: 19 June 2025
Available online: 30 June 2025

Keywords

Web-based System, Operational Efficiency, PHP, JavaScript, Customer Satisfaction, Tailoring Management, Prototyping Model.

Abstract

NavinSilk Tailoring Shop, a custom tailoring service, has built a reputation for quality craftsmanship but now faces significant operational challenges due to its reliance on manual processes. To address issues such as long customer wait times, inaccurate order tracking, and inefficiencies in workflow management, a web-based management system was developed using the Prototyping Model. This iterative approach enabled continuous user feedback and refinement. The system comprises seven modules: login and sign-up, manage profile, manage stitching type, manage workers, generate report, manage order, and manage appointment and view order status. Developed using an object-oriented approach, the web application leverages PHP, JavaScript, HTML, and CSS to streamline operations. The testing results show that 100% of the features function properly as expected. In conclusion, the NavinSilk Tailoring Management System has successfully digitized order management, customer interactions, and workflow tracking, significantly enhancing operational efficiency, accuracy and customer satisfaction while establishing a new benchmark for excellence in the tailoring industry.

1. Introduction

In the ever-evolving landscape of the tailoring industry, efficiency and customer satisfaction are paramount. NavinSilk Tailoring Shop, established in 2000, has carved out a reputable position through quality service and craftsmanship. However, as the business has grown, the reliance on traditional, manual methods for order management, customer interactions, and workflow tracking has become increasingly problematic. The advent of digital solutions presents an opportunity to overcome these inefficiencies and streamline operations. This paper introduces the NavinSilk Tailoring Management System, a web-based solution designed to enhance operational efficiency, accuracy, and overall customer satisfaction by digitizing the shop's management processes. Automating these operations not only improves record-keeping and process efficiency but also demonstrates how digital solutions can transform traditional tailoring businesses [1].

The primary motivation for this study stems from the challenges faced by NavinSilk in its current operations. Manual order entries, physical tracking of garments, and paper-based sales recording have led to inefficiencies, errors, and delays, adversely affecting both customer satisfaction and business growth. The hypotheses suggest that a web-based management system will significantly reduce errors, improve tracking efficiency, and enhance customer satisfaction. The objectives of the project include analyzing and designing the NavinSilk Tailoring Management System using an object-oriented approach, developing a web-based application for the system, and testing it with stakeholders to ensure it meets operational requirements and enhances customer satisfaction. The scope of this paper encompasses the development and implementation of the NavinSilk Tailoring Management System, including modules for login and sign up, manage profile, manage stitching type, manage worker, generate

report, manage order, and manage appointment and view order status Expected outcomes include streamlined operations, improved accuracy, and enhanced customer experience, setting a new standard in the tailoring industry.

The system is used by four main user groups. There are admin, tailor, staff, and customers. The admin, who is the owner of the shop, is responsible for overall management and administrative tasks. The tailor handles the measurement details for customers' garments, ensuring accurate and custom fits. The staff members work at the shop, assisting with order processing, customer service, and day-to-day operations. They can view details of what needs to be sewn each day, allocate orders by managing bill numbers and shelf locations, and track the sewing progress. The customers are those who utilize NavinSilk's tailoring services, booking appointments, and tracking their order status through the system. By addressing the specific needs of each user group, the NavinSilk Tailoring Management System aims to enhance the efficiency and quality of service provided.

The remainder of this paper is organized as follows: Section 2 provides a literature review on digital solutions in traditional industries. Section 3 discusses the methodology used for system development. Section 4 presents the results and discussion. Finally, Section 5 concludes the paper.

2. Related Work

This section primarily explains technology behind the NavinSilk Tailoring Management System, its decision-making capabilities, and how it compares to the existing system.

2.1 Web based Tailoring Management System

Web-based tailoring management systems facilitate the digital transformation of traditional tailoring shops by automating operations and enhancing efficiency and customer satisfaction. These systems eliminate geographical barriers, enabling customers to access services from anywhere with an internet connection. Typically, they follow a Model-View-Controller (MVC) architecture, ensuring scalability, security, and maintainability by separating the system into three layers: the presentation layer (HTML, CSS, JavaScript), the application layer (PHP, Python, Node.js), and the data layer (MySQL, PostgreSQL) [2]. This modular approach allows for changes in one layer with minimal impact on others. Frontend development utilizes responsive frameworks like React or Angular, while backend development manages business logic and data processing securely. Centralized data storage simplifies administration, enhances security, and allows for efficient data backup and restoration [3]. Regular software updates can be quickly deployed, mitigating security risks associated with outdated software.

2.2 Database in Web-based System

Databases are fundamental to web-based systems, providing a structured method for storing, managing, and retrieving data efficiently [4]. A database management system (DBMS) ensures data integrity, security, and consistency, supporting dynamic content generation, user authentication, data storage, and transaction management. Various types of databases are utilized in web-based systems, including relational databases like MySQL and PostgreSQL, which use SQL for data manipulation and are known for their robustness and ACID compliance [5]. NoSQL databases, such as MongoDB and Cassandra, offer flexible, scalable solutions for unstructured data and high-performance needs, while NewSQL databases like Google Spanner combine NoSQL scalability with traditional ACID guarantees. For the NavinSilk Tailoring Management System, MySQL is chosen for its high performance, reliability, scalability, and robust security features. It integrates well with web technologies like PHP, making it suitable for handling large data volumes and concurrent transactions efficiently [6]. MySQL also supports a wide range of operating systems, ensuring flexibility and ease of deployment in various environments. Furthermore, its strong community support and extensive documentation make troubleshooting and development more manageable. Despite some complexity in horizontal scaling and limitations compared to NoSQL databases, MySQL's benefits in performance, reliability, and cost-effectiveness make it an ideal choice for this project.

2.3 Programming languages in Web-based System

Programming languages are essential in web development, enabling the creation of dynamic, interactive, and functional web applications. HTML and CSS are used for structuring and styling web pages, providing the foundation for a responsive and visually appealing interface. JavaScript enhances user experience by adding interactivity and enabling dynamic content updates without reloading the entire page. PHP, chosen for its strong integration with databases, ease of use, and powerful backend processing capabilities, handles tasks such as form submissions, user authentication, and data processing. In the development of the NavinSilk Tailoring Management System, these languages are selected for their complementary strengths and widespread use. PHP can be utilized in both framework example Laravel, Symfony and non-framework approaches, offering flexibility and efficiency in development. Frameworks provide a structured development process, adhering to the Model-View-Controller

(MVC) architecture, while custom PHP coding allows for tailored solutions to specific needs.. This combination ensures the creation of a robust, scalable, and user-friendly web application, suitable for managing the comprehensive requirements of a modern tailoring shop.

2.4 Related works on measurement details in tailoring

The interface shown in Fig 1 was utilize as a reference for developing the measurement input functionality in the proposed tailoring management system. This interface, sourced from Mutembei's Online Tailoring Management system, provides a comprehensive framework for recording customer orders and measurements [7]. It includes fields for selecting the type of dress, fabric, material, and quantity alongside detailed measurement fields such as full length, shoulders, chest/bust, sleeve length, and waist measurements. Due to limitations in accessing measurement form designs in the other systems reviewed, this interface served as a valuable guideline for structuring the order-taking and measurement input module. Its systematic layout and attention to detail ensure accurate and efficient data entry, contributing to enhanced customer satisfaction and operational efficiency.

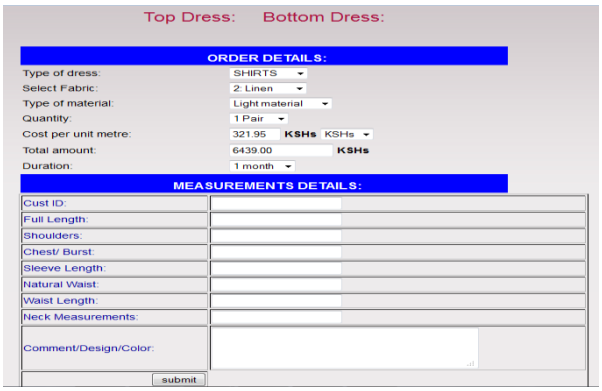

Fig.1 *Mutembei's Online Tailoring Management System*

2.5 Study of existing system

In this section, the introduction and the background study of three existing related systems is discussed. The three existing related systems are SmartMaster, Tech Tailor and Tailor Store. The functionalities and features of three related systems are identified. The comparison of these existing systems with the proposed system is figured out.

Table 1 *Comparisons of Features between Proposed System and Existing System*

Features	SmartMaster[8]	Tech Tailor [9	Tailor Store [10]	NAVINSILK Tailoring Management System
Sign Up and Login	Use username or email address or password	Use email and password		Use username and password
Manage Profile	User able to edit their profiles			Able to edit personal details
Manage Stitching Type	Cannot be accessed			Admin can add, edit and delete the stitching type and price
Manage Worker	Cannot be accessed			Admin can add and deactivate workers account
Generate Report	Cannot be accessed			Generate detailed monthly sales reports
Manage Order	Cannot be accessed	Record garment details	Cannot be accessed	Record stitching measurements details
Order allocation	Cannot be accessed			Enter shelf number and locate garments

Table 1 *Continue*

Upcoming sewing reminder system		Cannot be accessed		Reminders for upcoming sewing tasks
Track Order		Cannot be accessed		Track sewing progress with status
Manage appointment and view order status	Select date and time	View order status	Cannot be accessed	Book appointment and view order status

Based on the comparative analysis in Table 1, the existing system and the proposed NavinSilk Tailoring Management System to be efficient and user-friendly. NavinSilk was developed as a web-based system, enhancing accessibility and ease of use. This system integrates the most crucial features from SmartMaster, Tech Tailor and Tailor Store, ensuring a comprehensive solution tailored to meet the needs of a tailoring business. While some existing systems lack certain features, the NavinSilk Tailoring Management System address these gaps, providing a complete and efficient tailoring management solution.

3. Methodology

This section is mainly discussing the System Development Life Cycle (SDLC) model chosen to guide the development of the NavinSilk Tailoring Management System.

3.1 Prototyping Model

A Prototyping Model is chosen for this project because it allows for iterative development and continuous feedback from end users, which is crucial for refining the system to meet the specific needs of NavinSilk Tailoring Shop. This model supports the creation of an initial prototype to test design possibilities, illustrate concepts, and gain insights into user requirements and potential solutions. By engaging users early and throughout the development process, the Prototyping Model helps ensure that the final product is user-friendly and functionally robust. Table 2 depicts the software development actives associated with the task during the software development.

Table 2 *Software Development Activities and their tasks*

Phase	Task	Output
Planning	☐ Define the background of the case study ☐ Define the problem statement ☐ Prepare the proposal ☐ Gathering information about the proposal from Google Scholar	☐ Project proposal ☐ Gantt chart
Iteration 1 : Prototype (Interface)		
Analysis	☐ Conduct interviews ☐ Analyze the requirements ☐ Create UML diagrams	☐ Use Case Diagram, Activity Diagram, Class Diagram, Requirement Definition
Design	☐ Design the user interfaces ☐ Design the system architecture ☐ Design the data dictionary	☐ Architecture Diagram, interfaces, Data Dictionary ☐ Prototype 1 (with interface)
Implementation	☐ Implement Prototype 1	☐ Prototype 1
Iteration 2 : Prototype (Database)		
Analysis	☐ Validate the requirement	☐ Prototype with database
Design	☐ Design database	☐ Schema table
Implementation	☐ Implementation	☐ Prototype 2

Table 2 *Continue*

Implementation system	☐ Implement system ☐ Test the system	☐ Test case ☐ Requirement traceability matrix ☐ Completed system

3.2 Analysis

The list of requirement for each use case are listed in Appendix A.

3.3 Use Case and Class Diagram

There are four actors in NavinSilk. They are admin, staff, tailor, and customers, each engaging with the system through distinct use cases. This developed system covers seven modules which are sign up and login, manage profiles, manage stitching type, manage worker, generate report, manage order and manage appointment and view order status. The use case diagram illustrates these interactions, offering a clear visualization of how each user group engages with the system's various functions. This is depicted in Fig 2, highlighting the dynamic relationship between the system and its users.

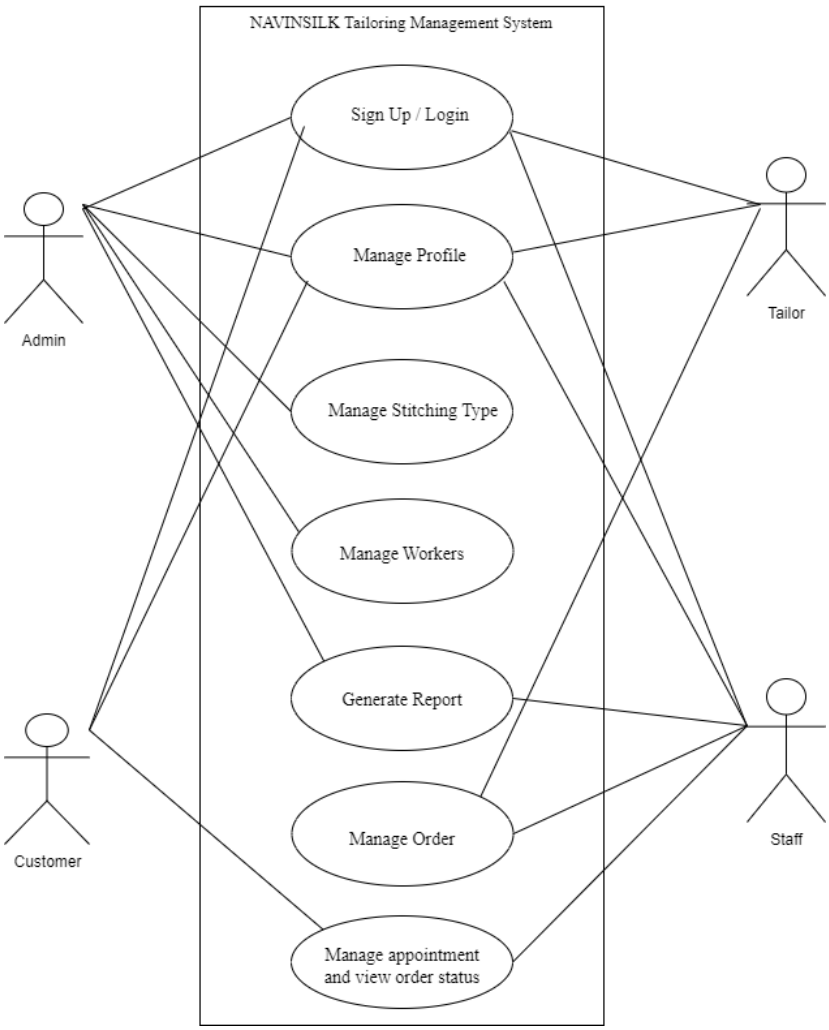

Fig 2 *Use Case Diagram for NAVINSILK Tailoring Management System*

Next, the relationship among the classes for the NavinSilk Tailoring Management System is shown in Fig 3. In general users, service, orders, receipt, order services, measurements, measurement details, stitching photos, garment photos, design, custom designs, service remarks, order stitching type and appointment. Each class has its own attributes and relationship.

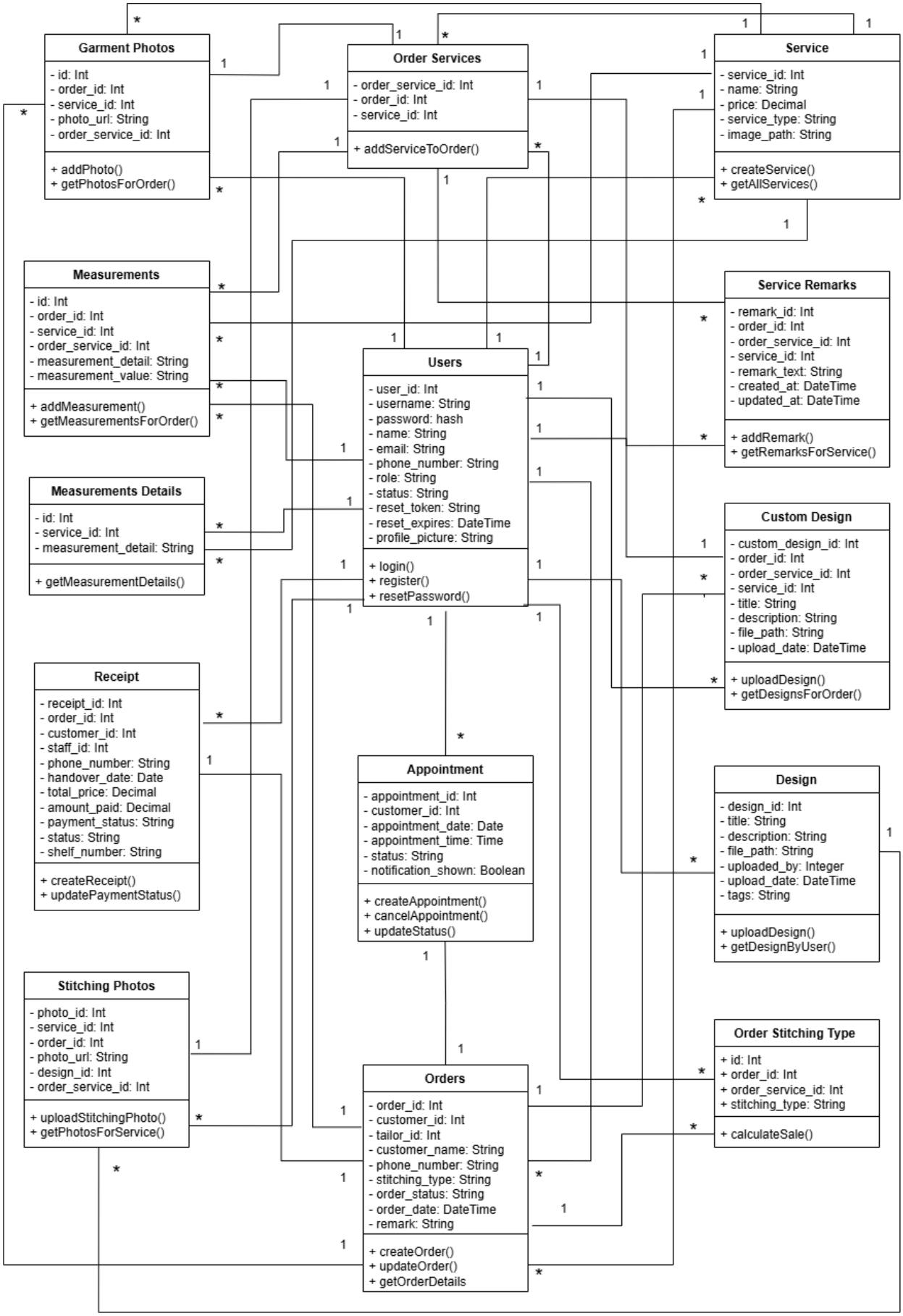

Fig. 3 *Class Diagram for NavinSilk Tailoring Management System*

3.4 Database Design

The database design is presented by the schema table as follows:

i. users(`user_id(PK), username(UNIQUE), password, name, email, phone_number, role, status, reset_token, reset_expires`).
ii. orders (`order_id(PK), customer_id(FK), tailor_id(FK), customer_name, phone_number, stitching_type, order_status, order_date`).
iii. order_services(`order_service_id(PK), order_id(FK), service_id(FK)`)
iv. order_stitching_types (`id(PK, order_id(FK), order_service_id(FK), service_id(FK)`).
v. service(`service_id(PK), name, price, service_type, image_path`).
vi. service_remarks(`remark_id(PK), order_id(FK), order_service_id(FK), service(FK), remark_text, created_at, updated_at`).
vii. stitching_photos(`photo_id(PK), service_id(FK), order_id(FK), order_service_id(FK), photo_url, design_id`).
viii. measurement (`id(PK), order_id(FK), service_id(PK), measurement_detail, measurement_value`).
ix. measurements_details (`id(PK), service_id,measurement_detail`).
x. garment_photos(`id(PK), order_id,service_id(FK), photo_url, order_service_id(FK))`.
xi. custom_designs(`custom_design_id(PK), order_id, order_service_id(FK), service_id, title, description, file_path, upload_date`).
xii. appointment(`appointment_id(PK), customer_id(FK), appointment_date, appointment_time, status,notification_shown`).
xiii. designs(`design_id(PK), title, description, file_path, uploaded_by, upload_date`).
xiv. receipt (`receipt_id(PK), order_id(FK), customer_id(FK), staff_id(FK), phone_number, handover_date, total_price, amount_paid, payment_status, status, shelf_number`).

4. Result and Dicussion

This section is separated into two parts, the implementation part that shows all webpage created and code segment used in the system while testing part that summarizes the test results of the system.

4.1 Implementation

In this section, the webpage created and code segment of each use cases in the system is illustrated and explained. The first interface is the Register Page shown in Fig 4(a), which is used by customers to create an account by providing their username, name, phone number, email, password, and confirming their password, along with completing a CAPTCHA verification. After the customer clicks the "Register" button, the system validates the form inputs, example checking for empty fields and valid email formats as shown in the Fig 4(b). The addEventListener method attaches a validation function to the form submission event to ensure all required fields are filled correctly before the data is sent to the backend. The Fig 4(c) is the Login Page shown , designed for admins, staff, tailors, and customers to securely log in using their username and password. After the user clicks the "Login Now" button,

the system validates the provided inputs ensuring the fields are not empty, as shown in the Fig 4(d). The validation function uses the addEventListener method to ensure the login process begins only when the username and password are entered.

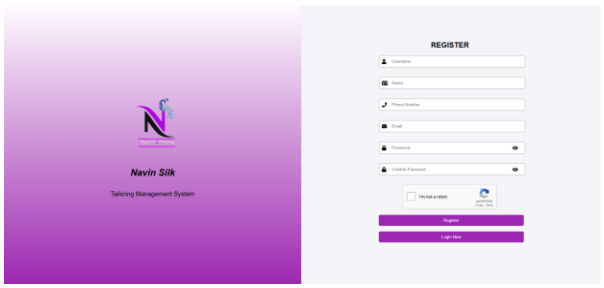

Fig 4(a) *Customer Register Page*

Fig 4(b) *Customer Register Code*

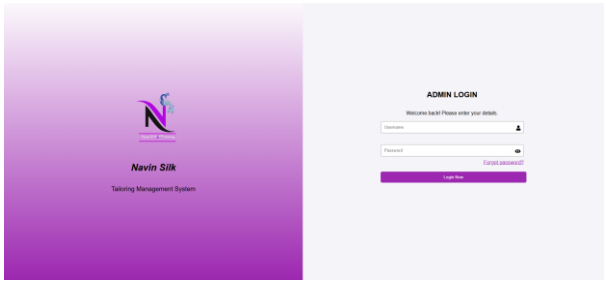

Fig 4(c) *Login Interface*

```
loginForm.addEventListener('submit', function(event) {
    let formIsValid = true;

    if (!username.value) {
        usernameError.textContent = 'Username required';
        formIsValid = false;
    } else {
        usernameError.textContent = '';
    }

    if (!password.value) {
        passwordError.textContent = 'Password required';
        formIsValid = false;
    } else {
        passwordError.textContent = '';
    }

    if (!formIsValid) {
        event.preventDefault();
    }
});
```

Fig 4(d) *Login Code*

The interface for the "Forgot Password" functionality is designed to allow users to securely initiate a password reset process. Users are prompted to enter their registered email address as shown in the Fig 5(a), which features a heading, an input field for the email, and a "Send Reset Link" button. Upon submission, the system processes the input and, if valid, sends a password reset link to the provided email, ensuring ease of use and accessibility. The code for this functionality shown in the Fig 5(b) validates the provided email and ensures that it is associated with a registered user of the specified role (example staff, tailor, admin, or customer). Once validated, a secure token is generated, and an expiration time of 1 hour is set. The token and its expiration are then stored in the database to associate it with the user. A password reset link containing the token is constructed and sent to the user's email using the PHPMailer library. This ensures that the user can securely reset their password via the provided link.

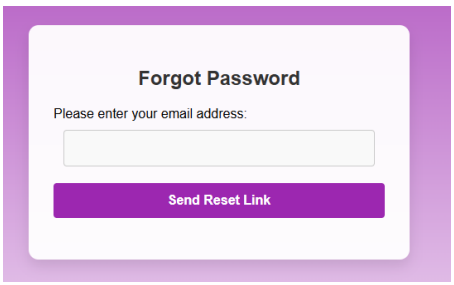

Fig 5(a) Forgot Password *Page*

```
$stmt = $conn->prepare("SELECT * FROM users WHERE email = ? AND role = ?");
$stmt->bind_param("ss", $email, $role);
$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows > 0) {
    // Email is registered, generate a reset token
    $user = $result->fetch_assoc();
    $token = bin2hex(random_bytes(50));
    $expires = date("Y-m-d H:i:s", time() + 3600); // Token expires in 1 hour
```

Fig 5(b) *Forgot Password Code*

Figure 6(a) shows the Manage Profile interface, allowing the admin to manage their personal information. Users can update their name, email, phone number, profile picture, and password. The profile picture section provides a preview of the selected image before submission. Additionally, password fields have a visibility toggle for user convenience. The interface is designed for simplicity, with options to update the profile or return to the dashboard. The Fig 6(b) handles the submission of the Manage Profile form when a user updates their profile information. It begins by retrieving the data submitted through the form using the $_POST array for text inputs like name, email, phone number, and passwords, and the $_FILES array for the uploaded profile picture. The extracted data includes the user's updated name, email, phone number, new password (if provided), and the uploaded profile picture. For the profile picture, the $upload_dir variable specifies the directory (uploads/) where the uploaded image will be

stored, and the $profile_picture_url is initially set to the existing profile picture, ensuring that the existing picture remains unchanged if a new one is not uploaded. This section sets the groundwork for processing updates, including handling file uploads and performing necessary validations.

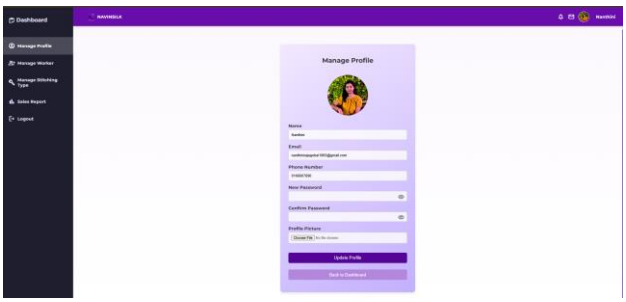

Fig 6(a) *Customer Register Page*

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $name = $_POST['name'];
    $email = $_POST['email'];
    $phone_number = $_POST['phone_number'];
    $profile_picture = $_FILES['profile_picture'];
    $password = $_POST['password'];
    $confirm_password = $_POST['confirm_password'];

    // Handle file upload
    $upload_dir = 'uploads/';
    $profile_picture_url = $profilePicture;
```

Fig 6(b) *Customer Register Code*

Fig 7(a) is the take measurement details interface for recording customer measurements. It displays customer details such as ID, name, phone number, and tailor details at the top for quick reference. Tailors can select different stitching types and input specific measurements like waist size, armhole, and bust, along with uploading garment photos for each stitching type. The interface allows adding multiple stitching types and measurements, with options to save all the data or delete specific entries for better flexibility and accuracy. Tailors can add multiple stitching types using the "Add More Stitching Type" button. Each stitching type includes fields for measurements and garment photo uploads. The addStitchingType() function dynamically generates a new stitching group when invoked, as shown in Fig 7(b).

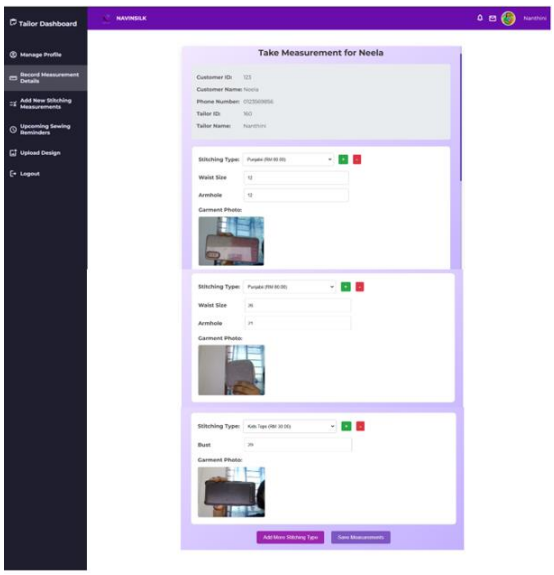

Fig 7(a) *Take Measurement Form*

```javascript
function addStitchingType() {
    const container = document.getElementById('stitching-container');
    const index = Date.now();
    const template = `
        <div class="stitching-group">
            <div class="input-group">
                <label>Stitching Type:</label>
                <select class="stitching-type" name="stitching_entries[${index}][stitching_type]" onchange="loadMeas
                    <option value="">Select Stitching Type</option>
                    <?php
                    // Rewind to start of $stitching_types, if needed
                    $stitching_types->data_seek(0);
                    while ($row = $stitching_types->fetch_assoc()):
                        ?>
                        <option value="<?= htmlspecialchars($row['service_id']) ?>">
                            <?= htmlspecialchars($row['name']) ?> (RM <?= htmlspecialchars($row['price']) ?>)
                        </option>
                    <?php endwhile; ?>
                </select>
                <button type="button" class="duplicate-button" onclick="duplicateStitchingType(this)">+</button>
                <button type="button" class="remove-button" onclick="removeStitchingType(this)">-</button>
            </div>
            <div class="measurement-fields"></div>
            <div class="take-photo-container"></div>
        </div>
    `;
    container.insertAdjacentHTML('beforeend', template);
}
```

Fig 7(b) *Take Measurment Form Code*

Fig 8(a) is the design selection interface for tailoring Order #674, where customer and order details, such as customer ID and name, are shown at the top for easy reference. Each stitching service section displays the related measurements, garment photos, and options to choose a design either from a preloaded gallery or by uploading a custom design. If the customer wants to provide their own design, they can scan the QR code to upload their desired design, which will then appear in the "Chosen Design" section for that service. Tailors can also add remarks for customization, and the interface includes a "Save & View Measurements" button to save the design choices and proceed with the tailoring process. After the tailor selects the "Custom Design" option, the system displays a QR code that allows the customer to upload their design. Once uploaded, the system dynamically fetches and displays the design in the "Chosen Design" section, as shown in the Fig 8(b). The fetchCustomDesigns function polls the server for new uploads every 5 seconds and updates the interface with the latest custom design.

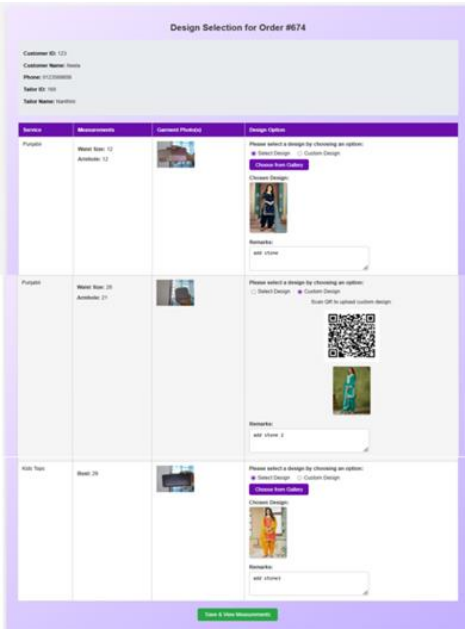

Fig 8(a) *Design Selection for Orders*

```
function fetchCustomDesigns(orderServiceId) {
    const orderId = <?= json_encode($order_id) ?>;
    const url = `design_selection.php?ajax=1&order_id=${orderId}&order_service_id=${orderServiceId}`;
    fetch(url)
    .then(r => r.json())
    .then(data => {
        const container = document.getElementById('custom-uploads-' + orderServiceId);
        if (!container) return;
        if (!Array.isArray(data) || data.length === 0) {
            container.innerHTML = `<p><em>No custom design uploaded yet.</em></p>`;
        } else {
            // Only display the latest custom design
            const latest = data[0];
            let safePath = latest.file_path ? latest.file_path.replace(/</g,'&lt;') : '';
            container.innerHTML = `
                <div style="margin-top:10px;text-align:center;">
                    <img src="${safePath}" alt="Custom Design"
                         style="max-width:100px;border:1px solid #ccc;border-radius:5px;">
                </div>
            `;
        }
    })
    .catch(err => console.error("Error in poll:", err));
}
```

Fig 8(b) *Design Selection for Orders*

Fig 9(a) is the Receipt for Order #674, summarizing the details of the customer's order. It displays the customer's name, phone number, and the tailor's name at the top, followed by a breakdown of stitching types with their quantities and prices. The measurements section lists the services along with specific measurement details and values. At the bottom, it includes the handover date, payment option (example: half payment), the amount paid, balance due, and the total price. A "Generate Receipt" button is provided to create a final receipt for the order. The payment option dropdown allows users to choose between "Full," "Half," or "Other" payments. Based on the selected option for "Full," the amount paid is equal to the total price, for "Half," the amount paid is calculated as 50% of the total price and for "Other," the user can manually input a custom payment amount like show in the Fig 9(b).

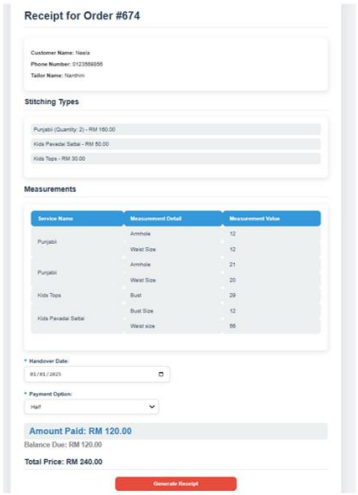

Fig 9(a) *Generate Receipt Interface*

```
function updatePrice() {
    const paymentOption = document.querySelector('select[name="payment_option"]');
    const customPaymentContainer = document.getElementById('custom_payment_contai
    const customPaymentInput = document.getElementById('custom_payment');
    const totalPrice = <?= json_encode($total_price) ?>;
    let amount = 0;
    const customPaymentError = document.getElementById('custom_payment_error');
    const generateButton = document.querySelector('button[type="submit"]');

    // Reset error message and button state
    customPaymentError.style.display = 'none';
    generateButton.disabled = false;
```

Fig 9(b) *Generate Receipt Code*

The Fig 10(a) shown the Upcoming Sewing Reminders feature is shown in both the staff and tailor to ensure timely updates for ongoing orders. In the staff page, reminders include details such as the receipt number, types of stitching, assigned tailor, handover date, and order status (example "On Time"). The tailor page includes similar details but with additional functionalities, such as a View Measurement Details button to access a detailed breakdown of measurements, garment photos, and design specifics. A modal window is available for tailors, displaying detailed measurements, which helps in verifying and completing the order accurately. The code shown in the Fig 10(b) calculates the stitching date (three days before the handover date) for each order and compares it with the current date to determine the order status. If the current date is on or before the stitching date, the status is displayed as "On Time"; otherwise, it is marked as "Late". This ensures that staff and tailor can view accurate status updates for timely follow-ups on orders.

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 1065-1084

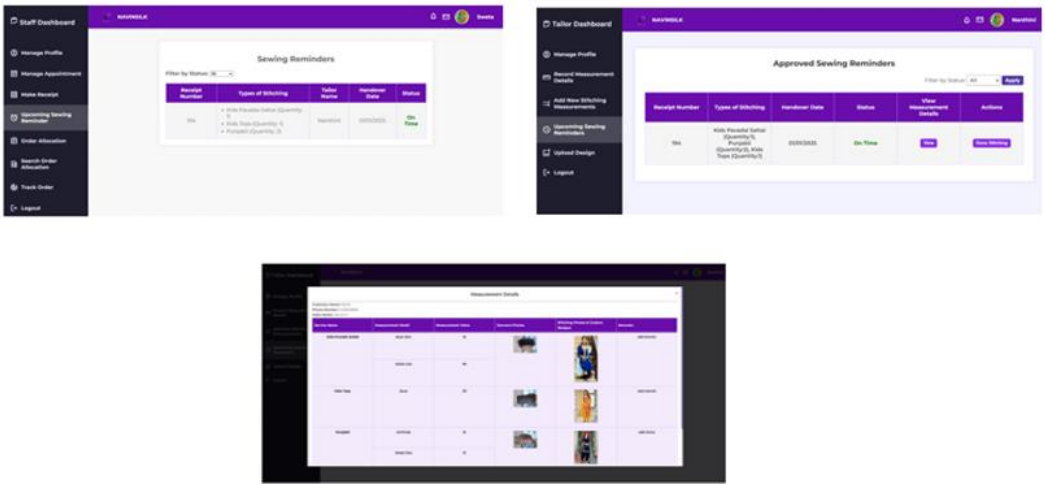

Fig 10(a) *Upcoming reminders interface*

```php
if (mysqli_num_rows($result) > 0) {
    while ($row = mysqli_fetch_assoc($result)) {
        $handover_date = $row['handover_date'];
        $formatted_date = date('d/m/Y', strtotime($handover_date));

        // Calculate stitching_date = handover_date - 3 days
        $stitching_date = date('Y-m-d', strtotime('-3 days', strtotime($handover_date)));

        // Determine the status display based on current_date and stitching_date
        if ($current_date <= $stitching_date) {
            // Current date is on or before stitching_date
            $status_display = "<span class='status-on-time'>On Time</span>";
        } else {
            // Current date is after stitching_date
            $status_display = "<span class='status-late'>Late</span>";
        }
```

Fig10(b) *Upcoming reminder code*

Fig 11(a) shown the Track Order section in the Staff page provides a centralized interface to monitor the progress of tailoring orders. It allows staff to search for specific orders using filters such as receipt number, month, date, assigned tailor, or status. The table displays key details like receipt number, customer name , tailor name, types of stitching services, and the order's current status (example, completed, pending and handover to customer). This feature enhances order management by enabling staff to quickly locate and assess the status of any order, ensuring smooth operations and timely updates. The searchByReceipt() function shown in Fig 11(b) retrieves the receipt number entered by the user. If no receipt number is provided, it alerts the user with a message prompting them to enter one and halts further execution. This ensures that the search functionality only proceeds when a valid input is entered.

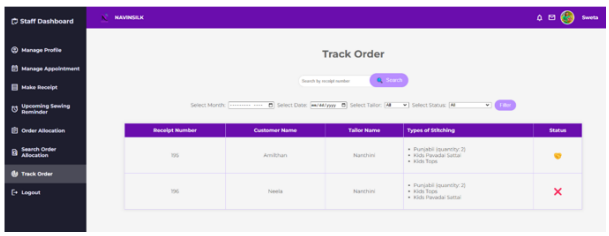

Fig11(a) *Track order status interface*

```javascript
function searchByReceipt() {
    const receiptNumber = document.getElementById("receiptNumber").value.trim();
    if (!receiptNumber) {
        alert("Please enter a receipt number.");
        return;
    }
```

Fig 11(b) *Track order status code*

4.2 Testing

The testing phase starts once the system prototype is completed. Two types of testing are required: system testing and acceptance testing. System testing evaluates whether the developed system is functional and usable. Acceptance testing is conducted by gathering feedback from users through questionnaires that focus on the system's features. This process ensures that the system operates as planned and aligns with the users' goals and requirements. The overall results of the testing are depicted in Table 3 and summarized in a pie chart as shown in Figure 10. To conclude, all of the function in the test case modules are functioning as intended, where there are no failed test cases for each module. Therefore, the outcome of each test case modules is 100%, which gives the overall result of 100%. The details of the test cases is shown in the Appendix B.

Table 3: *Overall Result Test Cases*

Test Case Modules	Number of Test Cases	Total Passed Test Cases	Total Failed Test Cases
TC_100 Sign Up & Login	4	4	0
TC_200 Manage Profile	5	5	0
TC_300 Manage Stitching Type	5	5	0
TC_400 Manage Workers	4	4	0
TC_500 Generate Report	2	2	0
TC_600 Manage Orders	4	4	0
TC_700 Manage Appointment and view order status	2	2	0

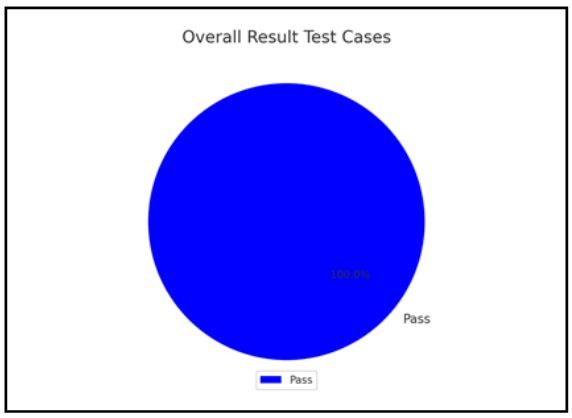

Figure 10 *Pie chart for overall test cases result*

4.3 Ucer Acceptance Testing

Total of 10 respondents were to indicate their acceptance level for the statements made on five point rating scale. The questionnaire statements and the evaluation from respondent for each statement are presented in Table 8. In conclusion, the majority of respondent rate the system's functionality and user interface as 5. This demonstrates that the system is more likely to be accepted by users. The results of the interface and module evaluation, based on 10 respondents, are shown as averages in the bar graph in Figure 11. The details of the user interface evaluation and the system module evaluation are provided in Appendix C.

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 1065-1084

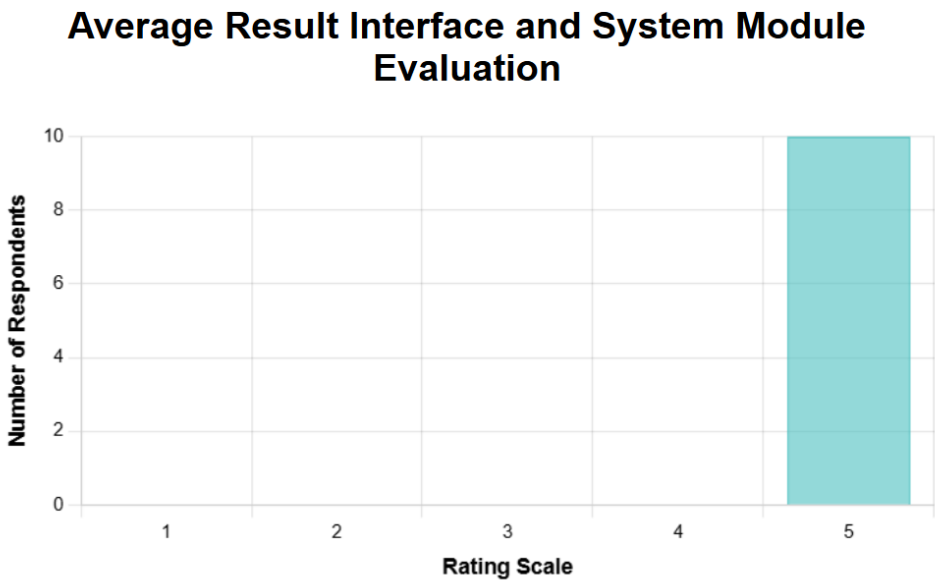

Figure 11 *The Average Result Interface and System Module Evaluation*

5. Conclusion

NavinSilk Tailoring Management System is a user-friendly web-based system that enables customers to book appointments efficiently and track their order status in real-time. It is not only beneficial for customers but also for the staff, tailors, and admin of NavinSilk. By having this kind of system, the staff can manage orders more effectively, ensuring accurate order processing, while tailors can keep precise measurements and track progress. Additionally, the admin can oversee operations and make informed decisions based on real-time data. However, as with any system, there are limitations. Currently, the system focuses on streamlining the appointment booking and order tracking processes. It does not yet include features such as automated inventory management. Future enhancements may include these additional features to better match evolving business needs and customer preferences.

Acknowledgement

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

References

[1] Yuliana, D. C. T., & Sulandjari, S. (2024). Adoption of digital technology by embroidery entrepreneurs in Salatiga City (Case study on small business Convection Persada, Tingkir). *Indonesian Interdisciplinary Journal of Sharia Economics (IIJSE)*, *8*(1), 800–820.

[2] Rustagi, P., & Kumar, Y. (2022). MVC architecture and its application.

[3] Buoye, P., & Akinbola, S. (2024). Safeguarding data integrity: A comprehensive exploration of database backup and recovery using tamper-resistant properties of blockchain. *Federal Polytechnic Ilaro Journal of Pure and Applied Sciences*, *6*(1).

[4] Zhao, Q., Tang, R., Peng, M., & Guo, M. (2024). A web-based approach for the efficient management of massive multi-source 3D models. *Journal of Geodesy & Geoinformation Science*, *7*(3).

[5] Lindgren, S. (2024). Performance comparison between two relational database management systems: B-tree indexing in PostgreSQL and MySQL.

[6] Vatjalainen, A. (2023). SQL versus NoSQL: Comparison case MySQL versus MongoDB.

[7] Mutembei, K. D. (2012). Online tailoring management system. *Mount Kenya University Repository*. https://erepository.mku.ac.ke/server/api/core/bitstreams/dc2a59d8-494e-49f4-adae-cc4dc8ccbf8b/content

[8] Smaster Suit Sdn Bhd (SmartMaster). (2020). Smart Master Malaysia premium menswear brand. *Smart Master*. https://smartmaster.com.my/

[9] Tailor Store Sweden AB. (2024). *Tailor Store*. https://www.tailorstore.com/my/en/

[10] Online Tailoring Services | Tailoring Service at Home | Tech-tailor| tech-tailor.com. (2021). https://www.tech-tailor.com/

Appendix A:

Table A.1 *List of Requirement*

Software Requirement Specification	Description
SRS_REQ_100	**Sign Up & Login module**
SRS_REQ_101	The system shall allow the user to log in with a valid username and password.
SRS_REQ_102	The system shall allow the user to sign up using their details, including a valid username, password, and confirmation of the password.
SRS_REQ_103	The system shall validate the username during the sign-up process.
SRS_REQ_104	The system shall redirect the user to the home page upon a successful login.
SRS_REQ_105	The system shall show a message confirming a successful login or sign-up.
SRS_REQ_106	The system shall show an error message if a text field is left blank or if there is an invalid login attempt.
SRS_REQ_107	The system shall manage exceptions, such as invalid credentials or already used email addresses.
SRS_REQ_108	The system shall update the database with the new password if the user forgets their password.
SRS_REQ_200	**Manage Profile**
SRS_REQ_201	The system shall allow the user to access profile management options from the navigation bar page.
SRS_REQ_202	The system shall show personal details and log out options on the navigation bar page.
SRS_REQ_203	The system shall allow the user to update personal details such as name, phone number, and email address.
SRS_REQ_204	The system shall verify and save the updated personal details in the database.
SRS_REQ_205	The system shall display a confirmation message after successfully updating personal details.
SRS_REQ_206	The system shall enable the user to change their password from the navigation bar page.
SRS_REQ_207	The system shall manage exceptions, such as invalid credentials or already used email addresses.
SRS_REQ_208	The system shall update the database with the new password if the user forgets their password.
SRS_REQ_209	The system shall show a confirmation message after the password has been successfully changed.
SRS_REQ_210	The system shall allow the user to log out from the navigation bar page.
SRS_REQ_211	The system shall display a pop-up message requesting log out confirmation.
SRS_REQ_212	The system shall log out the user and redirect them to the login page upon confirmation

SRS_REQ_213	The system shall be capable of handling exceptions.
SRS_REQ_300	**Manage Stitching Type**
SRS_REQ_301	The system shall allow the admin to access stitching type management options from the admin dashboard.
SRS_REQ_302	The system shall display options to add, update, delete the stitching type.
SRS_REQ_303	The system shall allow the admin to add a new stitching type by entering details.
SRS_REQ_304	The system shall validate and store the new stitching type details in the database.
SRS_REQ_305	The system shall display a confirmation message after successfully adding a new stitching type.
SRS_REQ_306	The system shall allow the admin to update an existing stitching type.
SRS_REQ_307	The system shall validate and store the updated stitching type details in the database.
SRS_REQ_308	The system shall display a confirmation message after successfully updating a stitching type.
SRS_REQ_309	The system shall allow the admin to delete a stitching type.
SRS_REQ_310	The system shall ask for confirmation before deleting a stitching type.
SRS_REQ_311	The system shall delete the stitching type from the database upon confirmation
SRS_REQ_312	The system shall display a confirmation message after successfully deleting a stitching type.
SRS_REQ_313	The system shall display alert messages for blank text fields during stitching type enter.
SRS_REQ_314	The system shall display alert messages for duplicate stitching type names during stitching type management.
SRS_REQ_400	**Manage Workers**
SRS_REQ_401	The system shall display the worker management page to the admin.
SRS_REQ_402	The system shall display the list of all workers.
SRS_REQ_403	The system shall allow the admin to view detailed information of any works
SRS_REQ_404	The system shall provide an option for the admin to generate new accounts for worker.
SRS_REQ_405	The system shall provide an option for the admin to deactivate worker.
SRS_REQ_406	The system shall display a success message after a worker is deactivate.
SRS_REQ_407	The system shall update the status of deactivate worker in the database.
SRS_REQ_500	**Generate Report**

SRS_REQ_501	The system shall display the report page.
SRS_REQ_502	The system shall allow the admin to generate reports.
SRS_REQ_503	The system shall allow the admin to export.
SRS_REQ_600	**Manage Order**
SRS_REQ_601	The system shall allow the tailor to record stitching measurement details.
SRS_REQ_602	The system shall allow the staff to view the amount of fabric that needs to be sewn.
SRS_REQ_603	The system shall allow the staff to enter the bill number and shelf location.
SRS_REQ_604	The system shall display the shelf location of the fabric based on the bill number.
SRS_REQ_605	The system shall send reminders to the staff three days before the handover date.
SRS_REQ_606	The system shall display the sewing status of fabrics using color-coded buttons.
SRS_REQ_607	The system shall update the order status in the database.
SRS_REQ_608	The system shall display confirmation messages for successful operations.
SRS_REQ_700	**Manage Appointment &View Order Status**
SRS_REQ_701,	The system shall allow customers to book appointments for stitching garments by selecting a date.
SRS_REQ_702	The system shall validate the selected appointment date.
SRS_REQ_703	The system shall record appointment details in the database.
SRS_REQ_704	The system shall display a confirmation message after successfully booking an appointment.
SRS_REQ_705	The system shall allow customers to view the status of their orders.

Appendix B:

Table A2 *Test Cases*

Test Cases	Software Requirement	Description	Status
TEST_100	SRS_REQ_100	**Sign Up & Login module**	Pass/Fail
TEST_100_001	SRS_REQ_101, SRS_REQ_104, SRS_REQ_105	Valid Login The user logs in with a valid username and password, the system shows a success message and redirects to the home page.	Pass
TEST_100_02	SRS_REQ_102, SRS_REQ_103, SRS_REQ_105	Sign Up (Valid Details) The users signs up with valid details, including a unique username and matching password/confirm password, the system confirms successful sign-up	Pass
TEST_100_03	SRS_REQ_106, SRS_REQ_107	Invalid Login/Blanks Fields The user tries to log in with either blank fields or incorrect credentials, the system displays an error message and handles the exception	Pass
TEST_100_04	SRS_REQ_108	Forgot Password The user resets the password with a registered email, the system updates the DB with the new password and confirms the update	Pass
TEST_200	SRS_REQ_200	**Manage Profile**	Pass/Fail
TEST_200_01	SRS_REQ_201, SRS_REQ_202	Access Profile Management The user navigates from the navigation bar, sees personal details and confirms that profile management	Pass
TEST_200_02	SRS_REQ_203, SRS_REQ_204, SRS_REQ_205	Update Personal Details The user updates name, phone and email. The system validates inputs, save them, and displays a confirmation message	Pass
TEST_200_03	SRS_REQ_203, SRS_REQ_207, SRS_REQ_208, SRS_REQ_209	Change Password The user enters a new password and confirm password. The system validates, updates the DB, and shows a success message.	Pass
TEST_200_04	SRS_REQ_210, SRS_REQ_211, SRS_REQ_212	Log out The user choose to log out from the navigation bar and redirect to the login page upon confirmation	Pass
TEST_200_05	SRS_REQ_207, SRS_REQ_208, SRS_REQ_213	Profile Exceptions Test scenarios where the user enters invalid data while updating details.	Pass
TEST_300	SRS_REQ_300	**Manage Stitching Type**	Pass
TEST_300_01	SRS_REQ_301, SRS_REQ_302	Access Stitching Type Management The admin navigates to the admin dashboard and see options manage stitching type	Pass
TEST_300_02	SRS_REQ_303, SRS_REQ_304, SRS_REQ_305	Add New Stitching Type The admins add a new stitching type; the system validates, store in the DB, and shows a success message.	Pass
TEST_300_03	SRS_REQ_306, SRS_REQ_307, SRS_REQ_308	Update Stitching Type The admin updates an existing stitching type, the system validates, store changes and confirms the update	Pass
TEST_300_04	SRS_REQ_309, SRS_REQ_310, SRS_REQ_311, SRS_REQ_312	Delete Stitching Type The admin deletes an existing stitching type, the systems ask for confirmation, deletes from DB, and shows a confirmation message	Pass

TEST_300_05	SRS_REQ_313, SRS_REQ_314	Validate & Alerts The admin attempts to add or update with blank fields or a duplicate stitching type name, the system displays appropriate error message	Pass
TEST_400	SRS_REQ_400	**Manage Workers**	Pass/ Fail
TEST_400_01	SRS_REQ_401, SRS_REQ_402	Display Worker Management The admin open the worker management page and sees a list of all workers	Pass
TEST_400_02	SRS_REQ_403	View Worker Details The view workers full detail	Pass
TEST_400_03	SRS_REQ_404	Generate New Worker Account The admin creates a new account for a worker, the system save it and confirms creation	Pass
TEST_400_04	SRS_REQ_405, SRS_REQ_406, SRS_REQ_407	Deactivate Worker The admin deactivates a worker, the system updates the DB, and display a success message.	Pass
TEST_500	SRS_REQ_500	**Generate Report**	Pass/ Fail
TEST_500_01	SRS_REQ_501 SRS_REQ_502	The admin navigates to the report page and generates a report	Pass
TEST_500_02	SRS_REQ_503	Export Report After generating a report, the admin exports it to PDF	Pass
TEST_600	SRS_REQ_600	**Manage Order**	Pass/ Fail
TEST_600_01	SRS_REQ_601, SRS_REQ_602	Record &View Fabric Needs The tailor records measurements and staff can view the required fabric amount	Pass
TEST_600_02	SRS_REQ_603, SRS_REQ_604	Bill Number & Shelf Location Staff enters a bill number and shelf locations, the system displays the correct shelf location when a bill number is provided	Pass
TEST_600_03	SRS_REQ_605, SRS_REQ_606	Reminders & Color Status The systems send a reminder three days before handover date and displays the sewing status with color-coded buttons	Pass
TEST_600_04	SRS_REQ_607, SRS_REQ_608	Update Order Status & Confirm The system updates the order status in the DB and shows confirmation message when operations are successful	Pass
TEST_700	SRS_REQ_700	**Manage Appointment &View Order Status**	Pass/ Fail
TEST_700_01	SRS_REQ_701, SRS_REQ_702, SRS_REQ_703, SRS_REQ_704	Book Appointment The customer selects a date for stitching. The system validates and records appointment details, then shows a confirmation message	Pass
TEST_700_02	SRS_REQ_705	View Order Status The customer checks the status of their orders (Example: “In progress”, “Completed”)	Pass

Appendix C:

Table A3 *User Interface Evaluation*

No	Statement	Scale					Total
		1	2	3	4	5	
1	Does the system help you book appointments and view order statuses more quickly?	0	0	0	0	10	10
2	Does the system make it easier to track your orders and garment details?	0	0	0	0	10	10
3	Does the system simplify the overall ordering and tracking process?	0	0	0	0	10	10
4	Does using the system improve your experience with managing appointments and garment orders?	0	0	0	0	10	10
5	Overall, do you find the system useful as a customer?	0	0	0	0	10	10

Table A4 *System Module Evaluation*

No	Statement	Scale					Total
		1	2	3	4	5	
1	Is it easy for you to log in and sign up for the system?	0	0	0	0	10	10
2	Can you navigate the system easily to track your garment orders?	0	0	0	0	10	10
3	Is it easy for you to become skilled at using the system for appointments and orders?	0	0	0	0	10	10
4	Do you find the system's layout and instructions clear and easy to follow?	0	0	0	0	10	10
5	Is the interface intuitive for viewing your order status?	0	0	0	0	10	10