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

AITCS

Vol. 6 No. 2 (2025) 1624-1643
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Moxsha Equipment Rental and Decoration Services Management System

Harini Anandarao[1], Azizul Azhar Ramli[1]*

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

*Corresponding Author: azizulr@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.02.087

Article Info

Received: 12 June 2025
Accepted: 3 November 2025
Available online: 30 November 2025

Keywords

Equipment Rental, Decoration Service Management, Mobile Application, Web-based System, Agile Methodology, Flutter, Laravel

Abstract

Moxsha Equipment Rental and Decoration Services Management System was developed to overcome the limitations of manual booking and administrative processes used by Moxsha Event, which often resulted in double bookings, scheduling issues, and miscommunication. The primary objective was to design, develop, and test a mobile application for customers and a web-based system for administrators that streamlines rental and decoration services through real-time availability tracking and secure payment features. The project employed the Agile methodology to enable iterative development and stakeholder engagement. Development tools included Flutter, Android Studio, Visual Studio, and a MySQL database. Testing confirmed improved booking accuracy, reduced operational delays, and enhanced customer experience. Key advantages include online payments with automated receipts, effective admin management tools, and simplified return handling. Future work may involve iOS support, integrated chat, advanced customization, delivery tracking, and analytics for decision-making. These improvements position the system as a robust solution for modernizing rental and decoration service operations.

1. Introduction

Event management is a specialized area in project management that involves planning, coordination, and execution of personal or corporate events within the event and hospitality industry [1]. These events can range from small gatherings to large functions like weddings, festivals, and conferences. Success in event management requires various skills, including strategic planning, risk analysis, marketing, budgeting, cash flow management, and human resource handling [2]. This demands clear task delegation, strong communication, efficient information handling, customer satisfaction, and real-time updates to reduce operational challenges.

Moxsha Event is a leading event management company that provides comprehensive services, decoration packages and equipment rentals. The company aims to deliver exceptional customer experiences by handling all aspects of event planning, from equipment rental to decor setup. To rent equipment, customers must check availability by visiting in person or contacting staff via chat, where availability is verified manually using documents, a logbook, and a calendar. For items such as furniture, dessert ware, and decorative pieces, customers must pay a refundable deposit. Moxsha Event offers personalized decoration packages, where customers can choose themes, specify the number of guests, and select custom options like basic lighting that match the theme.

At present, Moxsha Event faces several challenges due to its manual system. Issues such as double bookings, scheduling conflicts, and miscommunication with customers often occur. It is also difficult to track availability and rental items accurately. The lack of a proper system to manage bookings, order details, and equipment has led to

overbooking, last-minute cancellations, and delayed responses to customers, reducing customer satisfaction. Paper-based processes are time-consuming and take up space, making it hard for staff to find and manage documents. Errors in tracking orders and customer details also cause cash flow problems and delay payments, which affects the company's operations and financial management.

Hence, a management system for equipment rentals and decoration services will be developed to handle administrative tasks and improve service delivery. This system will streamline the booking process, enhance customer communication, and organize operations efficiently. A real-time availability feature will be included for rental services, allowing customers to check open dates and make instant bookings. The system will also let customers explore, select, book, and customize decoration packages based on their event preferences. Additionally, it will include a secure record-keeping system to manage booking details, customer information, financial transactions, and services provided. This will reduce paperwork and lower the risk of errors, data loss, and scheduling conflicts. The event management system will be designed, developed, and undergo user acceptance testing to ensure it meets user requirements.

2. Related Work

This section explores related work in online booking and rental management systems, examining key advancements, technologies, and comparisons with existing systems.

2.1 Online Booking and Rental Management

Online reservation systems play a key role in the hospitality industry by expanding customer reach and offering a convenient, cost-effective way to book service [3]. These systems are now widely used in the equipment rental sector to streamline operations and enhance user experience. For instance, Yunita's work on an Android-based outdoor equipment rental information system highlights the transition from manual data processing to a more efficient, database-driven application that allows for online service registration and management of rental data [4]. This shift simplifies the rental process and improves data accessibility and management, which is crucial for businesses aiming to enhance customer satisfaction.

The development of mobile applications for booking services has been a game-changer in the industry. Adducul's mobile bus ticketing system shows how user-friendly apps allow customers to book and pay directly from their smartphones [5]. This trend is mirrored in various sectors, where the convenience of mobile booking is increasingly becoming a standard expectation among consumers.

In response to this, the Moxsha System was developed to improve service delivery efficiency. It introduces a mobile-based platform with features such as availability tracking, online payment integration, automated documentation, and receipt generation. By digitizing bookings and payments, the system creates an organized record-keeping process and improves communication between customers and administrator staff. These functionalities aim to enhance the user experience while ensuring operational accuracy and productivity for the company. Implementing real-time tracking, safe online payment options and automated records resolves the major problems encountered by equipment rental services today [6], [7].

2.2 Method and Technology

Moxsha System uses mobile technologies to create a user-focused platform. The system is designed to improve real-time booking management and provide flexibility to meet user needs. Mobile-based platforms have become integral to business strategies by allowing customers to shop at any time from any places [8]. This approach changes how Moxsha interacts with customers and manages daily operations A key benefit of mobile platforms is that they enable instant communication between businesses and users [9].

The system will be developed using Android Studio with the Flutter Software Development Kit (SDK) to make use of Android technology features. Android Studio supports fast app development with tools that help in creating, testing, and optimizing features like booking management and payment gateways [10]. It also provides a layered software stack that includes native libraries and a framework for running applications [11]. The framework offers system services through APIs, which simplify communication between the mobile app and backend processes. Flutter adds value with pre-built widgets, a fast development cycle, and customizable user interfaces [12]. Adopting Android and Flutter ensures the Moxsha System is scalable, user-friendly, and adaptable to future needs.

2.3 Comparison with the Existing Systems

Party Nutty, Just Rent It! Malaysia and Areum Events are the selected system for comparative study. Each system was examined for its functions as it serves benchmarks for identifying the key requirements and innovative solutions for the proposed Moxsha System. A detailed comparison between the existing systems and the proposed system is presented in **Table 1**.

Table 1 *System's Comparison*

System Features	Party Nutty	Just Rent It! Malaysia	AREUM Events and Wedding	Moxsha System
Operating System	Web-based	Web-based	Web-based	Mobile-based application
Development Platform	WordPress	WordPress	WordPress	Android Studio and Visual Code
Programming Language	Unknown	Unknown	Unknown	Android Studio and Visual Code
Database	MySQL	MySQL	MySQL	MySQL
Login	X	X	Username/ email address and password	Email and password
Account Registration	X	X	√	√
Password Reset	X	X	√	√
Booking Cart	√	√	√	√
Availability Updates	√	√	√	√
Online Payment	X	X	√	√
Receipt Generation	X	X	X	√
Wishlist	X	X	X	√

This table provides a clear comparison to help identify the strengths and limitations of each system. It includes their purpose, operating system, development platform, and programming language. Key features like login, registration, booking cart, and wish list are also highlighted.

3. Methodology

The Agile model is an incremental approach that emphasizes flexibility, collaboration, and customer feedback [13], making it ideal for the Moxsha System development. By breaking the project into more manageable units, agile focuses on delivering functional components of the system in time. As the project progresses, tasks can be prioritized and adapted to changes in requirements during each iteration, guided by business value [14].

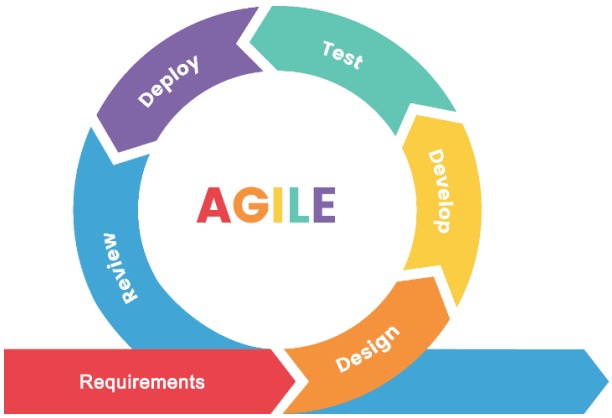


Fig. 1 *Agile Methodology [15]*

Figure 1 illustrates the six phases involved in the Agile methodology, highlighting the key stages of the process that includes requirements, design, development, testing, deployment and review. Each task and output of theses phases is presented in Table 2 below.

Table 2 *Software Development Activities and Tasks*

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1624-1643

Phase	Task	Output
Requirement	Meet with stakeholders, gather and confirm system needs, and create the initial backlog.	Requirements document, project timeline (Gantt chart), and system backlog.
Design	Create wireframes, system diagrams (UML, ERD, flowcharts), and review with stakeholders.	System design documents, diagrams, database schema, and UI mockups.
Development	Build the system using Flutter, Android Studio, and MySQL, and document the process.	Working application, connected database, and technical documentation.
Testing	Test system features, fix bugs, and collect user feedback through UAT.	Test reports, fixed issues, and UAT feedback.
Deployment	Launch the system, monitor its performance, collect feedback, and update documentation.	Live system, user feedback, and updated deployment documents.
Review	Present the system, get stakeholder input, assess outcomes, and improve the system.	Stakeholder feedback, updated backlog, and revised documentation.

Table 2 summarise the tasks for gathering requirements, designing the system, developing the code, testing functionality, deploying the system, and reviewing the final work. Each phase leads to specific outputs such as project documentation, system architecture, functional code, testing results, and refined system features based on stakeholder feedback.

3.1 System Requirements

By detailing the functional requirements, the system establishes a framework that enhances user interactions and optimizes processes within it. Non-functional requirements provide insights into the system's operational capabilities and constraints, ensuring that the software design and architecture align with the system qualities [16]. Table 3 and Table 4 displays the functional and non-functional requirements for Moxsha System. It is vital to ensure that the system addresses the needs, preferences, and limitations of its users by implementing a user-centered design approach [17]. This user requirements are shown in **Table 5.**

Table 3 *Functional Requirements*

No.	Modules	Task
1.	Registration and Login Module	- Allow the new customers to register a new account before logging in. - Allow the existing customers to log in with the email address and password. - Redirect the valid customers to the dashboard when a successful log in occurs. - Allow the existing customers to reset their passwords. - Allow the administrator to log in with email and password.
2.	Rental Module	- Allow the customers to view the rental item's details. - Allow the customers to specify rental duration and add item. - Allow the customers to add selected rental items to a shopping cart. - Allow the customers to cancel rental reservations before payment. - Allow the customers to checkout their selections. - Allow the administrator to create new rental items. - Allow the administrator to update details of existing rental items. - Allow the administrator to delete rental items.
3.	Decoration package module	- Allow the customers to view decoration package details. - Allow the customers to select decoration packages and dates. - Allow the customers to add a venue for their event.

Table 3: (cont.)

No.	Modules	Task
3.	Decoration package module	- Allow the customers to add selected decoration packages to a shopping cart. - Allow customers to cancel decoration bookings. - Allow the customers to checkout their selections. - Allow the administrator to create new decoration packages. - Allow the administrator to update details of existing decoration packages. - Allow the administrator to delete decoration packages.
4.	Transaction module	- Allow the customers to make online payments for their bookings. - Allow the customers to receive payment receipts. - Allow the administrator to manage payment
5.	User and administrator module	- Allow users to view their profile and account details. - Allow users to alter their profile and account details. - Allow the administrator to view bookings made by users. - Allow the administrator to update the statuses of bookings.

According to Table 3, the registration and login module handles account creation and login processes. The rental module allows customers to select and manage rental items, while the decoration package module handles decoration bookings. The transaction module supports online payments and receipt management. The user and administrator module enables profile management and booking oversight.

Table 4 *Non-Functional Requirements*

No.	Requirements	Description
1.	Performance	- The system must be useful, user-friendly and easy to navigate. - The system must load pages or features within 60 seconds. - The system must handle multiple users simultaneously.
2.	Operational	- The system must ensure uninterrupted service at all times. - The system must process bookings and payments efficiently. - The system must send automated confirmations for bookings. - The system must maintain connection between the mobile app and the admin dashboard.
3.	Security	- The system must protect user data with encryption protocols. - The system must restrict access to registered customers only. - The system must limit access to sensitive data to authorized personnel only.
4.	Cultural and political	- The system must use Ringgit Malaysia (MYR) as the default currency. - The system must ensure all content is culturally appropriate.

Table 4 provides the non-functional requirements for the Moxsha System, focusing on performance, operational efficiency, security, and cultural considerations. These ensure the system operates efficiently, protects user data, and follows local norms, to improvise the overall user experience.

Table 5 *User Requirements*

No.	User Requirements
1.	All users must have an account with a valid email address and password to access the system.
2.	Customers should be able to log out of the application.
3.	Customers should be able to manage their profiles by updating personal information.
4.	Customers should be able to change their account password.
5.	Customers should be able to view the rental items and decoration packages available.
6.	Customers should be able to view the available dates for booking services.
7.	Customers should be able to select their preferred booking date for services.
8.	Customers should be able to book rental equipment and decoration packages.
9.	Customers should be able to view the details of their booked services.
10.	Customers should be able to download payment receipts through the application.
11.	Administrator should be able to create new details of services and packages available.
12.	Administrator should be able to update the details of services and packages available.
13.	Administrator should be able to delete the details of services and packages that are unavailable.
14.	Administrator should be able to manage and monitor booking details from customers.
15.	Administrator should be able to update the service status for customer bookings.

Table 5 outlines the key requirements for both users and administrators within the system. Users must create accounts with valid credentials, manage profiles, view and book rental items and decoration packages, and make payments through the application. Administrator are responsible for creating, updating, and managing service and package details, as well as monitoring and managing customer bookings and service statuses.

3.2 System Analysis

The use case diagram identifies the primary actors involved, who are customers and administrators, by outlining their interactions with the system's features. The use case helps improve the GUI wireframes, making it easier to implement, test, and refine the system in different ways [18].

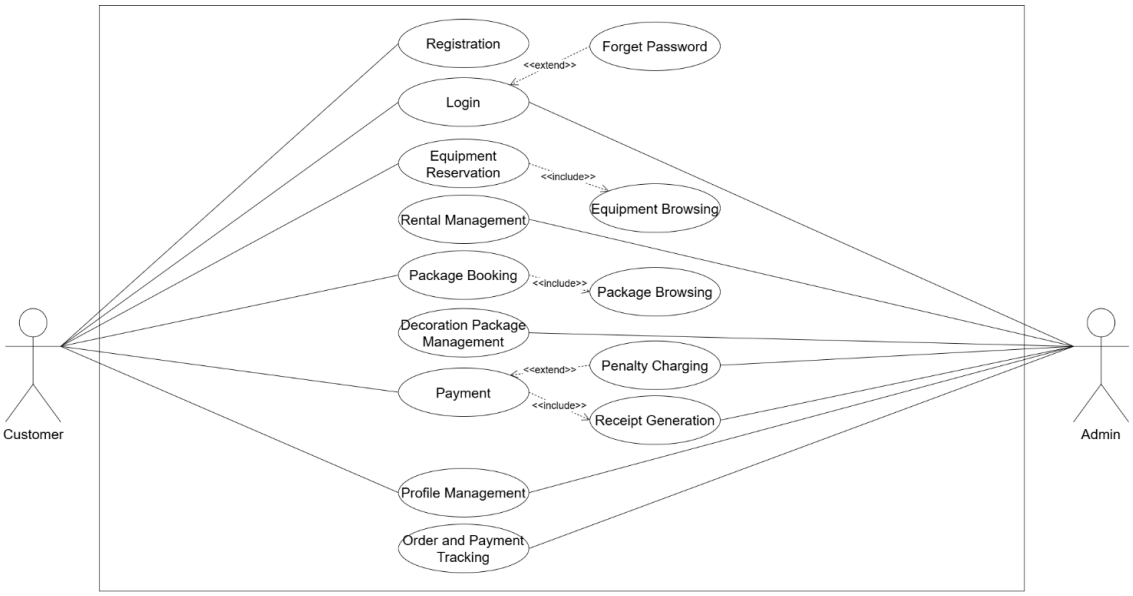


Fig. 2 *Use Case Diagram of Moxsha System*

Figure 2 shows the use case diagram for the Moxsha System. Customers can register, log in, reset passwords, browse services, reserve equipment, and book packages. They can complete payment transactions, receive receipts as confirmation, and track their orders. Additionally, customers can manage their profiles by updating personal details like name, email, and contact number. Administrators can log in to manage the company's profile, update organizational information, handle equipment and package details, and ensure that payment processing and order tracking work efficiently.

Appendix A present the UML class diagram of the Moxsha System, detailing its key entities, attributes, and the relationships between them. It shows how customers and administrator interact with the system to manage packages bookings, equipment rentals, payments, shipping, and notifications. Each class is defined with specific methods and attributes to support system functionality. This model improves the software quality by providing a systematic approach to analysis and design, making it an essential part of the development process [19].

3.3 System Design

This section outlines the Moxsha System design by focusing on its architecture, schema table, and interface design.

3.3.1 System architecture

Figure 3 shows the client-server architecture of the Moxsha System. Customers and admins access the system through a mobile app or web app on the client side.

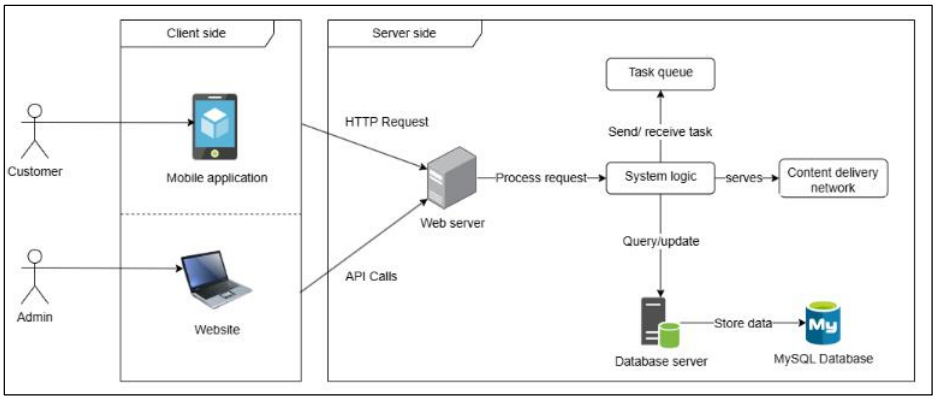


Fig. 3 *System Architecture of Moxsha System*

The web server handles incoming requests and passes them to the application logic, which processes the tasks and updates the MySQL database accordingly. This architecture is essential to maintain better execution and responsiveness in client-server interactions, as a high access rate by clients directly impacts the server performance [20].

3.3.2 Schema Table

The schema table defines the structure of the Moxsha System's, ensuring it maintains an accurate organization of data and establishes a clear relationship between the key components of the system [21]. The following are the tables from the database that have been designed:

i. `admins (id, name, email, password, remember_token, created_at, updated_at)`
ii. `admin_notes (id, text, done, created_at, updated_at)`
iii. `carts (id, user_email, item_id, price, quantity, total, start_rental, end_rental, created_at, updated_at)`
iv. `cat (id, category, image, created_at, updated_at)`
v. `decoration_carts (id, created_at, updated_at)`
vi. `items (id, subcat_id, item, stock, price, image, visibility, description)`
vii. `items_image (id, item_id, image, created_at, updated_at)`
viii. `notifications (id, title, message, type, data, user_email, is_read, created_at, updated_at)`
ix. `order (id, user_email, user_name, address, total_amount, status, delivery_date, created_at, updated_at)`
x. `order_items (id, order_id, item_id, item_name, price, quantity, total, start_rental, end_rental, return_status, created_at, updated_at)`
xi. `package_images (id, package_id, image, created_at, updated_at)`

xii. `package_orders (id, package_id, user_email, start_rental, end_rental, quantity, total_price, status, selected_options)`

xiii. `package_order_carts (id, user_email, package_id, quantity, start_rental, end_rental, selected_options, price_per_day, total_price, created_at, updated_at)`

xiv. `packages (id, package_name, description, theme_colors, hall_sizes, guest_counts, lightings, image, price, created_at, updated_at)`

xv. `penalties (id, order_id, missing_items, damaged_items, damage_percent, late_days, missing_charge, damaged_charge, late_charge, total_charge, notes, status, created_at, updated_at)`

xvi. `penalty_percentages (id, label, days_min, days_max, percentage, created_at, updated_at)`

xvii. `sales (id, item_id, quantity, created_at, updated_at)`

xviii. `subcat (id, category_id, subcat_name)`

xix. `user_addresses (id, user_email, name, phone, street, postal_code, city, state, is_default, created_at, updated_at)`

xx. `users (id, username, firstName, lastName, email, phoneNo, password, remember_token, created_at, updated_at)`

xxi. `wishlists (id, user_email, item_id, created_at, updated_at)`

3.3.3 Interface Design

This section presents the customer and admin interface designs of the Moxsha system, created in Draw.io.

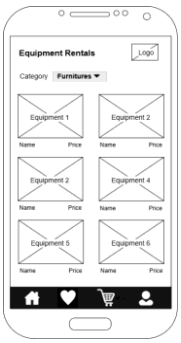


Fig. 4 *Services Interface*

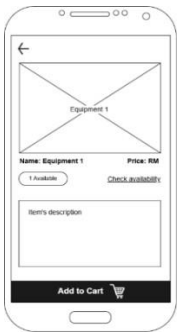


Fig. 5 *Rental/ Booking Interface*

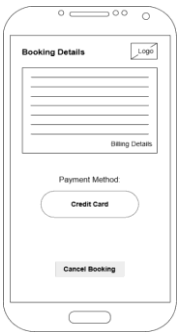


Fig. 6 *Payment Interface*

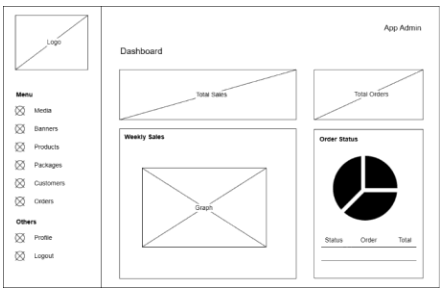


Fig. 7 *Admin Dashboard*

Figure 4 to Figure 7 shows the main four Moxsha system's interfaces, the services interface displays available equipment and decoration packages, the rental and booking interface allows users to select items and set dates, the payment interface handles secure transactions and receipts, and the admin dashboard manages orders, services, and overall system performance.

4. Results and Discussion

Results and discussion present the implementation outcomes of each functional module in the Moxsha System.

4.1 Implementation

This section explains how each module of the Moxsha System was developed, including the main source code and user interface (UI) for both the customer and admin sides.

```
$data = new User;
$data->firstName = $request->firstName;
$data->lastName = $request->lastName;
$data->username = $request->username;
$data->email = $request->email;
$data->password = $request->password;
$data->phoneNo = $request->phoneNo;
```

Fig. 8 *Customer Account Registration Source Code*

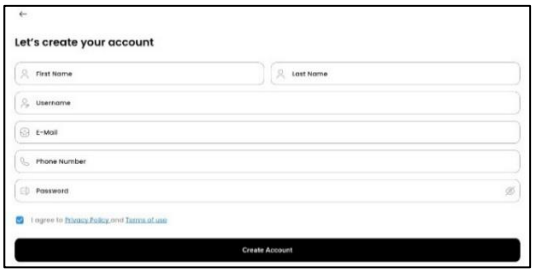


Fig. 9 *Customer Account Registration User Interface*

```
$result = User::where('email', $req->email)->first();

if ($result && Hash::check($req->password, $result->password)) {
    return response()->json([
        'success' => true,
        'id' => $result->id,
        'firstName' => $result->firstName,
        'lastName' => $result->lastName,
        'username' => $result->username,
        'email' => $result->email,
        'phoneNo' => $result->phoneNo,
        'token' => $result->createToken('authToken')->plainTextToken, // Generate a tokens
    ]);
}
```

Fig. 10 *Customer Account Login Source Code*

Fig. 11 *Customer Account Login User Interface*

```
if ($result) {
    if ($req->password == $result->password) {
        return response()->json([
            'success' => true,
            'server_ip' => $serverIpAddress,
            'id' => $result['id'],
            'name' => $result['name'],
            'email' => $result['email'],
            'password' => $result['password'],
            'token' => $result,
        ]);
    }
    return response()->json([
        'success' => false,
    ]);

}

return response()->json([
    'success' => false,
    'token' => $result,
```

Fig. 12 *Administrator Account Login Source Code*

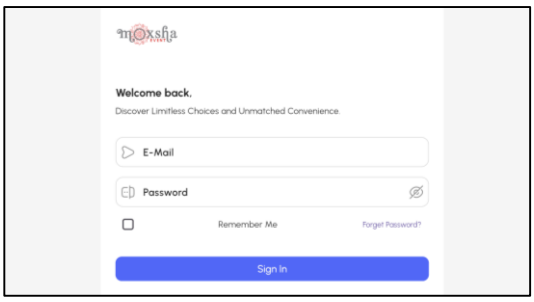


Fig. 13 *Administrator Account Login User Interface*

Figure 8 to Figure 13 show the implementation of the account registration and login module for both customers and administrators. Customers must fill in personal details and verify via OTP before accessing the system. Admins can log in and reset passwords through secure email verification.

```
$product = Items::find($id);
if (!$product) {
    return response()->json([
        'success' => false,
        'message' => 'Product not found'
    ], 404);
}

if ($request->hasFile('image')) {
    $image = $request->file('image');
    $filename = \Str::uuid() . '.' . $image->getClientOriginalExtension();
    $image->move(public_path('img/items'), $filename);
    $product->image = "/img/items/$filename";
}


$product->item = $request->item;
$product->description = $request->description;
$product->subcat_id = $request->subcat_id;
$product->stock = $request->stock;
$product->price = $request->price;
```

Fig. 14 *Equipment Rental Source Code*

Fig. 15 *Equipment Rental User Interface*

```
// Create item record
$item = Items::create([
    'item' => $validated['item'],
    'description' => $validated['description'] ?? '',
    'subcat_id' => $validated['subcat_id'],
    'stock' => $validated['stock'],
    'price' => $validated['price'],
    'image' => $imagePath,
    'visibility' => $visibility,
]);
```

Fig. 16 *Equipment Management Source Code for Products*

Fig. 17 *Equipment Management User Interface*

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1624-1643

Figure 14 to Figure 17 illustrate the equipment rental and management module for both customers and administrators. Customers can browse rental items, view stock, add to cart, and place orders with deposit payment. Penalties are applied for late or damaged returns. Admins manage rental categories, items, images, and stock.

```php
public function show($id)
{
    $package = Package::find($id);
    if (!$package) {
        return response()->json(['success' => false, 'message' => 'Package not found'], 404);
    }
    return response()->json($package);
}
```

Fig. 18 *Decoration Module Source Code*

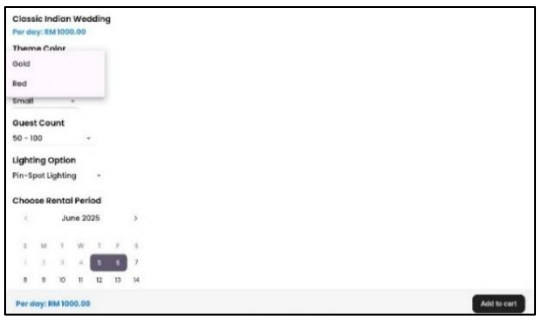


Fig. 19 *Decoration Package User Interface*

```php
// Create package records
$package = Package::create([
    'package_name' => $validated['package_name'],
    'description' => $validated['description'] ?? '',
    'price' => $validated['price'],
    'theme_colors' => json_encode($validated['theme_colors'] ?? []),
    'hall_sizes' => json_encode($validated['hall_sizes'] ?? []),
    'guest_counts' => json_encode($validated['guest_counts'] ?? []),
    'lightings' => json_encode($validated['lightings'] ?? []),
    'image' => $imagePath,
]);
```

Fig. 20 *Decoration Package Management Source Code for Attributes*

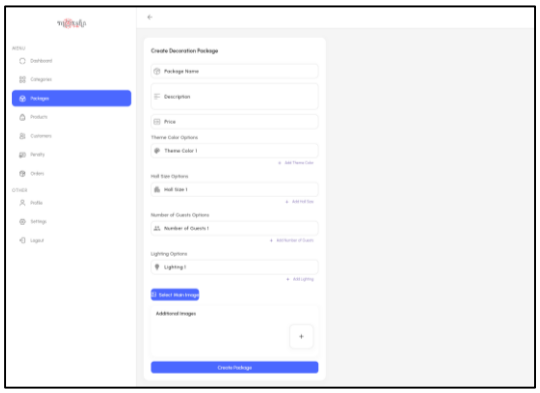


Fig. 21 *Decoration Package Management User Interface for Attributes*

Figure 18 to Figure 21 present the decoration package module. Customers can choose event dates, select packages, customize attributes, and save favorites. Admins manage package categories, images, and customizable options.

```php
DB::beginTransaction();

try {
    // Decode incoming JSON arrays safely
    $items = $request->has('items') ? json_decode($request->items, true) : [];
    $packages = $request->has('packages') ? json_decode($request->packages, true) : [];

    // Check if both are empty → block empty checkout
    if (empty($items) && empty($packages)) {
        return response()->json([
            'success' => false,
            'message' => 'Cannot place an empty order.'
        ], 400);
    }

    $orderId = null;
    $packageOrderIds = [];
```

Fig. 22 *Source Code for Booking Payment*

Fig. 23 *Interface for Booking Payment*

Figure 22 and Figure 23 show the source code and user interface for the booking payment module. This module manages payment and penalty transactions. Customers make payments and receive digital receipts. Admins verify payments, apply penalties if needed, and generate transaction records.

```php
public function show($id)
{
    $order = Order::with('items')->find($id);
    if (!$order) {
        return response()->json(['message' => 'Order not found'], 404);
    }
    return response()->json($order);
}
```

Fig. 24 *Order Management Source Code*

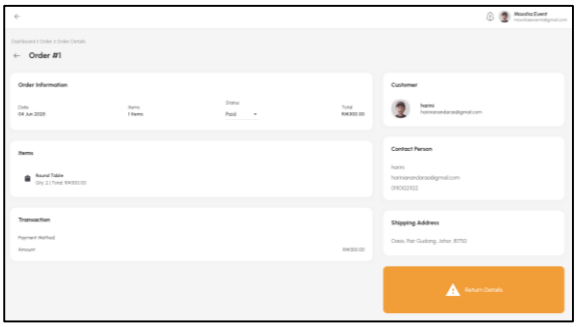


Fig. 25 *Order Management User Interface*

User and administration management module includes order tracking, penalty settings, payment methods, and customer data management, as shown in Figure 24 and Figure 25. Admins update order statuses, manage penalties, control payment options, and view customer histories.

4.2 Functional Testing

Functional testing was conducted on all major modules of the Moxsha System to verify that each implemented function behaves as expected when used by both customers and administrators. The results of these tests are presented in detailed tables, and the focus of analysis is based on actual system behaviors recorded during testing.

Table 6 *Functional Test Cases for Customer Registration and Login Module*

Test Case	Test Cases Description	Expected Result	Actual Result
1	Register a new customer with valid details and OTP.	Account is created after correct OTP.	Account successfully created.
2	Register using an already registered email.	Registration blocked with error message.	Registration blocked, error shown.
3	Login with valid email and password.	Registration fails with error message.	Error message shown.
4	Login with valid customer email and password.	User logged in and redirected to homepage.	Login successful, redirected.
5	Login with invalid details.	Login denied with error message.	Access denied, error shown.
6	Request password reset using registered email.	Verification code sent to email.	Verification code received.
7	Enter correct code and new password.	Password reset successfully.	Password reset successful.
8	Enter incorrect verification code during reset.	Password reset fails with error.	Error message shown.

Table 6 outlines the test cases related to customer registration, login, and password reset processes. It verifies the system's ability to handle user account creation, authentication, and account recovery securely and accurately. Similar functionality applies to administrators, who are registered in advance and can securely log in and update their passwords when necessary.

Table 7 *Functional Test Cases for Customer Rental Module*

Test Case	Test Cases Description	Expected Result	Actual Result
1	Customer selects a valid rental date range.	Date range accepted and available for booking.	Date range accepted and processed correctly.
2	Customer views rental items by selected service type.	List of available items shown.	Available items displayed.
3	Customer checks item stock quantity.	Accurate stock information shown.	Stock shown correctly.

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1624-1643

Table 8: (cont.)

Test Case	Test Cases Description	Expected Result	Actual Result
4	Customer adds items to cart.	Items added successfully to cart.	Items added to cart.
5	Customer edits cart contents.	Cart updates reflect changes.	Cart updated correctly.
6	Customer adds items to favorites.	Items saved in favorites list.	Favorites list updated.
7	Customer views top rental products.	Top products displayed on homepage.	Top products shown correctly.
8	Customer cancels reservation before payment.	Reservation removed, cart updated.	Reservation cancelled, cart cleared.
9	Customer checks out with valid payment.	Payment processed, booking confirmed.	Payment accepted, booking confirmed.
10	Customer selects unavailable rental dates.	Error message about date unavailability.	Error displayed for unavailable dates.

Table 7 presents the functional tests for customer interactions with the rental system. It includes service selection, rental date selection, viewing items, managing the cart and favourites, and completing or cancelling bookings.

Table 9 *Functional Test Cases for Administrator Rental Module*

Test Case	Test Cases Description	Expected Result	Actual Result
1	Admin creates a rental item under a service type.	Item added and visible under the selected service.	Rental item created and visible to customers.
2	Admin updates rental item details.	Changes saved and reflected in the system.	Details updated successfully.
3	Admin deletes a rental item.	Item removed and no longer shown to customers.	Rental item deleted from the list.
4	Admin hides a rental item.	Item hidden from customers but kept in system.	Item hidden on customer side, visible to admin.
5	Admin unhides a hidden rental item.	Item becomes visible to customers again.	Item visible to customers.
6	Admin adds a new rental category.	New category created and listed.	Category added successfully.
7	Admin adds a subcategory under a rental category.	Subcategory created under selected category.	Subcategory created and linked correctly.
8	Admin assigns items to subcategories.	Items appear under correct subcategory.	Items correctly categorized.

Table 8 describes the administrative functionalities for managing rental items. It includes creating, updating, hiding, deleting, and categorizing rental items to ensure they are properly maintained in the system.

Table 10 *Functional Test Cases for Customer Decoration Package Module*

Test Case	Test Cases Description	Expected Result	Actual Result
1	Customer selects a valid event date.	Event date is validated and accepted.	Event date accepted successfully.
2	Customer views available decoration packages.	All decoration packages are displayed.	Decoration packages displayed correctly.

Table 11: (cont.)

Test Case	Test Cases Description	Expected Result	Actual Result
3	Customer selects and customizes decoration packages.	Customizations saved and shown in order summary.	Customizations saved successfully.
4	Customer edits shopping cart before checkout.	Cart updates based on user changes.	Cart updated with correct details.
5	Customer provides venue address and selects payment method.	Address and payment method saved.	Address and payment info recorded.
6	Customer completes payment for booking.	Payment processed and booking confirmed with receipt.	Payment successful and confirmation received.
7	Customer cancels decoration booking before payment.	Booking stays in cart but marked as cancelled.	Booking cancelled and cart updated accordingly.

Table 9 covers functional tests for customers interacting with decoration packages. It ensures that customers can select, customize, and book decoration services, manage their cart, and complete or cancel bookings effectively.

Table 12 *Functional Test Cases for Administrator Decoration Package Module*

Test Case	Test Cases Description	Expected Result	Actual Result
1	Administrator creates a new decoration package with complete details.	Package is saved and shown to customers.	Package created successfully.
2	Administrator updates decoration package details and attributes.	Changes saved and visible with updated options.	Package updated successfully.
3	Administrator deletes a decoration package.	Package removed and no longer available.	Package deleted successfully.
4	Administrator manages customization options.	Options saved and linked to packages.	Customization attributes managed successfully.

Table 10 lists test cases related to managing decoration packages by administrators. It includes package creation, updates, deletion, customization management, and visibility controls.

Table 13 *Functional Test Cases for Customer Transaction Module*

Test Case	Test Cases Description	Expected Result	Actual Result
1	Customer completes payment including deposit and penalties.	Payment is processed and booking status is updated.	Payment processed and status updated.
2	Customer selects a payment method during checkout.	Payment method is accepted and used.	Payment method accepted and used.
3	System generates a digital receipt after payment.	Receipt is generated and sent instantly.	Digital receipt generated and delivered immediately.
4	Customer with unpaid penalties tries to place a new order.	System blocks the order and prompts to settle penalties.	Order blocked with penalty notification.
5	Customer pays penalties to enable new orders.	Penalties cleared and ordering access restored.	Penalties paid, new orders allowed.

Table 11 documents test cases for customer payment processes. It covers selecting payment methods, completing payments, receiving receipts, handling penalties, and ensuring order restrictions are enforced or lifted based on payment status.

Table 14 *Functional Test Cases for Administrator Transaction Module*

Test Case	Test Cases Description	Expected Result	Actual Result
1	Administrator sets penalty parameters.	Parameters are saved successfully in the system.	Penalty parameters saved successfully.
2	System calculates penalty based on damage percentage.	Penalty charges are calculated automatically based on set parameters.	Penalty charges calculated correctly.

Table 12 includes administrator-side transaction functionalities such as assigning penalty parameters and ensuring that penalty charges are automatically calculated based on damage or policy rules.

Table 15 *Functional Test Cases for User Management*

Test Case	Test Cases Description	Expected Result	Actual Result
1	User updates personal and contact details.	Changes are saved and shown in user profile.	Profile updated successfully. .
2	User resets password via the reset process.	Password reset completed after verification.	Password reset successfully.
3	User adds or updates delivery/event addresses.	Addresses saved and shown correctly in profile.	Addresses saved and displayed.
4	User views booking history and order status.	All details displayed accurately.	Booking/order history visible.
5	User receives booking and payment notifications.	Notifications sent promptly with correct info.	Notifications received correctly.

Table 13 includes test cases for user account management, such as updating profiles, resetting passwords, managing addresses, viewing order histories, and receiving system notifications.

Table 16 *Functional Test Cases for Administration Management*

Test Case	Test Cases Description	Expected Result	Actual Result
1	Admin tracks customer bookings and order details.	Booking/ order data shown accurately.	Booking/order data displayed correctly.
2	Admin updates booking and payment statuses.	Status changes saved and reflected system-wide.	Status updates successful.
3	Admin resets passwords.	Password reset completed with notifications.	Password reset successful.
4	Admin updates penalty charge rules or amounts.	Changes saved and used in penalty calculations.	Penalty charge updated.

Table 14 describes the administrative functions for overseeing customer orders and payment statuses, managing user accounts, and maintaining payment methods and penalty configurations.

4.3 Integration Testing

Integration testing demonstrates that all related modules within the Moxsha System work together as a unified and reliable workflow for both customer and admin functionalities.

4.3.1 Customer

Customer integration testing ensures that all customer-related modules in the Moxsha System function seamlessly together.

Table 17 *Customer Integration Test Cases*

Test Case	Test Cases Description	Modules Involved	Expected Result	Actual Result
1	Customer registers, verifies OTP, and logs in.	Registration and Login	Account created and login successful.	Account created and redirected to home page.
2	Customer views rental items and adds to cart.	Rental	Items listed, stock shown, cart updated.	Items displayed and added to cart.
3	Customer updates cart and proceeds to checkout.	Rental, Transaction	Cart updated and payment initiated.	Cart reflects changes and payment screen shown.
4	Customer completes payment and receives booking receipt.	Rental, Transaction	Payment processed, receipt generated.	Payment completed and receipt displayed.
5	Customer customizes decoration and makes payment.	Decoration, Transaction	Customization saved and payment successful.	Customization and payment completed.
6	Customer tries booking decoration on a booked date.	Decoration	Booking blocked with error message.	System blocked duplicate booking.
7	Customer with unpaid penalty tries to place order.	Rental, Transaction	Booking blocked until penalty cleared.	Order blocked with penalty notice.
8	Customer pays penalty and places a new order.	Transaction, Rental	Order allowed after penalty cleared.	New booking processed successfully.
9	Customer views booking history and status.	User, Transaction	History displayed accurately.	Records shown correctly.

The results presented in Table 15 confirm that the customer modules in the Moxsha System, including registration, authentication, item browsing, booking, payment, and profile management, work together correctly without errors. The tests verified that data flows correctly between modules and that each process behaves as expected from start to finish.

4.3.2 Admin

Admin integration testing focuses on verifying that administrative functions operate consistently across connected modules. It ensures that tasks performed by the admin, such as managing content, updating records, and configuring rules, trigger the correct system responses.

Table 18 *Admin Integration Test Cases*

Test Case	Test Cases Description	Modules Involved	Expected Result	Actual Result
1	Admin logs in with valid credentials.	Admin Login	Login successful, dashboard shown.	Admin logged in and dashboard displayed.
2	Admin creates rental item with category and subcategory.	Admin Rental, Rental	Item shown under correct category.	Item appears for customer under correct category.

Table 19: (cont.)

Test Case	Test Cases Description	Modules Involved	Expected Result	Actual Result
3	Admin hides a rental item.	Admin Rental, Rental	Item not visible to customer.	Penalty charge updated.
4	Admin creates decoration package with options.	Admin Decoration, Decoration	Package listed with customization options.	Package and options shown to customer.
5	Admin updates decoration attribute.	Admin Decoration, Decoration	Updated attribute reflected in UI.	Changes visible to customer.
6	Admin sets penalty rules for returns or damages.	User and Administrator, Transaction	Penalty auto-calculated correctly.	Penalty calculated based on rules.
7	Admin updates booking status.	User and Administrator, Transaction	Changes visible to customer.	Status updated for both users.

The results presented in Table 16 confirm that the admin modules in the Moxsha System, including login authentication, rental and decoration management, penalty configuration, and status updates, are properly integrated. The system ensures that actions performed by the administrator are reflected accurately across related modules, supporting backend operations and consistent communication with the customer-facing components.

4.4 User Acceptance Testing (UAT)

User Acceptance Testing checks if the Moxsha System meets the needs of customers and administrators, focusing on usability, feature satisfaction, and performance, with the next sections showing the results.

4.4.1 Customer

The results of the user acceptance testing for the customer side of the Moxsha system are presented below. This testing aimed to gather user feedback on the system's features and functionality from a customer's point of view. A total of 50 users participated, and the test was conducted using a Google Form.

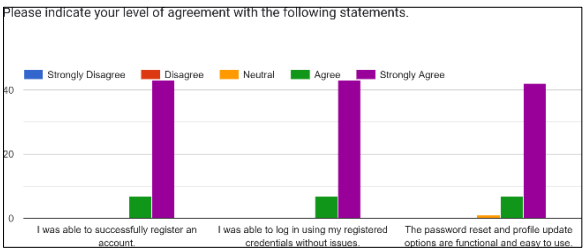


Fig. 26 *Account Registration and Profile Management*

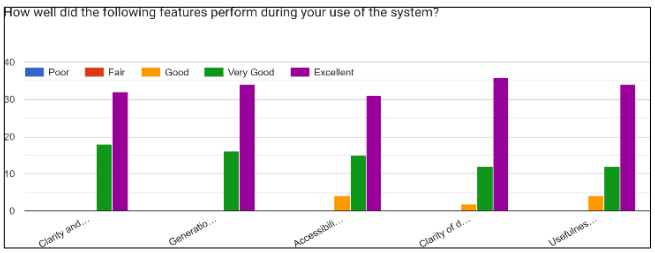


Fig. 27 *User Evaluation of Booking and Payment Features*

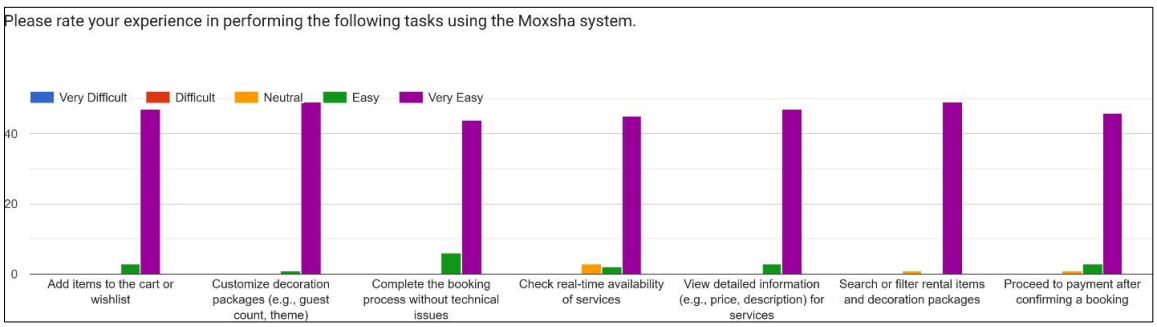


Fig. 28 *User Ratings on Task Ease in Browsing, Customization, Booking, and Payment*

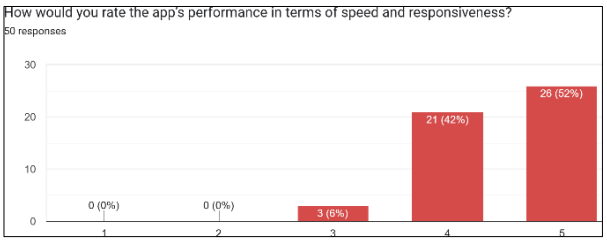


Fig. 29 *User Rating on System Speed and Responsiveness*

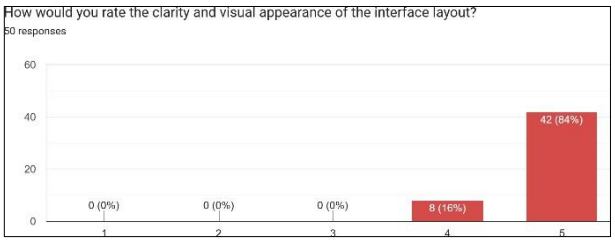


Fig. 30 *User Rating on Interface Clarity*

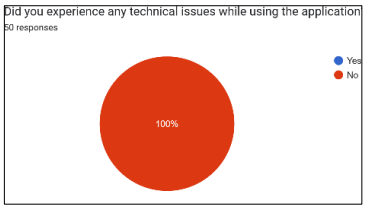


Fig. 31 *User Feedback on Technical Issues Encountered*

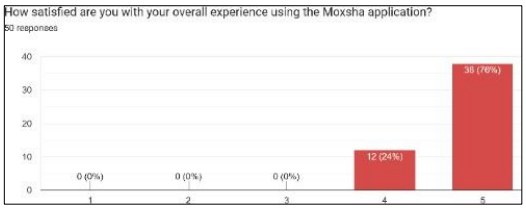


Fig. 32 *Overall User Satisfaction with the Moxsha Application*

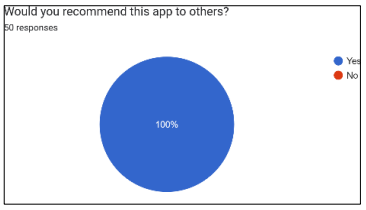


Fig. 33 *User Recommendation of the Moxsha Application*

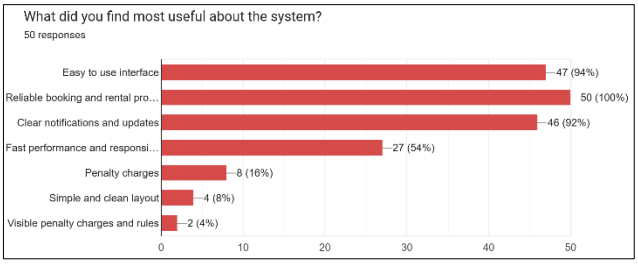


Fig. 34 *User Preferences for System Features*

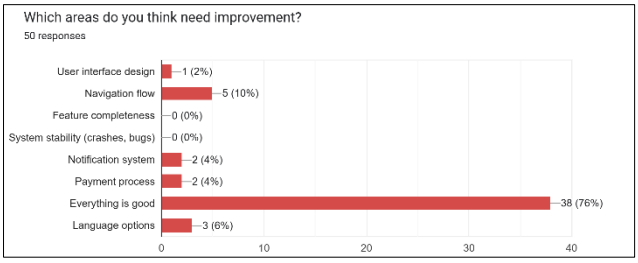


Fig. 35 *Areas for Improvement Identified by Users*

Based on the results presented in Figure 26 to Figure 35, it is evident that the Moxsha System delivers a highly satisfactory experience from the customer's perspective. Users found the system intuitive and reliable across key functions such as registration, login, service browsing, customization, booking, and payment. The majority of responses indicated strong ease of use and excellent feature performance, particularly in areas like payment processing, booking confirmations, and notification clarity. No technical issues were reported, and the system's speed, design, and responsiveness received very positive ratings.

All users expressed satisfaction with the system, with every respondent willing to recommend it to others. While most users felt no improvements were needed, several valuable suggestions were made for future enhancement, including dark mode, real-time chat, live tracking, and biometric login options. Overall, the findings confirm that the Moxsha System effectively meets user needs and expectations,
providing a seamless and dependable user experience.

4.4.2 Administrator

The admin-side User Acceptance Testing was conducted with the primary stakeholder, Mr. Saravana Kumar, to assess key administrative functions such as booking, inventory, payments, and penalties. Feedback was gathered using a structured five-point scale form evaluating usability, responsiveness, and feature effectiveness. The signed form, included in Appendix B, confirms his satisfaction and includes a suggestion to add a quick chat feature for better customer communication.

5. Conclusion

The Moxsha Equipment Rental and Decoration Services Management System has successfully addressed the challenges posed by the previous manual approach by offering a structured digital solution. The system introduced key features such as real-time availability checking, secure online payment, automated receipt generation, and integrated penalty tracking. These functions significantly enhance the customer experience while

supporting accurate and organized administrative processes. Users are able to register, make bookings, manage profiles, and receive timely updates, all through a seamless interface. Administrators are equipped with effective tools to manage rentals, monitor transactions, configure services, and maintain consistent service quality. Although the system achieves its core objectives, there is room for further enhancement to improve accessibility and user engagement. The system is currently optimized for Android devices and core functionalities, with opportunities for future enhancement in areas such as cross-platform support, advanced customization, delivery tracking, and improved offline accessibility. Planned future developments include extending platform compatibility to iOS devices, integrating real-time chat between customers and administrators, enabling delivery tracking, review or feedback submission, and providing advanced customization features for decoration services. These improvements are expected to increase system reliability and satisfaction, while also supporting the long-term growth and operational effectiveness of Moxsha Event.

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] C. Bladen, N. Wilde, J. Kennell, and E. Abson, *Events Management: An Introduction*. Routledge, Dec. 2022. doi: 10.4324/9781003102878.

[2] J. Allen, R. Harris, and L. Jago, *Festival & Special Event Management Essentials*. John Wiley & Sons, 2022, p. 422. [Online]. Available: https://books.google.com/books/about/Festival_Special_Event_Management_Essent.html?id=fbIxEAAAQBAJ

[3] N. Faliha, A. E. Siti, R. Kusdi, and K. Andriani, "Online reservation system and online customer review: Its impact on brand image, trust and hotel booking decision," *International Journal of Economics, Business and Accounting Research (IJEBAR)*, vol. 5, no. 4, Dec. 2021. doi: 10.29040/IJEBAR.V5I4.3367.

[4] I. Yunita and A. M. Harahap, "Android-based outdoor equipment rental information system," *Journal of Artificial Intelligence and Engineering Applications*, 2024. [Online]. Available: https://ioinformatic.org/

[5] R. B. Adducul, "Mobile bus ticketing system: Development and adoption," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 1.3, pp. 189–196, Jun. 2020. doi: 10.30534/ijatcse/2020/2891.32020.

[6] J. M. Magno, R. S. Pelayo, and K. T. Soberano, "Apartment rental management system for real-time transaction and task organization," *International Journal of Multidisciplinary Research and Analysis*, vol. 7, no. 7, Jul. 2024. doi: 10.47191/IJMRA/V7-I07-29.

[7] N. A. Omar, "Effect of technological innovations on the accounting practices efficiency in Kenya," *African Journal of Commercial Studies*, vol. 3, no. 2, pp. 118–126, Oct. 2023. doi: 10.59413/ajocs/v3.i2.5.

[8] G. Wilson, W. Brown, and O. Johnson, "The impact of mobile technologies on consumer behavior in retail marketing," Jul. 25, 2024. doi: 10.20944/preprints202407.2030.v1.

[9] L. T. Khrais and A. M. Alghamdi, "The role of mobile application acceptance in shaping e-customer service," *Future Internet*, vol. 13, no. 3, 2021. doi: 10.3390/fi13030077.

[10] C. Chaubey and A. Sharma, "The integrated development environment (IDE) for application development: Android Studio and its tools," *AIP Conference Proceedings*, vol. 2427, no. 1, Feb. 2023. doi: 10.1063/5.0116494.

[11] M. El, A. Tebib, M. Graa, P. Andre, O.-E.-K. Aktouf, and O.-E.-K. A. Aktouf, "A survey on secure Android apps development life-cycle: Vulnerabilities and tools," *International Journal on Advances in Security*, vol. 16, no. 1 & 2, pp. 54–71, 2023. [Online]. Available: https://hal.science/hal-04181107

[12] A. Tashildar, N. Shah, R. Gala, T. Giri, and P. Chavhan, "Application development using Flutter," Aug. 2020. [Online]. Available: https://www.irjmets.com

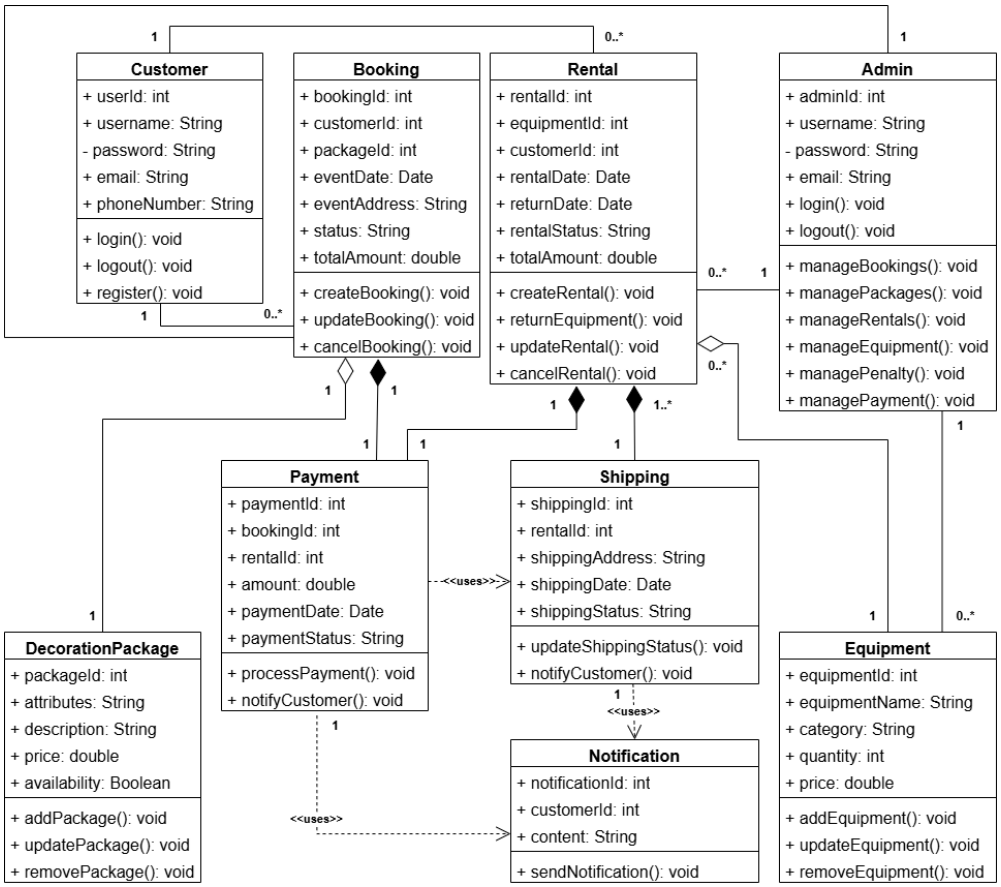
[13] A. Omonije, "Agile methodology: A comprehensive impact on modern business operations," *International Journal of Science and Research (IJSR)*, vol. 13, no. 2, pp. 132–138, Feb. 2024. doi: 10.21275/SR24130104148.

[14] N. H. Borhan, H. Zulzalil, A. Hassan, N. Hayati, and M. Ali, "Requirements prioritization in agile projects: From experts' perspectives," *Journal of Theoretical and Applied Information Technology*, vol. 15, p. 19, 2022. [Online]. Available: https://docs.google.com/forms/d/e/1FAIpQLSeDT

[15] N. Okeke, "Agile methodology: Meaning, advantages, disadvantages & more," *TargetTrend*, accessed Nov. 15, 2024. [Online]. Available: https://targettrend.com/agile-methodology-meaning-advantages-disadvantages-more/

[16] M. A. Khan, M. S. Khan, I. Khan, S. Ahmad, and S. Huda, "Non-functional requirements identification and classification using transfer learning model," *IEEE Access*, vol. 11, pp. 74997–75005, 2023. doi: 10.1109/ACCESS.2023.3295238.
[17] M. J. Hossain and M. S. Islam, "Planning a user-centered process improvement for academic libraries: A pathway of satisfying users' needs," *Library Philosophy and Practice*, 2021. [Online]. Available: https://digitalcommons.unl.edu/libphilprac
[18] V. Vranić, J. Lang, M. L. Nores, J. J. P. Arias, J. Solano, and G. Laseca, "Use case modeling in a research setting of developing an innovative pilgrimage support system," *Universal Access in the Information Society*, Nov. 2023. doi: 10.1007/s10209-023-01047-1.
[19] A. Hafeez, M. Ahmed, M. Furqan, W.-U.- Rehaman, and I. Husain, "Importance and impact of class diagram in software development," *Indian Journal of Science and Technology*, vol. 12, no. 25, pp. 1–4, Jul. 2019. doi: 10.17485/ijst/2019/v12i25/145739.
[20] D. S. Pereira, L. F. V. Bezerra, J. S. Nunes, I. M. B. Filho, and F. A. S. Lopes, "Performance efficiency evaluation based on ISO/IEC 25010:2011 applied to a case study on load balance and resilience," *Workshop de Testes e Tolerância a Falhas (WTF)*, pp. 108–120, May 2023. doi: 10.5753/WTF.2023.787.
[21] V. Yesin, M. Karpinski, M. Yesina, V. Vilihura, and K. Warwas, "Ensuring data integrity in databases with the universal basis of relations," *Applied Sciences (Switzerland)*, vol. 11, no. 18, Sep. 2021. doi: 10.3390/app11188781.

Appendix A: UML Class Diagram



Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 1624-1643

Appendix B: Administrator UAT Feedback and Suggestion

STAKEHOLDER / EVALUATOR DETAILS

Full Name : Saravana Kumar

Organization/ Role : Moxsha Event/ Business Owner

Contact (Email/Phone) : +60 12-714 4832

INSTRUCTIONS

Please evaluate the system based on your experience by marking a check (√) in the appropriate box. Use the following scale:

1= Very Poor	2= Poor	3= Neutral	4= Good	5= Excellent

EVALUATION CRITERIA

No.	Evaluation Item	1	2	3	4	5
1	Intuitiveness of website navigation for administrative tasks.					/
2	Consistency and organization of the interface layout across admin pages.					/
3	Clarity and legibility of displayed information on the management dashboard.				/	
4	Report generation is functional and accurate.					/
5	Responsiveness and loading speed of the admin pages.				/	
6	Efficiency of system response time during data retrieval and updates.					/
7	Accessibility and reliability of the login process.					/
8	Effectiveness in managing and processing customer bookings.					/
9	Simplicity of updating and maintaining equipment inventory records.					/
10	Flexibility in configuring and modifying decoration service packages.					/
11	Transparency and accuracy of payment tracking and status updates.				/	
12	Reliability of real-time data for service and equipment availability.					/
13	Ease of accessing user profile and contact information.					/
14	Clarity and functionality of managing penalty charges for late returns, unreturned or damages order.					/

FINAL COMMENTS/ SUGGESTIONS

Please provide any additional comments or suggestions for improvement:

Honestly, this system makes my life so much easier. Managing bookings, updating decoration packages, tracking payments, and handling penalties all in one place means no more messing around with papers or files. It is really simple and everything is super organized. One thing I would love is a quick chat feature so I can answer customers questions right away based on their orders. Overall, I am really happy with how it works! Looking forward to working with you to keep enhancing it in the future.

I hereby confirm that I have tested the system and provided my feedback truthfully based on my experience.

Name: Saravana Kumar

Date: 31 May 2025