Applied Information Technology And Computer Science Vol. 5 No. 1 (2024) 849-866

AITCS

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/aitcs
e-ISSN :2773-5141

SK Farm Livestock Management System

Puvaan Raaj A/L Shankar[1], **Mohd Zainuri Saringat**[1]*

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

DOI: https://doi.org/10.30880/aitcs.2024.05.01.048
Received 19 June 2023; Accepted 17 June 2024; Available online 30 August 2024

***Abstract**: A management system is designed to aid a business in managing its operations. SK FARM is a livestock farm that buys and sells livestock. The absence of a computer or web-based system is the owner's major complaint since all administrative and business activities are completed by hand. To create, develop, and test a web-based livestock management system for SK Farm was the goal of this project. The system used computerized techniques and databases which is capable to arrange the owner's financial information, orders, employee details and order placement for the customer. Information stored in databases provides security to safeguard the data while also making adding, updating, and removing operations much simpler. The PHP programming language was used to create it. The project adhered to the technique for software development using an incremental model. The system was hosted on the AWS web hosting service, and the database used the MySQL service.*

***Keywords**: SK FARM, Management System, PHP, AWS*

1. Introduction

The economy of both the developing nations and rural livelihoods heavily rely on livestock. Producers and others involved in sometimes complex value chains rely on them for income and jobs. For the impoverished, particularly for women and pastoralist communities, they constitute a vital resource and safety net. Additionally, they are a significant source of food for billions of rural and urban people. Thornton, [1] describe livestock as animals such as cattle, sheep, cows, and other animals which are kept on a farm. To continually grow a farm business, the owners should start to integrate technologies in their businesses. A management system is the way in which an organization manages the interrelated parts of its business to achieve its objectives. These objectives can relate to several different topics, including product or service quality, operational efficiency, environmental performance, health, and safety in the workplace and many more [2].

This project proposed a specific sales and management system for a farmer who does livestock business. The owner of the farm, SK FARM is Mr. Saravana Kumar who has a total of 4 livestock farms. Currently, there are several methods used by the owner to run the business. Firstly, all business-related process is done manually. This includes recording of orders from customer, financial documentation, employee details documentation, order documentation and documentation of livestock

*Corresponding author: zainuri@thm.edu.my

stocks. Secondly, all the documents and files related to the business are kept as hardcopies inside the owner's small office. Thirdly, customers need to physically come to the farm for ordering purposes and must wait for a longer period before they receive their product.

Thus, a livestock management system for this farm is proposed to solve the business problems faced by the owner. By using a web-based system, the owner and his customers can experience an easier business dealing. Moreover, a management system will also help the owner to handle all the stock records, financial records, employee details. Additionally, this proposed system will have a feature which enables the customer to make order.

2. Related Work

2.1. Livestock Management System

Management systems are systematic frameworks created to govern an organization's policies, procedures, and processes and encourage internal improvement, according to The British Standards Institution [3]. A corporation can improve operations, reduce risk, and foster stakeholder confidence by implementing a tried-and-true management system, such as ISO 9001 Quality Certification [4].

While livestock management is a system that involves managing stocks, financial information, and several other things, there are different types of livestock management services which could include autonomous tracking of livestock, track and monitor field progress workers.

2.2. Study of SK Farm Manual Process

The paper and files are kept in a folder as part of the present manual management method. The customer's order receipt and order list were saved in folders and then organized in a storage box. According to the farm, the item's order list and receipt are divided into various files. By looking at the order lists' dates inside the files, the owner can search for the order list they need. Additionally, all financial information and stock information are preserved in papers and files. The stock information is categorized according to the type of livestock. The Financial information are also categorized into total sales, total expenses, and total revenue. The owner also receives all his order manually which is via a phone call or the WhatsApp chat application. After receiving the order, the owner will keep the details in paper and store in a storage. Figure 2.1 shows the AS-IS model for the existing spare part management system.

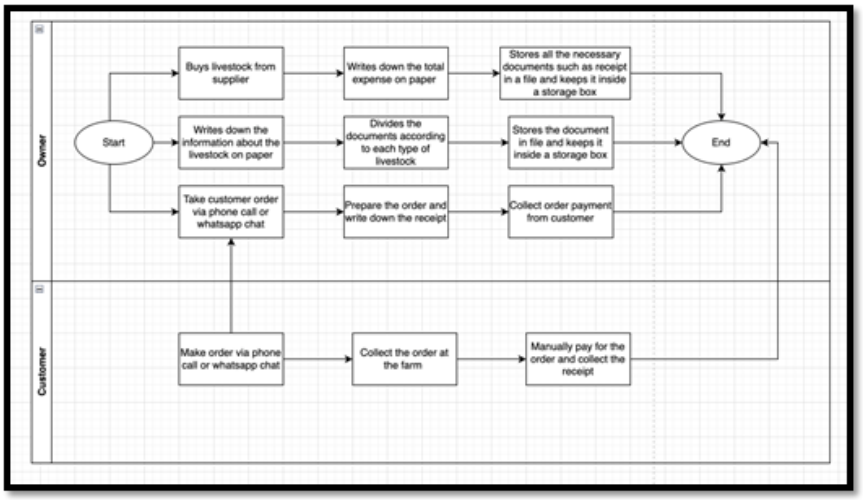


Figure 2.1: As-Is Model of SFLMS

2.3. Comparison with Existing System

Based on the discussion in the previous section, the comparison between the three systems are summarized in Table 2.1. At this section the features of system or application are compared with the proposed system. Table 2.1 discusses about the comparison between the chosen existing system and proposed system.

Table 2.1: Comparison between AgriWebb, Farmbride, Ranch Manager Open and SFLMS

Features	AgriWebb[5]	Farmbrite[6]	Ranch Manager Open[7]	SFLMS
Platform	Web-based	Web-based	Desktop Software	Web-based
Register	1.Use personal name, username, email and phone number. 2. Send activation link in email to activate account.	1.Use personal name, username, email and phone number.	none	1.Use personal name, email and phone number. 2. Send activation link in email to activate account.
Login	1. Use username and password	1. Use username and password	No	1. Use username and password
Change Profile Details	No	No	No	1. user can change their personal details. 2. Users can also change their account password.
Manage Stock	1. User chooses their preferred animal to add, update or delete it stocks. 2. User can add animals treatment details.	1. User chooses their preferred animal to add, update or delete it stocks.	1. User chooses their preferred animal to add, update or delete it stocks.	1. User chooses their preferred animal to add, update or delete it stocks. 2. The stocks is integrated with customer side for ordering purposes.
Manage Financial Information	No	1. User can view financial information such as transaction, P&L Statements and cash flow	No	1.User can view financial information such as total sales, total revenue, and total expenses.

Manage Employee details	No	No	No	1.User can add employee details and their work schedule.
Manage Order	No	No	No	1.User will be able to manage all the orders from the customer. 2. Customers are able to make order
Make Payment	No	No	No	l. User can make payment for their order by choosing their preferred payment method and input their details.

Based on the comparison in Table 2.1, it shows that every system has its advantages and disadvantages. SK Farm Livestock Management System will excel in change account details, manage financial information, manage employee details, manage order, and make payment. In terms of change account details, the users are allowed to change their personal details or password in case if they forgot and intended to change it. Manage financial information will show all the details involved with finance such as total sales total revenue and total expense. Manage employee details will display the current employee's working and their working schedule. Manage order will allow the customer and the owner to manage order. The customer will make order and the owner will manage the order Next, make payment feature will allow the customer to pay for their order and they only need to come to the farm to collect their order.

3. Methodology

The framework that is used to organize, schedule, and manage the process of constructing an information system is referred to as a system development methodology. Such frameworks have taken many different forms throughout the years, each with distinct strengths and limitations [8]. Prototyping can take many different forms, including extreme, gradual, and quick throwaway prototypes. This system was developed using an incremental development paradigm. The incremental model, sometimes referred to as the successive version model, is a widely used method of software development in which the SDLC divides the software requirements into numerous independent modules or increments (Software Development Life Cycle). Phases of the incremental process model include planning, analysis, design, development, testing and implementation.

3.1 Requirement Analysis

A business model must initially be created to conduct requirement analysis. The flow of the existing system process, as detailed in Chapter 2, can be represented as a model company. The analytical data gathered was utilized to create a use case diagram. Use case diagrams are created to demonstrate the essential functions of a system and as a reference for sequence diagrams and activity diagrams for each use case. Use case diagrams are created to demonstrate essential functions of a system and as a reference for sequence diagrams ad activity diagrams for each use case.

Functional requirement analysis is a process that shows the developer the functionalities of the modules for the proposed system. As shown in Table 4.1, there are eight modules in the systems, which

are the Register module, Login module, Change account details module, Manage livestock stock module, Manage financial information module, Manage employee details module, Manage order module, and Make payment module.

Table 3.1: The functional requirements for the system

Modules	Functionalities
Register	1) Customer register's account to enter the customer homepage.
Login	1) Owner login to the system and conduct the management process in the system. 2) Customer login to the system and conduct the ordering process in the system.
Change account detail	1) User will be able to change their personal details and account password.
Manage livestock stock	1) The owner will add, delete, and update the stocks of livestock on the farm. 2) The owner views the stock of livestock.
Manage financial information	1) The owner views the total sales of the farms. 2) The owner adds the expenses of the farms into the system. 3) The owner views the total expenses of the farms. 4) The owner views the total revenue of the farms. 5) The owner generates the financial report.
Manage employee details	1) The owner adds, deletes, and updates employee details. 2) The owner views the current employee list. 3) The owner inputs the working schedule of the employee.
Manage order	1) The customer will be able to make order. 2) The customer will be able to view the shopping cart. 3) The owner will be able to view all the orders. 4) The owner will be able to print the receipt for the orders.
Make Payment	1) The customer will make payment for their order. 2) The customer can view the receipt for the purchase.

Non-functional requirement analysis is a process that shows the developer the capabilities and constraints of the proposed system. As shown in Table 4.2, there are three non-functional requirements, which are Security, Operational, and Performance.

Table 3.2 Shows the non-functional requirement for the system

Modules	Functionalities
Security	1) The password must be in eight characters with a combination of alphabet and number. 2) The system can only be accessed with the insertion of the correct username and password.
Operational	1) The system should be able to function well at all web browsers. Customer login to the system and conduct the ordering process in the system.
Performance	1) The interactions between users and webpages should not be longer than three seconds.

3.2 Use Case Diagram

A use case diagram is a way to summarize details in a system and the users within that system [9]. The actor in the use case diagram refers to the scope in Chapter 1. According to the scope that has been stated in Chapter 1, there are a total of eight modules that would be implemented into the system. Therefore, only eight use case modules are in Figure 3.1.

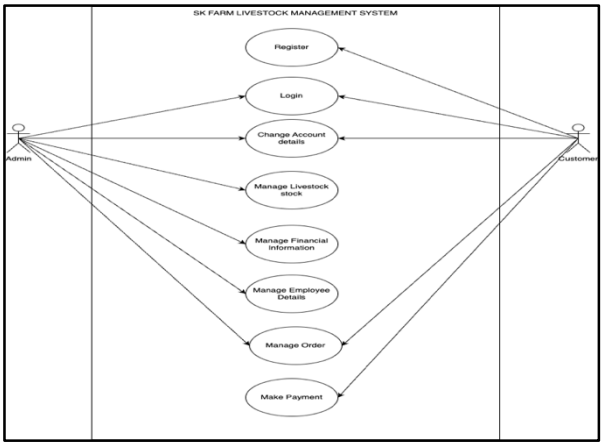


Figure 3.1: Use case diagram for SFLSM

3.3 Class Diagram

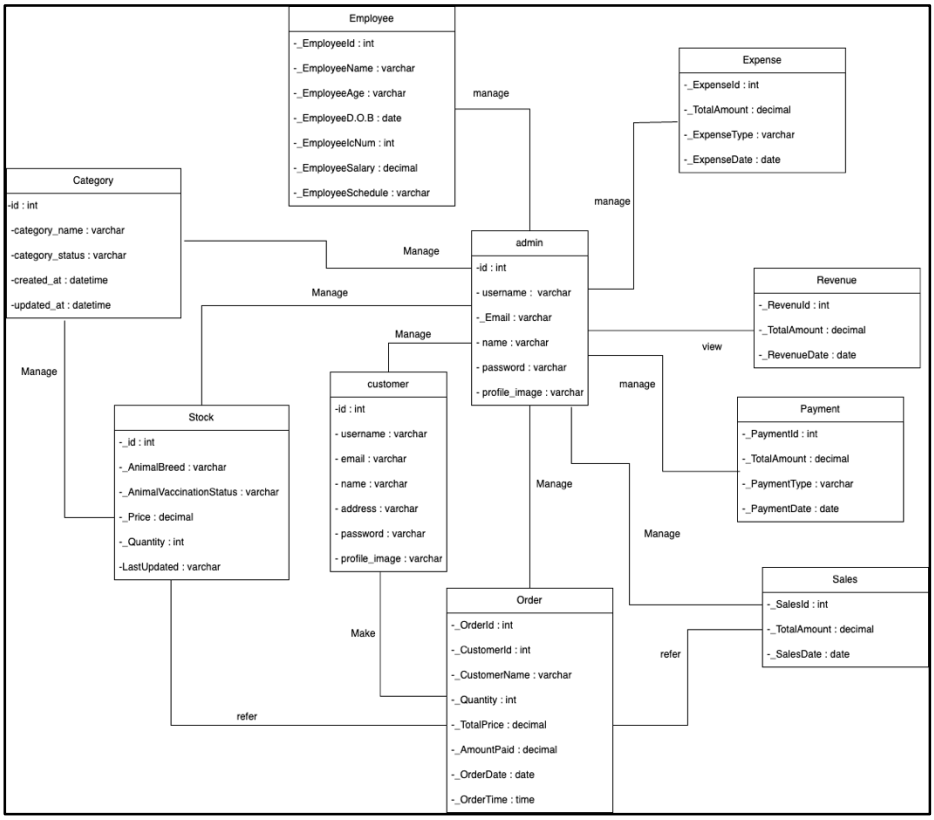


Figure 3.2 : Class diagram

3.4 System Architecture

The structural arrangement for a system is known as the architectural design. Its preparation aims to identify the main structural components in the system as well as the interactions between each of these components. SFLMS adopts the MVC system architecture. MVC (Model-View-Controller) is a software architecture design pattern that separates an application into three main components: model, view, and controller. This model is responsible for managing application data. It stores and retrieves data from a database or other data source, and it handles any logic related to the data. Display (View)

is responsible for displaying data to the user. A view is a page that users see and interact with. The Controller is responsible for handling user input and interaction. The controller accepts input from the user, communicates with the model to retrieve or update data, and updates the display with the new data. The relationship between these 3 components can be clearly seen in Figure 3.3.

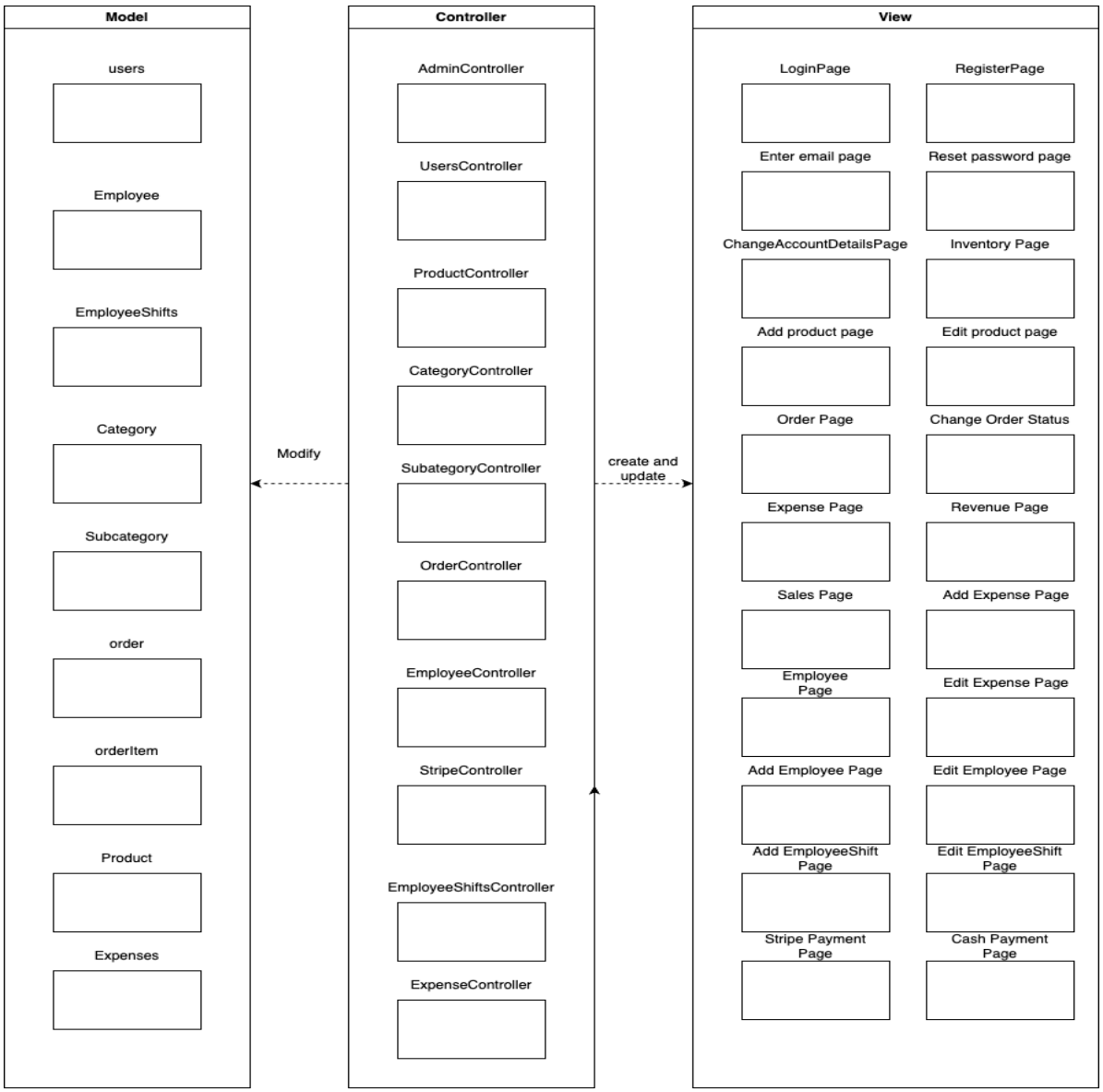


Figure 3.3: System architecture

3.4 Interface Design

User login and register interface in the system is shown in the figure 3.4 and 3.5. Both user(admin and customer) will login into the system to use the features of the system. The register page will be only shown to the customer. They should register an account in order to enter into the homepage and start ordering. Figure 3.6 shows the change account detail interface. The change account module allows the user to change their personal details and password.

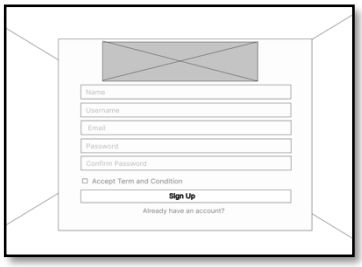


Figure 3.4: Register Account

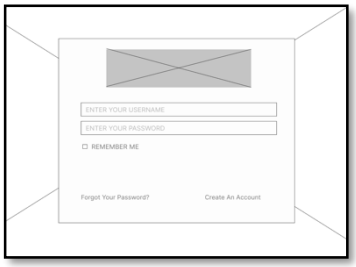


Figure 3.5: Login interface

Figure 3.6: Change account details

Figure 3.7 shows the manage livestock interface. In this module the admin can add, update and delete stock. Figure 3.8 shows the add stock interface, figure 3.9 shows the update and delete stock interface.

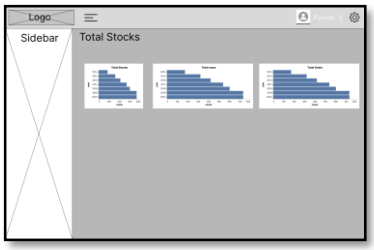


Figure 3.7: Stock Interface

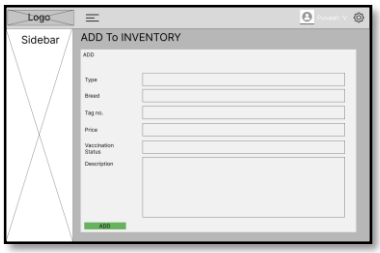


Figure 3.8: Add stock interface

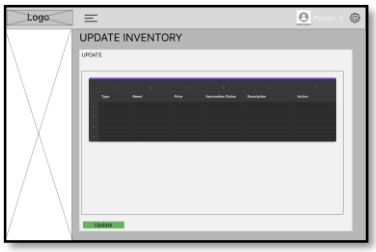


Figure 3.9: Delete stock interface

Figure 3.10 shows the total sales interface which is included in the manage financial information module. In this module the admin can add sales and expense, and view sales, expense and revenue. Figure 3.11 shows the add sales page. Figure 3.12 shows the total expense page. Figure 3.13 shows the add expenses page, lastly figure 3.14 shows the revenue page.

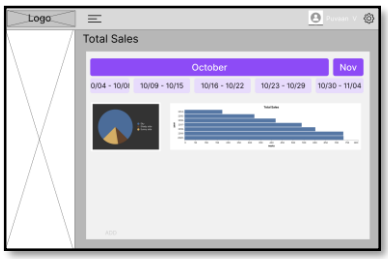


Figure 3.10: Total sales interface

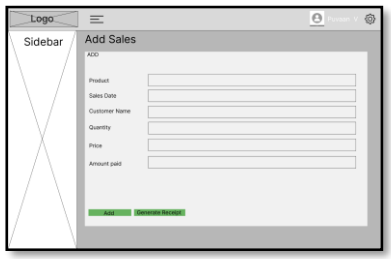


Figure 3.11: Add sales interface

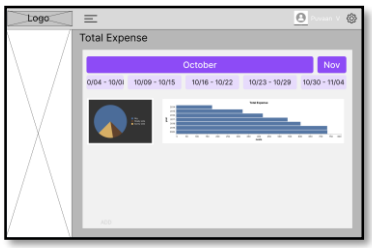


Figure 3.12: Total expenses

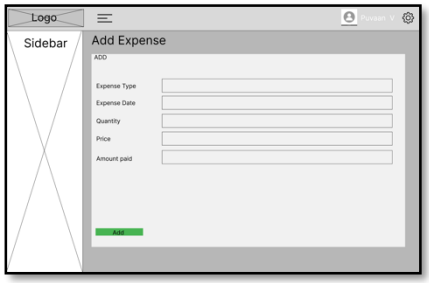


Figure 3.13: Add sales interface

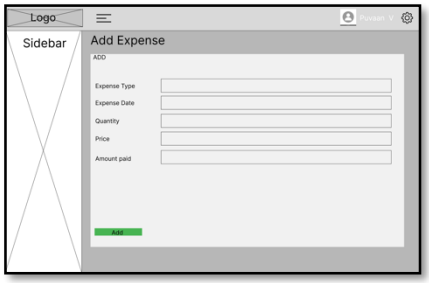


Figure 3.14: Total revenue interface

Figure 3.15 and Figure 3.16 shows the add and update employee interface. The admin will add and update the employee details. Figure 3.17 shows the working schedule of the employee. The manage order module comes with 4 function. Firstly view the total orders, add orders manually and lastly adding

order. View and add order are done by the customer while make order will be done by the customer. Figure 3.18 shows the orders summary page, figure 3.19 shows the total orders page, figure 3.20 shows the add order page and lastly figure 3.21 shows the product page where the customer make order. Lastly figure 3.22 shows the payment option interface. The customer will choose their preferred payment method and continue to make payment for their order.

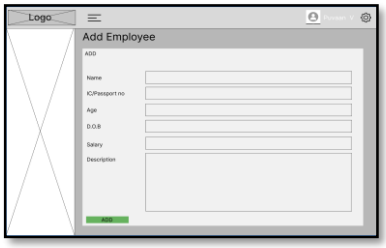


Figure 3.18: Order summary interface

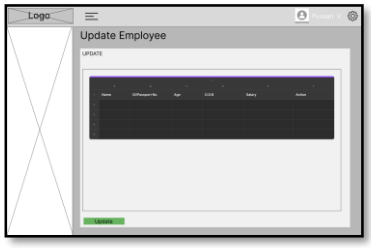


Figure 3.19: Total order interface

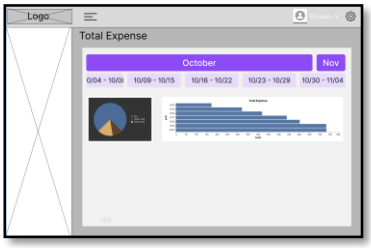


Figure 3.20: Add order page

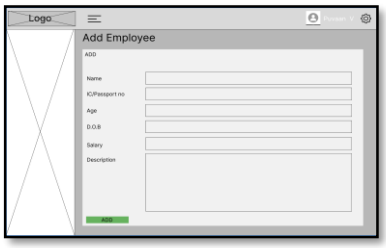


Figure 3.15: Add employee interface

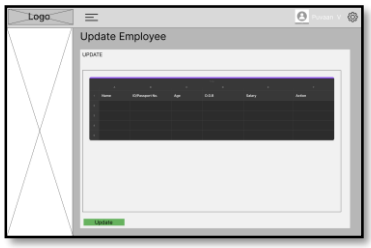


Figure 3.16: Update employee details

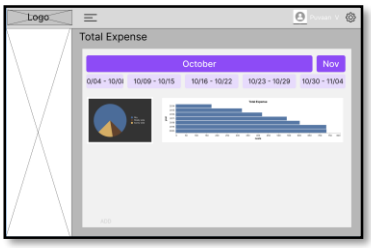


Figure 3.17: Total expenses

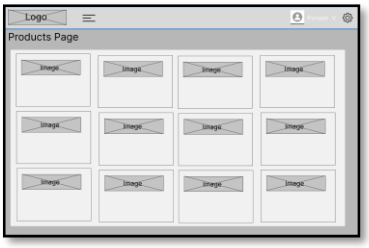


Figure 3.21: Product page

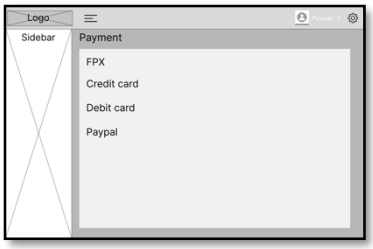


Figure 3.22: Payment Page

4. Results and Discussion

The system was primarily developed using Laravel, a powerful PHP framework, with frontend development done using HTML, CSS, and JavaScript. The most important development tools were Visual Studio Code for editing code, phpMyAdmin for managing MySQL databases, XAMPP for local testing, Composer for managing dependencies, and Git & GitHub for version control and collaboration.

4.1. Register Module

Figure 5.1 illustrates the PHP code for user registration using the Laravel framework, which gathers and validates user input before saving the new user with a hashed password in the database and logging them in before forwarding them to the home page. Figure 5.2, on the other hand, shows the user interface (UI) for the account registration page, which has input fields for a username, email address, and password. The 'Register & Submit' button triggers the code.

```php
public function store(Request $request): RedirectResponse
{
    $request->validate([
        'name' => ['required', 'string', 'max:255'],
        'email' => ['required', 'string', 'email', 'max:255', 'unique:'.User::class],
        'password' => ['required', 'confirmed', Rules\Password::defaults()],
    ]);

    $user = User::create([
        'name' => $request->name,
        'email' => $request->email,
        'password' => Hash::make($request->password),
    ]);

    event(new Registered($user));

    Auth::login($user);

    return redirect(RouteServiceProvider::HOME);
}
```

Figure 4.1: Account Registration Source Code

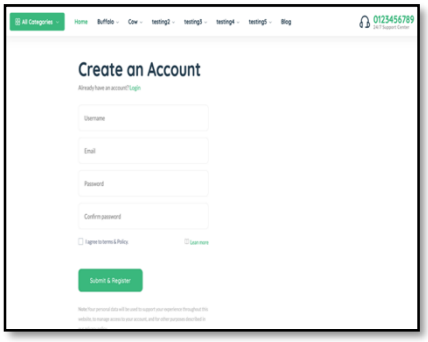


Figure 4.2: Register New User Interface

4.2. Login Module

The Laravel PHP method for handling user login is shown in Figure 5.3. It entails checking user credentials, regenerating the session, and then rerouting users to their dashboard with a success message. The "SK FARM Online Store" login page (Figure 5.4) has fields for entering your email address and password, a "Log in" button to start authentication, a "Remember Me" button for convenience, and a "Forgot Password" link for those who need to reset their password. Figure 5.5 also shows the login form, highlighting the email and password fields as well as the "Remember me" checkbox for future logins.

```php
public function store(LoginRequest $request)
{
    $request->authenticate();

    $request->session()->regenerate();

      $notification = array(
        'message' => 'Login Successfully',
        'alert-type' => 'success'
    );

    $url = '';
    if ($request->user()->role === 'admin') {
        $url = 'admin/dashboard';
    } elseif ($request->user()->role === 'vendor') {
        $url = 'vendor/dashboard';
    } elseif ($request->user()->role === 'user') {
        $url = '/dashboard';
    }

    return redirect()->intended($url)->with($notification);
}
```

Figure 4.3: User Login Source Code

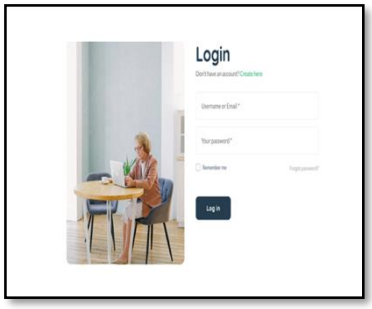


Figure 4.4: User login interface

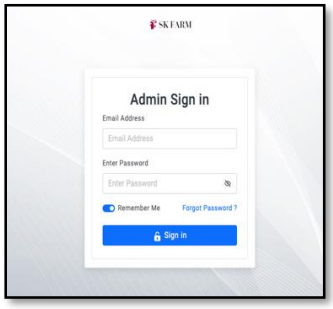
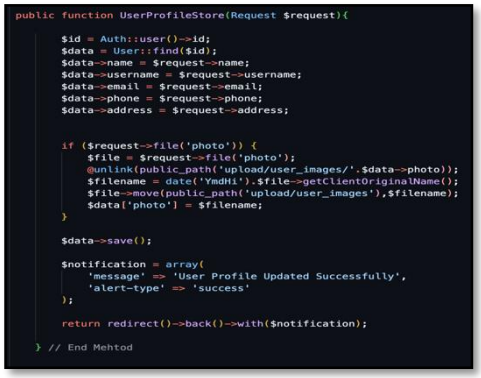


Figure 4.5: Admin Login Interface

4.3. Change Account Details Module

The Laravel function UserProfileStore, which manages user profile updates, is shown in Figure 5.6. The user's unique ID is retrieved, user data is updated, a new photo submission is handled, modifications are saved, and a confirmation message is sent. The user profile update interface is shown in Figure 5.7. It has pre-populated input boxes, an area for the profile photo, and a "Save Change" button that initiates the update. The final sentence of the text makes reference to the admin profile management page, which is an extension of the 'admin.admin_dashboard' template and makes it possible to edit data, including the profile photo, by sending a POST request to the 'admin.profile.store' route.



```
public function UserProfileStore(Request $request){

    $id = Auth::user()->id;
    $data = User::find($id);
    $data->name = $request->name;
    $data->username = $request->username;
    $data->email = $request->email;
    $data->phone = $request->phone;
    $data->address = $request->address;


    if ($request->file('photo')) {
        $file = $request->file('photo');
        @unlink(public_path('upload/user_images/'.$data->photo));
        $filename = date('YmdHi').$file->getClientOriginalName();
        $file->move(public_path('upload/user_images'),$filename);
        $data['photo'] = $filename;
    }

    $data->save();

    $notification = array(
        'message' => 'User Profile Updated Successfully',
        'alert-type' => 'success'
    );

    return redirect()->back()->with($notification);

} // End Mehtod
```

Figure 4.6: Change Account Details Source Code

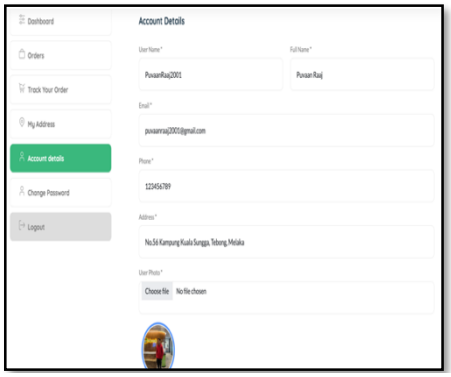


Figure 4.7: User Change Account Details Interface

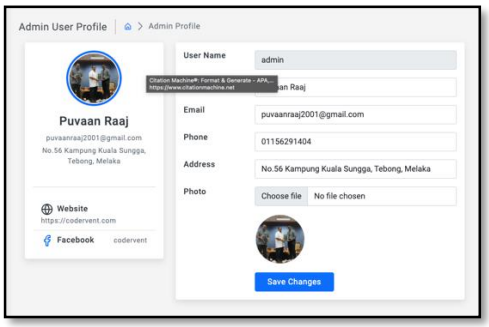


Figure 4.8: Admin Change Details Page

4.4. Manage Livestock Stock Module

By utilising Laravel's ORM for database interactions, the given PHP script in Laravel controls products, providing CRUD operations and image handling (Figures 5.9–5.14). Fetching all products using the AllProduct method, adding new products using the AddProduct method, editing and updating products using the EditProduct and UpdateProduct methods, changing product images using the UpdateProductThumbnail and UpdateProductMultiimage methods, and managing product activity status and deletion using the ProductInactive, ProductActive, and ProductDelete methods are some of the more specific methods.

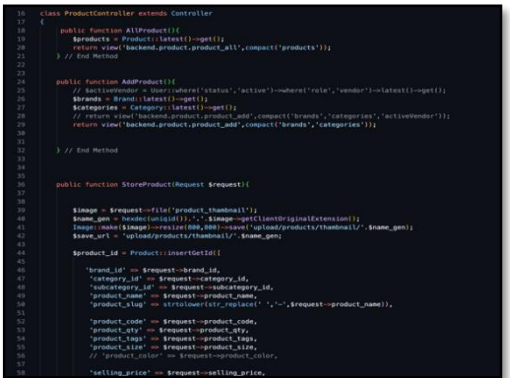


Figure 4.9: Product View, Store And Add Source Code

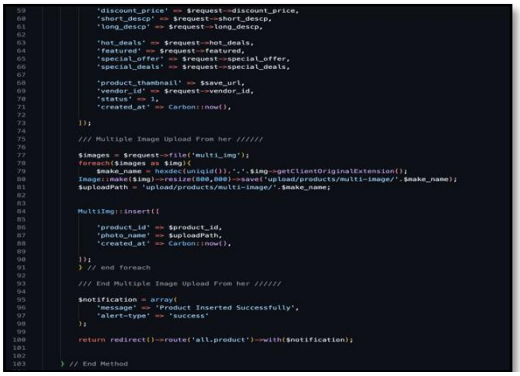


Figure 4.10: Store Product Source Code

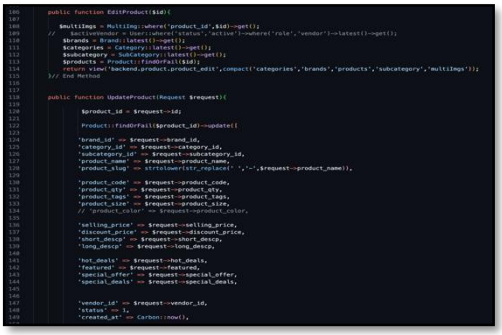


Figure 4.11: Edit Product And Update Product Source Code

Figure 4.12: Update Product Thumbnail Source Code

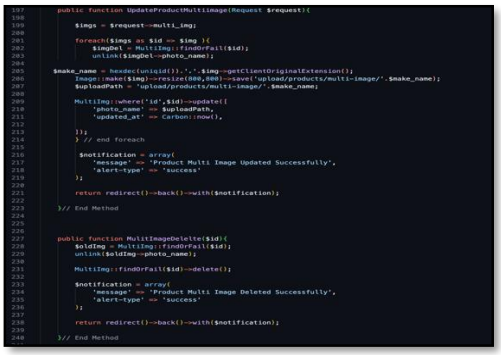


Figure 4.13: Update Product Multiimage, Product Active, And Product Inactive Source

Figure 4.14: Product Delete And Product Stock Source Code

Every product is listed in the interfaces (Figures 5.15–5.17), which also offer opportunities to add, edit, and delete them as well as to change parameters like name, tags, price, photos, and other details.

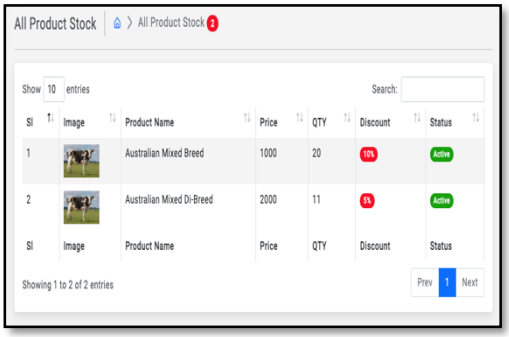


Figure 4.15: All Product List Interface

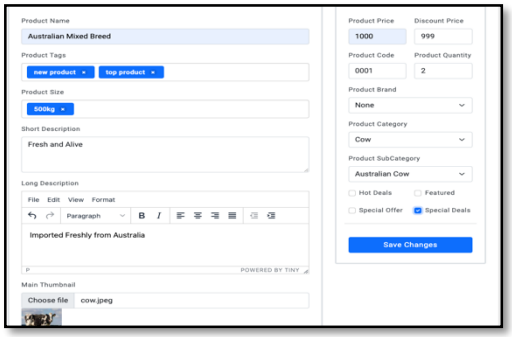


Figure 4.16: Add New Product Interface

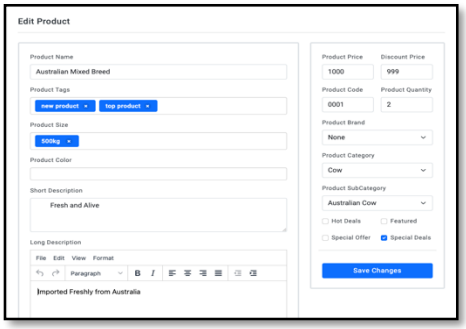


Figure 4.17: Edit Product Interface

4.5. Manage Order Module

Laravel's OrderController class controls the processing and display of orders at various stages (Figures 5.18–5.19). Orders are retrieved and managed based on their status using methods like PendingOrder(), AdminOrderDetails($order_id), AdminConfirmedOrder(), AdminProcessingOrder(), and AdminDeliveredOrder(). The status of orders and the number of products are updated by other methods like PendingToConfirm($order_id), ConfirmToProcess($order_id), and ProcessToDelivered($order_id). An invoice in PDF format is created via the AdminInvoiceDownload($order_id) function.

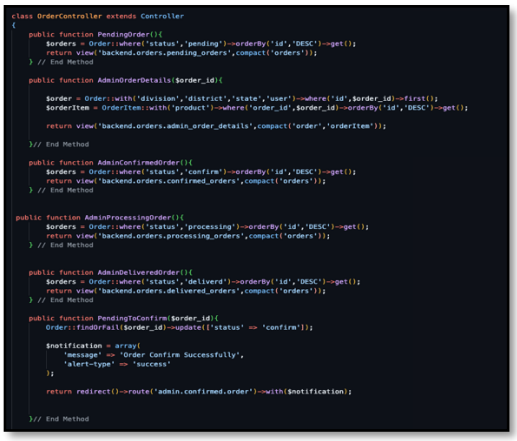


Figure 4.18: Order Controller Source Code

Figure 4.19: Continuation Of Order Controller Source Code

Lists of orders at various levels are displayed in the interfaces (Figures 5.20–5.24), together with choices for examining order details, altering order status, and managing checkout and payment information.

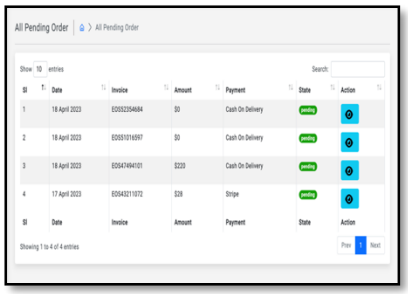


Figure 4.20: All Pending Order Interface

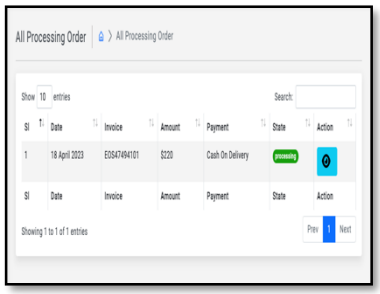


Figure 4.21: All Confirmed Order Interface

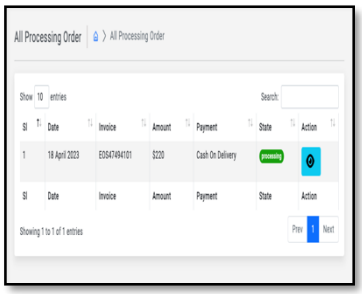


Figure 4.22: All Processing Order Interface

Figure 4.23: All Delivered Order Interface

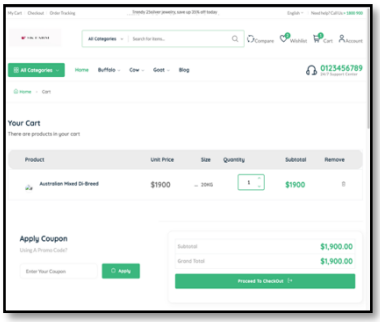


Figure 4.24: Checkout Order Interface

4.6. Manage Employee Details

In Laravel, employee data and shifts are managed via the EmployeeController and EmployeeShiftController, respectively (Figures 5.25–5.27). These controllers offer methods for CRUD operations as well as methods to handle employee status and records, such as EmployeeInactive, EmployeeActive, and EmployeeDelete. Similar options are provided by the EmployeeShiftController for viewing, creating, updating, and deleting shift records.

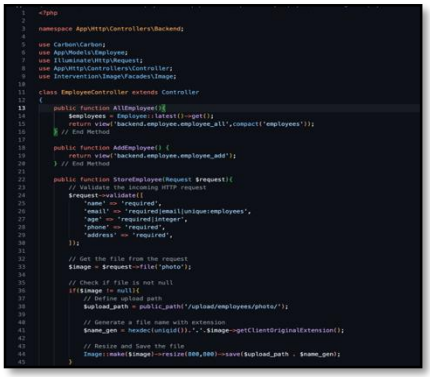


Figure 4.25: Employee Controller Source Code

Figure 4.26: Employee Controller Source Code Continuation

Figure 4.27: Employee Shift Controller Source Code

Employee and shift data can be managed using the interfaces shown in Figures 5.28 to 5.33. A list of every employee is shown in Figure 5.28, along with buttons to modify, remove, or change their status. The forms in Figures 5.29 and 5.30 are for adding and editing employee records. A table with each employee's shift information and edit and remove buttons may be found in Figure 5.31. Finally, interfaces to add and modify employee shift times are provided in Figures 5.32 and 5.33.

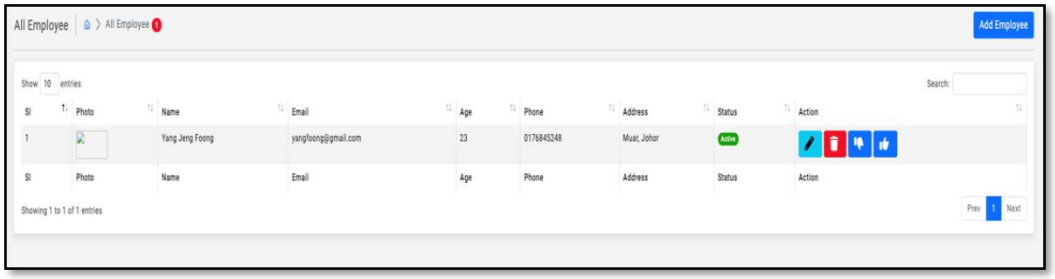


Figure 4.28: All Employee List Interface

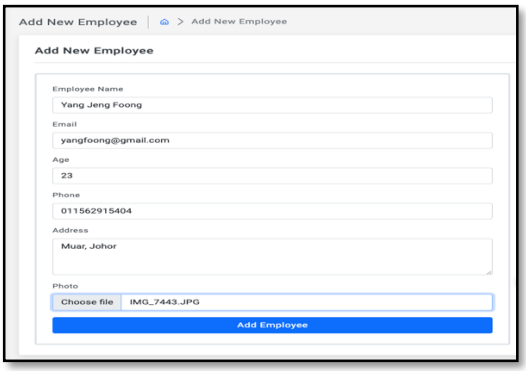


Figure 4.29: Add New Employee

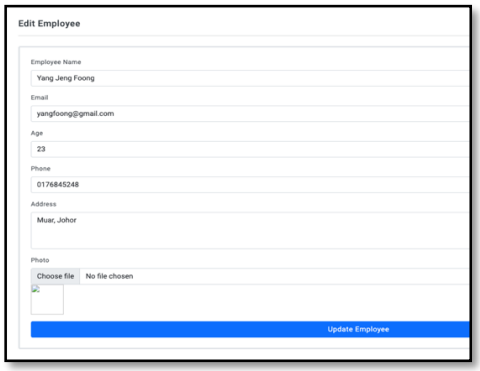


Figure 4.30: Edit Employee Interface

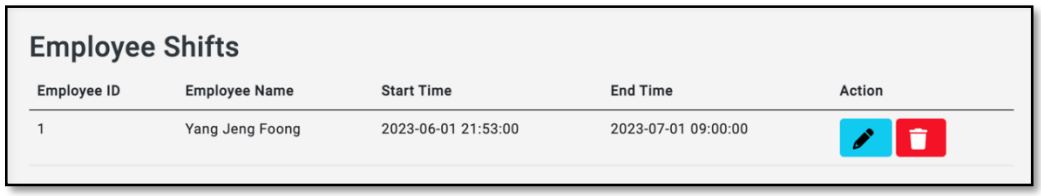


Figure 4.31: All Employee Shift Interface

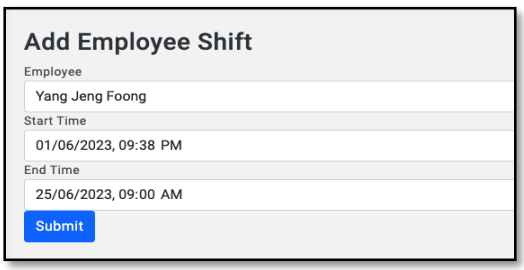


Figure 4.32: Add Employee Shift

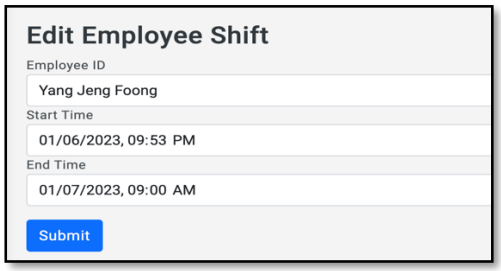


Figure 4.33: Edit Employee Shift

4.7. Manage Financial Information

The many expense-related processes in a Laravel application are handled by the ExpenseController depicted in Figures 5.34 and 5.35. To display on the total_expenses view, the viewExpenses function retrieves all expense data from the Expense model. addStoreExpense generates a new expense record in the database, whereas expense serves as the view for adding new expenses. For editing and updating, editExpense retrieves a specific cost record. Spending changes that record. Using deleteExpense, you can remove a specific expense record from the database.

Figure 4.34: Expense controller source code

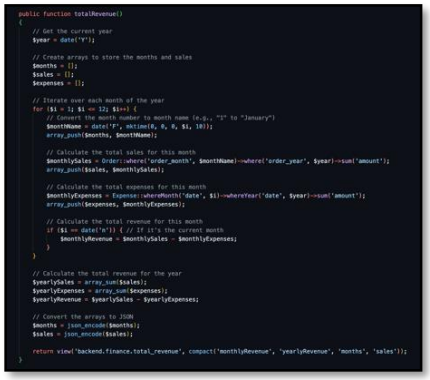


Figure 4.35: Total Revenue Source Code

In Figure 5.35, the totalRevenue function iterates through each month and retrieves sales and expense data in order to generate the total revenue data for a year. It computes monthly profit, yearly sales totals, and expenses, then gets the data ready for a view. There are ways to interact with expense data using the interfaces depicted in Figures 5.36 to 5.40. Figure 5.36 lists all expenses and offers the ability to alter or remove particular data. Forms to alter and add additional expenses are shown in Figures 5.37 and 5.38. Total sales are shown in Figure 5.39, and total revenue data is shown in Figure 5.40 in a graphical manner using Chart.js, offering an aesthetically pleasing and interactive way to view monthly and annual revenue data.

Figure 4.36: All Expenses Interface

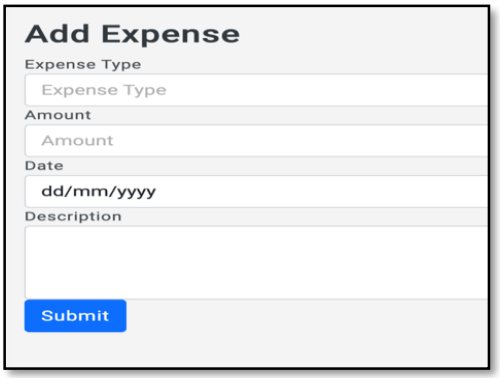


Figure 4.37: Add Expense Interface

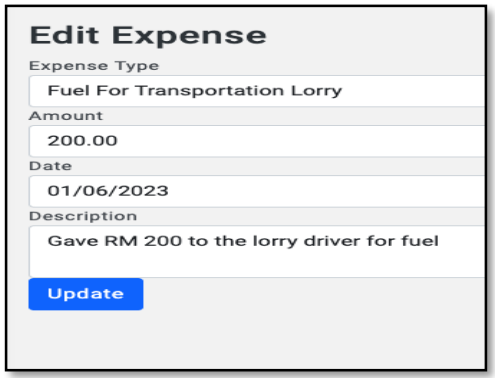


Figure 4.38: Edit Expense Interface

Figure 4.39: Total Sales Interface

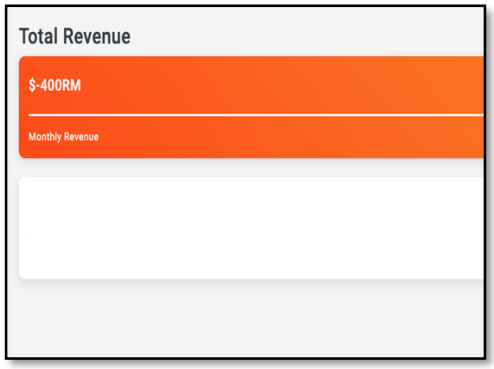


Figure 4.40: Total Revenue Interface

4.8. Make Payment

By calculating the entire order amount, producing a matching Stripe charge, and entering the order in the database with the necessary information, the StripeOrder function, shown in Figure 5.41, controls the Stripe payment process. A secure Stripe Payment Interface is shown in Figure 5.42, which makes use of the Stripe.js framework to process credit card information securely. Figure 5.43 shows the Cash Payment Interface, which offers a "Cash On Delivery" option in which payment is made in cash when the goods is delivered, giving clients yet another level of payment flexibility.

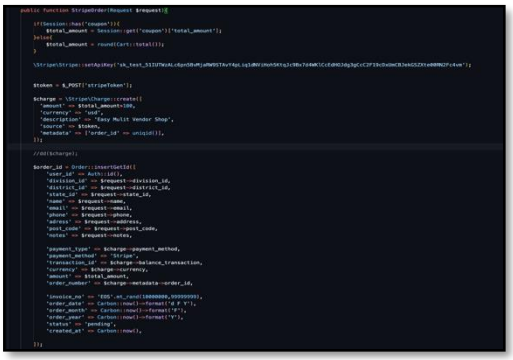


Figure 4.41: Stripe controller source code

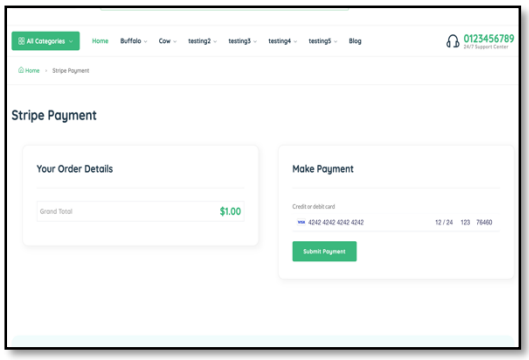


Figure 4.42: Stripe Payment Interface

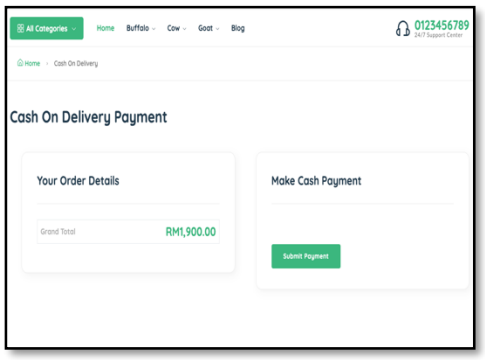


Figure 4.43: Cash payment interface

5. Testing

The system testing for the SK FARM Livestock Management System involved a total of 46 test cases across 8 modules. The tests covered user registration and login, managing livestock stock, financial information, employee details, and orders, and facilitating payments. All but one test case, TC_200_04 (associated with displaying an error message for incorrect login credentials), passed, indicating that the system is largely functioning as intended. The failed test case will be investigated and fixed for a complete pass. The test results are summarized in Table 5.

Table 5: Test result of test cases

Test Cases	Total Test Cases	Total Success	Total Failed
TC_100	4	4	-
TC_200	5	4	1
TC_300	6	6	-
TC_400	7	7	-
TC_500	5	6	-
TC_600	6	6	-
TC_700	7	6	-
TC_800	6	6	-
Total	46	45	1

6. Conclusion

The development and implementation of the SK Farm Livestock Management System has brought considerable benefits to the farm owner and customers. It minimizes human handling concerns, lowers data redundancy, and improves the accuracy of financial data by automating the administration of stock and orders. By making the process of placing orders, making payments, and creating accounts simple, it also enhances the consumer experience. Advanced analytics, reporting tools, and mobile platform connectivity are all missing from the solution. These constraints allow room for future improvements, such adding analytics and utilizing mobile platforms, which might optimize agricultural operations, provide insightful business data, and increase user accessibility and convenience. In conclusion, the system effectively handles a variety of issues that the farm owner faces, which helps to improve the effectiveness and efficiency of the farm management process.

Acknowledgment

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

References

[1] Thornton, P. K. (2010). Livestock production: Recent trends, future prospects. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *365*(1554), 2853–2867. https://doi.org/10.1098/rstb.2010.0134.

[2] "What is a management system?: Benefits," *Benefits | BSI Australia*. [Online]. Available: https://www.bsigroup.com/en-AU/About-BSI/FAQs/What-is-a Management-System/. [Accessed: 26-Nov-2022]. "What is a management system?: Benefits,"

[3] British Standards Institution (2021). *What is a Management System?*. Retrieved on November 19, 2022, from https://www.bsigroup.com/en-AU/About- BSI/FAQs/What-is-a-Management-System/.

[4] Joo, Y. J., Kim, N. and Kim, N.H. (2016). Factors predicting online university students' use of a mobile learning management system (m-LMS). *Educational Technology Research and Development*. 64(4): 611–630.

[5] AGRIWEBB Pty. Ltd.(2000). *Agriwebb*. Retrieved on November 15 2022, from https://www.agriwebb.com/.

[6] Farmbrite (2013). *Farmbrite. Easier farm management starts here.* Retrieved on November 16 2022, from https://www.farmbrite.com/.

[7] Ranch Manager (2019). *Ranch Manager. Livestock Management Software.* Retrieved on November 17 2022, from http://www.ranchmanageropen.com/index.html

[8] "All certificate & diploma courses list, certificate & diploma courses in 1 month, Online Certificate & Diploma course," *Diploma and Certificate Courses*. [Online]. Available: https://www.shikshaglobe.com/news/incremental-model-in-sdlc-shikshaglobe-1600. [Accessed: 22-Dec-2022].