[FARM MANAGEMENT SYSTEM]

[MOHAMAD HAFISZUDIN BIN AMID]

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

[FARM MANAGEMENT SYSTEM]

[MOHAMAD HAFISZUDIN BIN AMID]

This report is submitted in partial fulfillment of the requirements for the Bachelor of [Computer Science (Database Management)] with Honours.

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2024

DECLARATION

I hereby declare that this project report entitled

[FARM MANAGEMENT SYSTEM]

is written by me and is my own effort and that no part has been plagiarized without citations.

STUDENT :  Date : 27/8/2024

([MOHAMAD HAFISZUDIN BIN AMID])

I hereby declare that I have read this project report and found this project report is sufficient in term of the scope and quality for the award of Bachelor of [Computer Science (Database Management)] with Honours.

SUPERVISOR :  Date : 6/9/2024

([NOOR AZILAH BINTI DRAMAN@MUDA])

DEDICATION

To my family, whose unwavering support and encouragement have been my guiding light throughout this journey. To my mentors and teachers, who have imparted wisdom and inspired me to strive for excellence. And to all my friends, whose constant belief in my abilities has kept me motivated and resilient.

ACKNOWLEDGEMENTS

I owe a debt of gratitude to my supervisor, Noor Azilah Binti Draman @ Muda, for all of her help and advice on this project. Her perceptive criticism and devoted assistance have been crucial toward its successful conclusion.

My evaluator, Dr. Norashikin Binti Ahmad, has deepest appreciation for her thorough assessment and insightful suggestions. I sincerely thank my family for their constant encouragement and support during my academic career.

Finally, I would want to express my sincere gratitude to all of my lecturers, seniors, and other mentors who have helped me both theoretically and practically.

ABSTRACT

This project focuses on developing a comprehensive Farm Management System to enhance livestock efficiency and productivity. The system addresses key aspects of farm management, including sales tracking, expense management, and financial reporting, providing farmers with a robust tool to monitor and optimize their farm's financial performance.

The Farm Management System features an intuitive, user-friendly interface, ensuring accessibility for farmers with varying technical expertise. It incorporates data analytics capabilities, informed decision-making based on current data and trends. This functionality is crucial for optimizing resource allocation, managing inventory, and planning future farming activities.

By automating routine tasks and offering comprehensive financial insights, the system aims to reduce administrative burdens, allowing farmers to focus on core farming activities. The project's primary objectives include increasing profitability, improving resource utilization, and enhancing the sustainability of farming operations.

A case study evaluating the system's implementation demonstrates its practical application and effectiveness in a real-world farming context. The results underscore the system's ability to streamline operations, provide valuable financial insights, and support decision-making processes. Overall, this project advances livestock technology and offers a solution for modernizing farm management practices.

ABSTRAK

Projek ini memfokuskan kepada pembangunan Sistem Pengurusan Ladang yang menyeluruh untuk meningkatkan kecekapan dan produktiviti penternakan di Malaysia. Sistem ini merangkumi aspek-aspek utama pengurusan ladang seperti penjejakan jualan, pengurusan perbelanjaan, dan pelaporan kewangan, memberikan alat yang kukuh kepada petani untuk memantau dan mengoptimumkan prestasi kewangan ladang mereka.

Sistem Pengurusan Ladang ini mempunyai antara muka yang intuitif dan mesra pengguna, memastikan aksesibiliti untuk petani dengan tahap kepakaran teknikal yang berbeza-beza. Ia juga menggabungkan keupayaan analitik data, membolehkan pembuatan keputusan yang dimaklumkan berdasarkan data dan tren semasa. Fungsi ini amat penting untuk mengoptimumkan pengagihan sumber, menguruskan inventori, dan merancang aktiviti pertanian masa hadapan.

Dengan mengautomasikan tugas-tugas rutin dan menawarkan wawasan kewangan yang komprehensif, sistem ini bertujuan untuk mengurangkan beban pentadbiran, membolehkan petani untuk memberi tumpuan kepada aktiviti teras pertanian. Objektif utama projek ini termasuk meningkatkan keuntungan, memperbaiki penggunaan sumber, dan meningkatkan kelestarian operasi pertanian.

Kajian kes yang menilai pelaksanaan sistem ini menunjukkan aplikasi praktikal dan keberkesanannya dalam konteks ladang sebenar. Keputusan kajian menekankan keupayaan sistem untuk menyelaraskan operasi, menyediakan wawasan kewangan yang berharga, dan menyokong proses pembuatan keputusan. Secara keseluruhannya, projek ini memajukan teknologi penternakan di Malaysia dan menawarkan penyelesaian untuk memodenkan amalan pengurusan ladang.

TABLE OF CONTENTS

LIST OF TABLE

LIST OF FIGURE

CHAPTER 1 : INTRODUCTION

1.1 INTRODUCTION

This project, the Farm Management System (FMS), is important because it helps farmers do a better job in their work. Traditional farming ways sometimes have problems with mistakes in keeping records and using resources like water and food resource.

Farm Management System wants to fix this by using technology to make things easier for farmers. It can automatically handle farm information, like what livestock are being preserved or when to give them food.

This way, it reduces mistakes and saves time. Farm Management System also helps farmers use the right amount of food resource and other things, saving money and being better for the environment.

Overall, this project is valuable as it makes farming smarter, more efficient, and better for everyone.

1.2 PROBLEM STATEMENT

Hard and Disorganized Data Handling: Farmers often write things down on paper, which can cause mistakes. There's no one place where all the information is kept, making it tough to manage and understand.

Not Good at Using Resources: Farmers might use too much water, food resource, or medicine and vitamin because they don't have good ways to keep track their routine. We need a way to watch these things closely to save money and not harm the environment.

Can't Make Good Decisions: Farmers find it hard to make good choices because they don't have the right information at the right time. This can affect how well the livestock to grow, how healthy animals are, and how much money the farm makes.

1.3 OBJECTIVE

i). Streamline Data Management: Develop a centralized platform to automate data collection and management, reducing errors and providing a unified database for comprehensive livestock information.

ii. Enhance Resource Efficiency: Implement tools for precise monitoring and control of resources, including vaccine scheduling, food and water control, to optimize resource usage and reduce waste.

iii. Empower Decision-Making: Provide farmers with data analytics and decision support tools, enabling them to make informed choices related to livestock care, and financial planning.

1.4 SCOPE

The scope involved in the Farm Management System in this system is divided into two parts, which are involvement of user and module types which is:

1.4.1 Target user

Admin

Worker

1.4.2 Modules

i). User Module: Manages user profiles, roles, and permissions to ensure secure and personalized access to the system.

ii). Shed Management Module: Oversees the maintenance, organization, and inventory of farm sheds, ensuring optimal storage and conditions.

iii). Livestock Management Module: Tracks and manages the health, feeding, and movement of livestock to ensure their well-being and productivity.

iv). Sale Management Module: Handles the recording, tracking, and reporting of all sales transactions, providing insights into revenue and market trends.

v). Customer Management Module: Manages customer information, interactions, and relationships to enhance satisfaction and foster loyalty.

vi). Financial Management Module: Oversees budgeting, accounting, and financial reporting to ensure accurate and efficient financial operations.

1.5 PROJECT SIGNIFICANT

The Farm Management System (FMS) provide significant benefit by revolutionize livestock practices by streamlining data management, enhancing resource efficiency, and empowering decision-making. By developing a centralized platform, the Farm Management System automates data collection and management, minimizing errors and offering a unified database for comprehensive livestock information. The system implements precise monitoring and control tools for resources, including vaccine scheduling and food and water management, optimizing usage and reducing waste. Furthermore, the Farm Management System equips farmers with advanced data analytics and decision support tools, enabling informed choices related to livestock care and financial planning, ultimately improving productivity and sustainability.

1.6 EXPECTED OUTPUT

The Farm Management System (FMS) expected output is to enhances livestock operations by simplifying data handling, optimizing resource usage, and helping decision-making. By integrating an open-source DBMS, the Farm Management System efficiently tracks and manages information, eliminating the need for manual record-keeping and ensuring easy access to performance data. The system includes tools to monitor and control essential resources such as water, food, and health medicines, promoting better utilization and reducing waste. Additionally, advanced data analytics within the Farm Management System support informed decision-making, enabling farmers to make timely and effective choices to improve productivity and sustainability.

1.7 CONCLUSION

In conclusion, the Farm Management System represents a transformative advancement in livestock operations, providing a centralized platform that streamlines data management, enhances resource efficiency, and empowers farmers with vital decision-making tools. By automating data collection and integrating advanced monitoring systems, the Farm Management System ensures accurate and comprehensive livestock information while optimizing the use of resources such as vaccines, food, and water. Additionally, the incorporation of data analytics and decision support tools enables farmers to make informed choices, ultimately leading to improved productivity, sustainability, and overall farm management. This innovative system not only reduces manual labor and errors but also fosters a more efficient and effective agricultural environment.

CHAPTER 2 : PROJECT METHODOLOGY AND PLANNING

2.1 INTRODUCTION

This chapter will give an overview of the planning and technique used in the "Farm Management System". The project's database life cycle (DBLC) phases are established, along with the particular tasks that correspond with each phase, the project timetable, and milestones. The goal of the structured approach is to provide a reliable and user-friendly Farm Management System by ensuring methodical development and execution.

2.2 PROJECT METHODOLOGY

The Database Life Cycle (DBLC) phase is followed by the " Farm Management System " project, guaranteeing a methodical approach to database development and implementation. A database system's planning, development, deployment, and maintenance can be approached methodically with the help of the phases that make up the Database Life Cycle (DBLC).

Phases of the DBLC that apply to this Farm Management System project are:

i). Database Initial Study

The initial study in the Farm Management System (FMS) project follows the System Development Life Cycle (SDLC) methodology, starting with the planning phase where objectives such as streamlining data management, enhancing resource efficiency, and empowering decision-making are defined, and the scope of modules (User, Shed Management, Livestock Management, Sale Management, Customer Management, Financial Management) is outlined. A feasibility study evaluates the technical, economic, and operational aspects, including the use of open-source DBMS and cloud databases. In the requirements analysis phase, detailed functional and non-functional requirements are

gathered through stakeholder interviews, focusing on features like data automation, resource monitoring, and analytics tools. The system design phase then translates these requirements into a comprehensive blueprint for the FMS, ensuring all specifications are met to create an effective and efficient farm management solution.

ii). Database Design

The database design for the Farm Management System (FMS) involves creating a scalable schema to organize data for User, Shed Management, Livestock Management, Sale Management, Customer Management, and Financial Management modules and creating ERD. It includes relational tables for user profiles, shed inventory, livestock records, sales transactions, customer interactions, and financial data. Relationships between tables ensure data integrity and support complex queries. Open-source DBMS are used for flexibility, scalability, and secure access, enabling comprehensive insights for effective farm management.

iii) Implementation and Loading

In the implementation and loading phase, Installing and setting up the selected database management system (DBMS), such as XAMPP, on the server for optimum performance. In order to implement the logical schema, tables, indexes, and constraints must be created. Additionally, stored procedures, functions, and triggers for business logic must be written. To import, extract, and modify data from current systems into the new one, scripts are created. To guarantee a seamless transition, data completeness and integrity are checked after loading. Establishing the database, loading or converting data, and configuring the DBMS are the main activities.

iv). Testing and Evaluation

To make sure the database satisfies all functional and performance criteria, the testing and assessment step is essential. Unit testing is used to confirm the operation of individual components, integration testing is used to make sure various system components operate together flawlessly. Testing scenarios for the Farm Management System to make sure any defects or problems found during testing will be noted and fixed. To make sure the system can manage the anticipated load, performance testing will

also be carried out. Delivering a dependable database system that satisfies all needs is the aim.

v). Operation

In operation phase, the performance will be monito using the monitoring tools to track any problem that will be happen when the system will be running so that any problem will be noted and make a further solving.

vi). Maintenance and Evaluation

In this final phase which is maintenance and evaluation, this will make sure the error that come and any error in will be gather and will fix and the feedback form the user will be gather and will be the source for any improvement that needs to do to our system.

2.3 PROJECT SCHEDULE AND MILESTONE

Table 2.1: Project Schedule

Milestones	Expected Document	Start Dates	End Date
Identify and analyze the requirements of the Farm Management System. Problem identification and analysis. Define the scope and objectives of system	Project Proposal	11 March 2024	22 March 2024
i). Conceptual design of proposed system ii). Develop a logical design to map entities to relational tables.	i) ERD ii) Context Diagram iii) DFD	22 March 2024	1 April 2024
i). Install and setup DBMS.	i). XAMMP installation.	1 April 2024	5 April 2024

ii). Create the database.	ii) Database schema		
System Development	i) System Interface ii) Front end iii) Back end	6 April 2024	10 Jun 2024
System and Database testing	Test Plan and Result	10 Jun 2024	14 Jun 2024
Project demonstration	i) Present a demonstration of the system to the supervisor. ii) Make improvement to the systems.	20 June 2024	20 June 2024
Report Submission	Report Submit	20 June 2024	20 June 2024

Table 2.2: Gantt Chart Farm Management System

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Requirement analysis															
Design of the proposed system															
Implementation of system															
System testing															
Project demo															
PSM report															
Final presentation															

2.4 CONCLUSION

In conclusion, the Farm Management System, project meticulously follows the Database Life Cycle (DBLC) methodology to ensure a reliable and easy to understand solution. Starting with the initial study and requirements analysis, the project outlines clear objectives and evaluates feasibility. The database design phase establishes a scalable schema, while the implementation and loading phase focuses on setting up the DBMS and migrating data. Rigorous testing and evaluation ensure the system meets all functional and performance criteria. Once operational, the system's performance is continuously monitored, and maintenance ensures ongoing improvements and error resolution. This structured approach guarantees a robust and effective farm management solution tailored to the needs of its users in Farm Management System.

CHAPTER 3 : ANALYSIS

3.1 INTRODUCTION

In this chapter, we will proceed with the analysis of the current Farm Management System in Malaysia like mylembu which is not very details in functionality and only provide the cow buying not with the management like system for farmer to manage their cow. System analysis is a systematic process that involves information gathering, workflow modeling, data analysis, and system requirement definition. This analysis aids in the creation, implementation, or improvement of a system to effectively fulfill the goals it set out to. The current system will be improve using suitable function and quality data.

3.2 PROBLEM ANALYSIS

The current farm management system in Malaysia, exemplified by platforms like Mylembu, exhibits several limitations that hinder its effectiveness and utility for farmers. The primary issue with Mylembu is its lack of comprehensive functionality. It primarily focuses on facilitating cow purchases without providing robust management tools for farmers to oversee the various aspects of livestock care and farm operations. This limited scope means that crucial activities such as health monitoring, feeding schedules, breeding records, and financial management are not adequately supported, leaving farmers to manage these critical tasks through less efficient means.

3.3 THE PROPOSE IMPROVEMENT/SOLUTION

To improve the current farm management system in Malaysia, platforms like Mylembu should expand beyond facilitating cow purchases to offer comprehensive livestock management tools. This includes developing a centralized database for detailed

record-keeping, integrating workflow automation to streamline farm operations, and incorporating advanced data analytics for actionable insights. A user-friendly interface and accessibility will ensure farmers can easily manage health records, and financial transactions. These enhancements will create a more efficient, productive, and profitable farm management system, addressing the current system's limitations and better supporting farmers in their daily operations.

3.4 REQUIREMENT ANALYSIS OF TO-BE SYSTEM

3.4.1 Functional Requirement (Process Model)

Functional requirements specify the precise functions and behaviors that a system must possess, with a focus on the transfer, storing, and processing of data.

1. Worker
    - Access worker information.
    - Access the vaccination appointment.
    - Access the customer details.
    - Handle management of shed in farm.
    - Handle management of livestock in farm system.
    - Handle management of livestock meat and milk collect.
    - Handle management sale for the meat and milk sale.
    - Handle management of expense.

2. Admin
    - Access all worker function.
    - Review the financial report.
    - Review the expense total.
    - Review the payment status of customer in total.

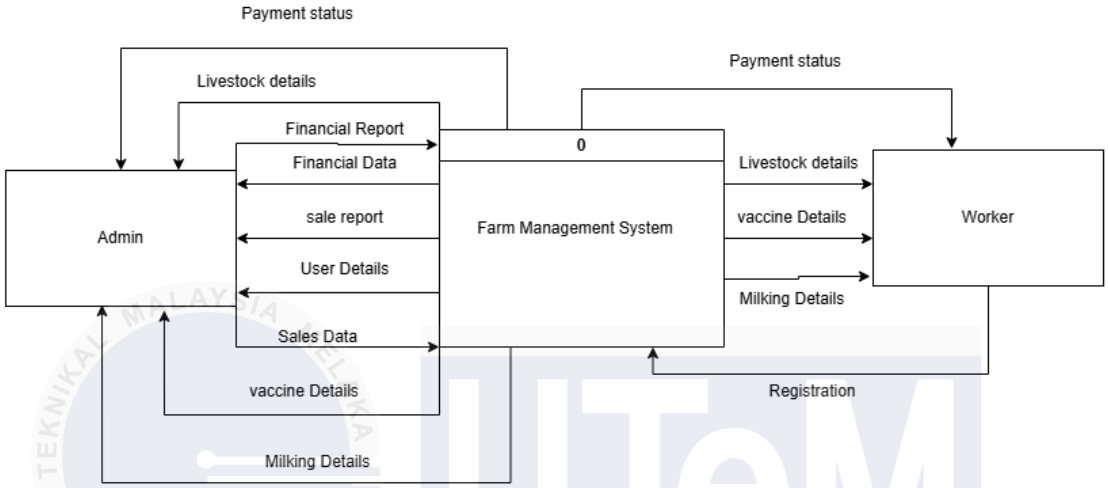

Figure 3.1 Context Diagram of Farm Management System

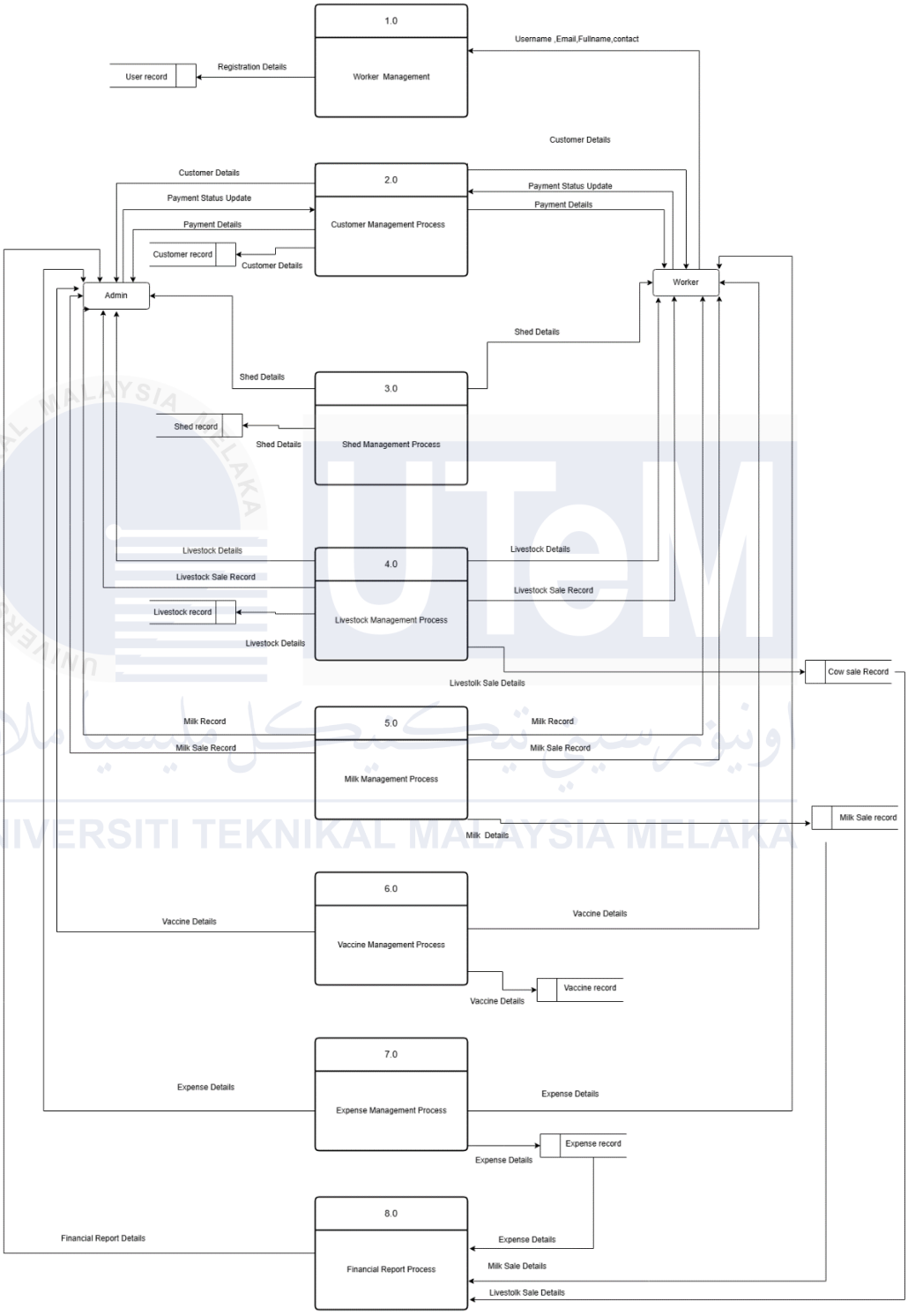

Figure 3.2 Data Flow Diagram Level 0 of Farm Management System

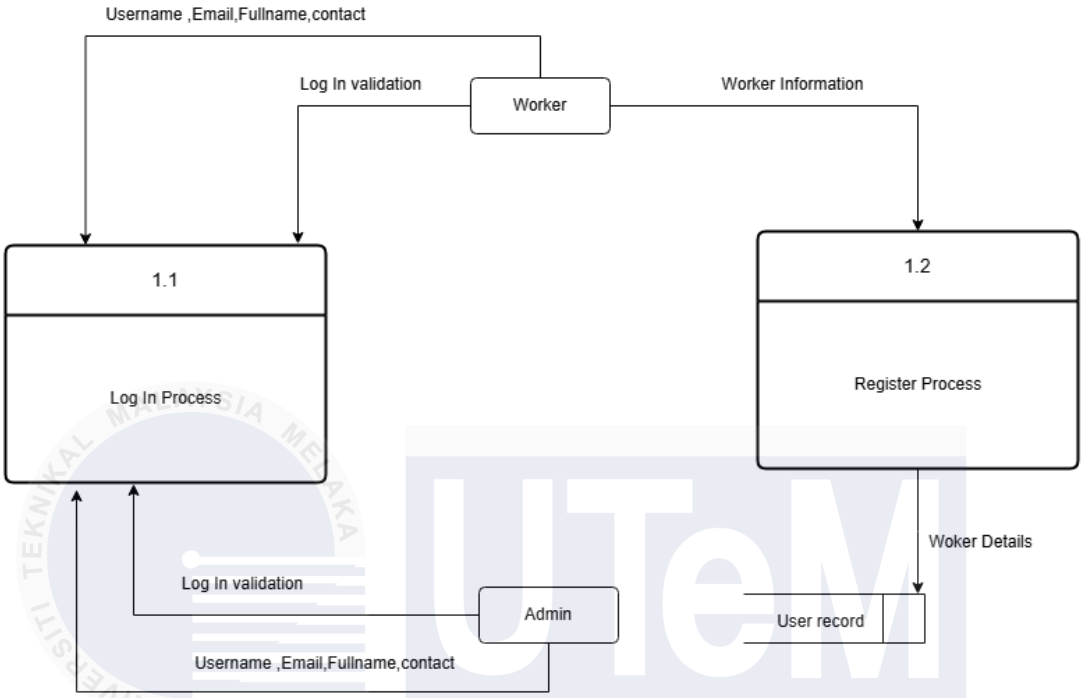

Figure 3.3 Worker Management DFD Level 1 of Farm Management System

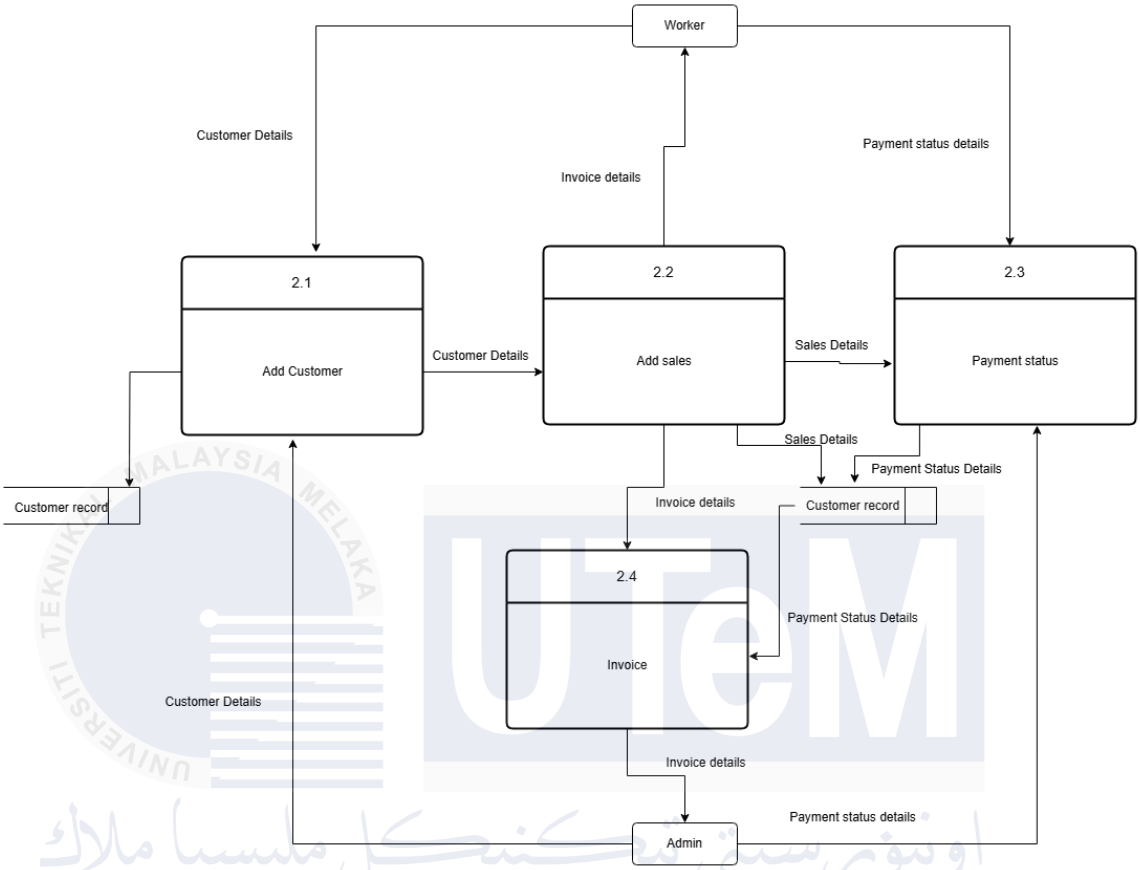

Figure 3.4 Customer Management DFD Level 1 of Farm Management System

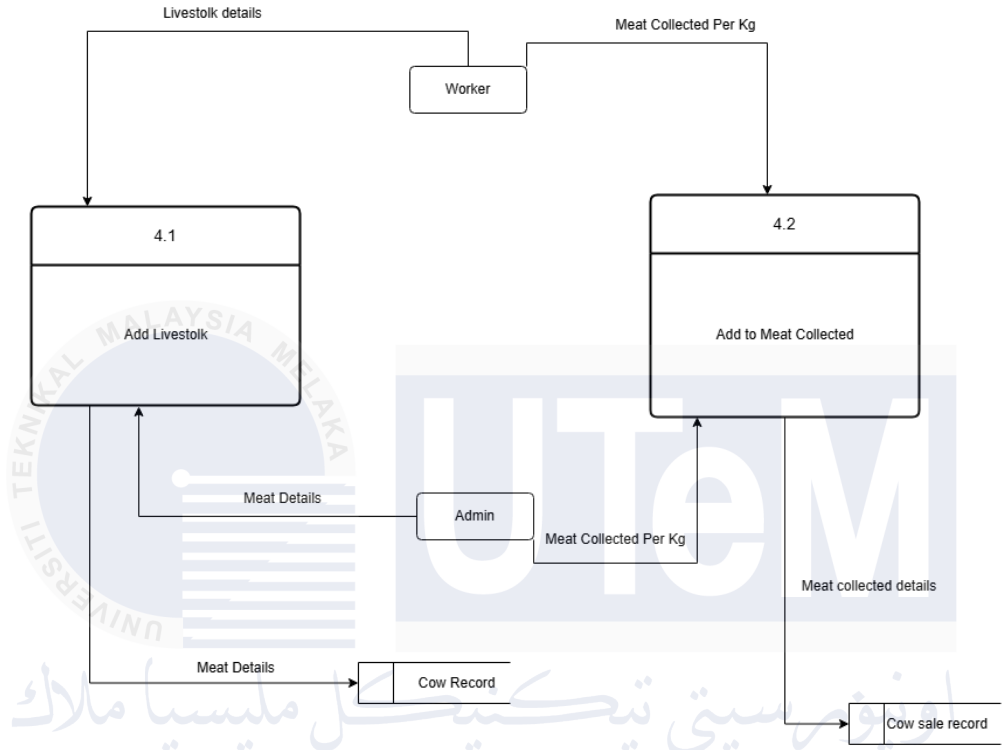

Figure 3.5 Livestock Management DFD Level 1 of Farm Management System

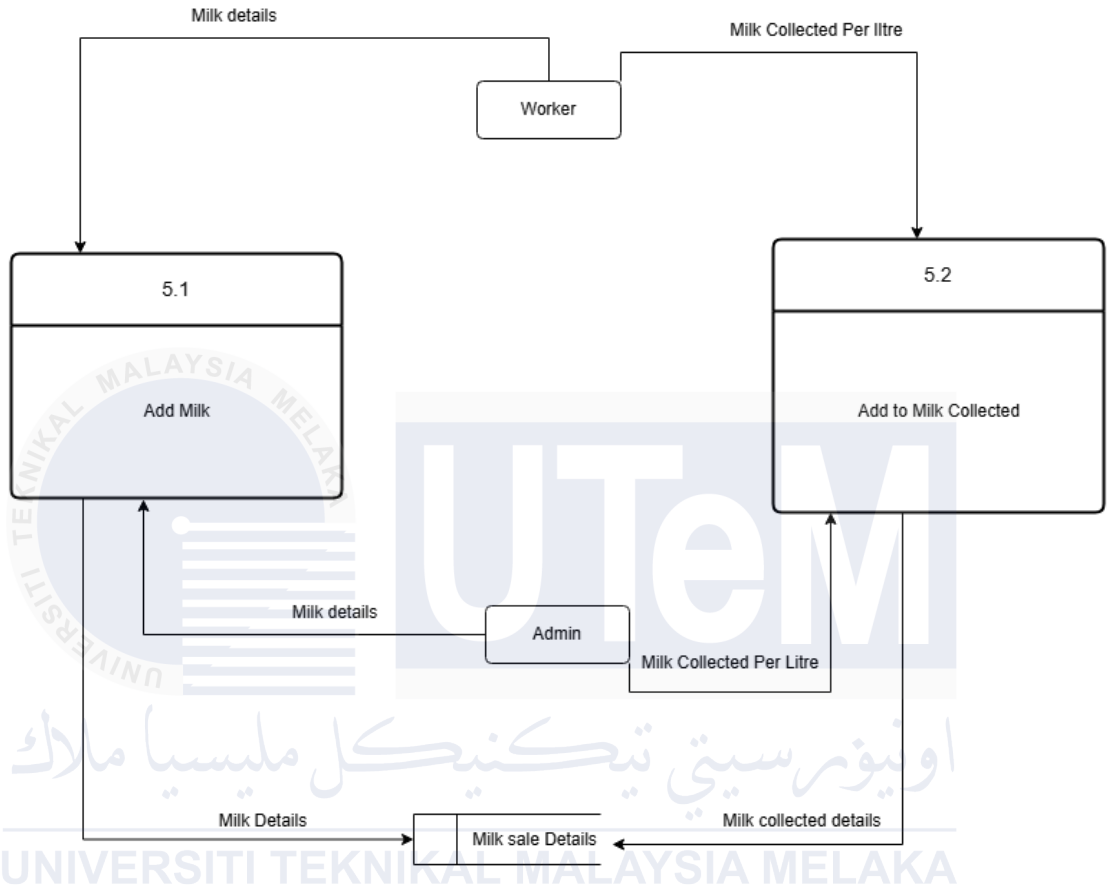

Figure 3.5 Milk Management DFD Level 1 of Farm Management System

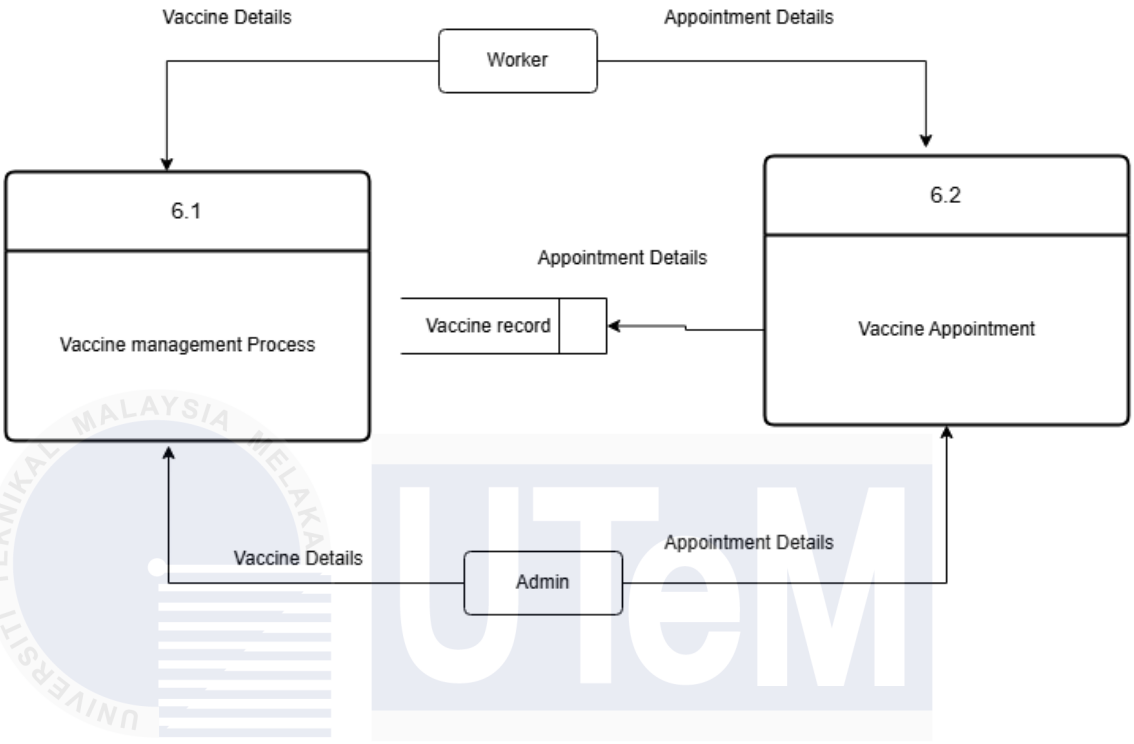

Figure 3.6 Vaccine Management DFD Level 1 of Farm Management System

3.4.2 Non Functional Requirement

a) Usability

The system interface should be intuitive and easy to use, requiring minimal training for farmers.

b) Scalability:

The system should be scalable to support the addition of new modules and increased data volume without requiring significant changes to the existing infrastructure.

c) Performance:

Data retrieval and processing times should not exceed 2 seconds for standard queries.

3.4.3 Other Requirement

3.4.3.1 SOFTWARE REQUIREMENT

The "Farm Management System" requires a set of software tools to support its development, deployment, and operation which is :

a) Sublime

Highly versatile and lightweight text editor known for its speed, simplicity, and powerful features like syntax highlighting, code snippets, and a customizable interface, making it popular among developers and programmers.

b) PHP

Widely-used open-source scripting language especially suited for web development, enabling the creation of dynamic and interactive web pages.

c) Draw.io

User-friendly online diagramming tool that allows users to create, edit, and share a wide variety of diagrams, including flowcharts, UML diagrams, and network diagrams, with robust collaboration features and integration with popular platforms like Google Drive and Confluence.

d) XAMMP

Free open-source cross-platform web server solution stack package developed by Apache Friends, which includes Apache HTTP Server, MariaDB, and interpreters for scripts written in PHP and Perl, making it easy to set up a local server for web development and testing.

e) Microsoft Word

Widely-used word processing software developed by Microsoft, offering a comprehensive suite of features for creating, editing, and formatting text documents, including templates, collaboration tools, and integration with other Microsoft Office applications.

f) Canva

Versatile graphic design platform that allows users to create a wide array of visual content, including social media graphics, presentations, posters, and more, through an intuitive drag-and-drop interface and a vast library of templates and design elements.

3.4.3.2 HARDWARE REQUIREMENT

- **Desktop**

1. AMD Ryzen 5 5600G with Radeon Graphics @ 3.90 GHz
2. Memory : 16 GB RAM
3. Storage: 500 GB SSD
4. Graphics: NVIDIA GeForce RTX 3060 Ti
5. Display : LG Desktop

3.5 CONCLUSION

In conclusion, the Farm Management System is designed to enhance farm operations by addressing both functional and non-functional requirements. Functionally, it provides detailed capabilities for workers and administrators, including managing livestock, handling sales, and reviewing financial reports. Non-functional requirements ensure the system is user-friendly, scalable, and performs efficiently. The development and deployment of the FMS rely on various software tools such as Sublime, PHP, draw.io, XAMPP, Microsoft Word, and Canva. Additionally, robust hardware specifications, including a high-performance desktop, are necessary to support the system's operations. This comprehensive approach ensures a reliable, efficient, and effective management solution for farms.

CHAPTER 4 : DESIGN

4.1 INTRODUCTION

In this chapter given it provides the important for all other phases in the system development process, designing the Farm Management System. In order to solve specific problems found, this step transforms the logical design from system analysis into a comprehensive physical system design. It includes input/output requirements, processing specifications, database definitions, and schemas. Determining the data structure, control processes, protocols, and interfaces that will serve as the system's main framework must be done during the design phase.

To design the Farm Management System, Entity-relationship diagrams (ERD) , business rules, data dictionary , data normalization, Database Management System (DBMS) and Graphical User Interface (GUI) are used.

4.2 SYSTEM ARCHITECTURE DESIGN

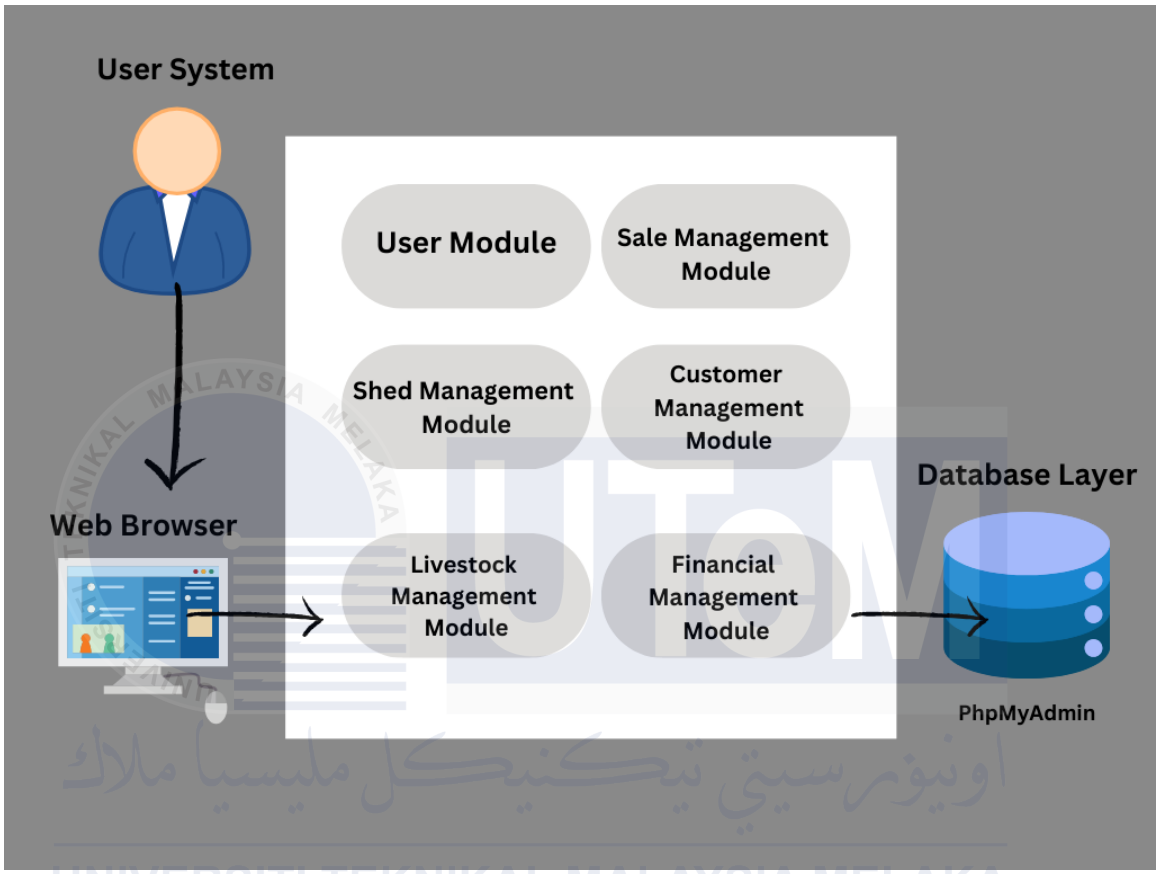

FIGURE 4.0: SYSTEM ARCHITECTURE

4.2.1 CONCEPTUAL DESIGN

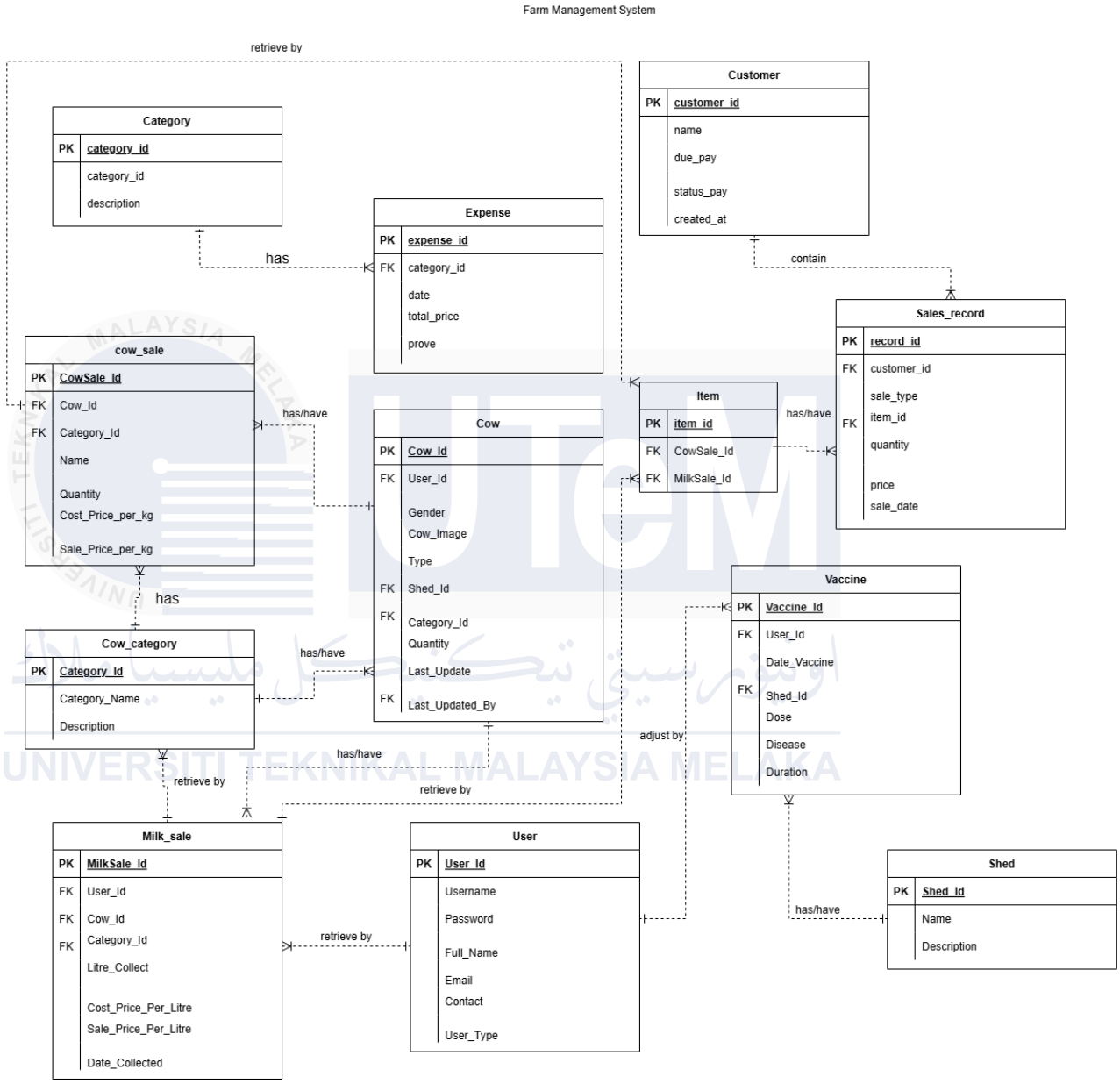

Figure 4.1: Entity Relationship Diagram (ERD)

4.2.1.1 BUSINESS RULE

Business Rules:

- **Category Management:**
  - Each category (Category) can be linked to multiple cows, cow sales, and milk sales.
  - Each expense must belong to a single category.

- **Customer Management:**
  - Each customer can have multiple sales records.

- **Expense Management:**
  - Each expense is categorized by a specific category.
  - Each expense record must include details like amount, purpose, and date.

- **Cow Management:**
  - Each cow belongs to a single category.
  - Each cow can have multiple associated cow sales and milk sales.
  - Each cow must have a unique identifier and may include details like gender, image, and type.
  - Cows are linked to specific shed locations through the shed ID.

- **Cow Sale Management:**
    - Each cow sale must reference a specific cow and a specific category.
    - Cow sales include details such as cost price per kg and sale price per kg.

- **Milk Sale Management:**
    - Each milk sale must be linked to a category and may include cost price per liter and sale price per liter details.

- **Item Management:**
    - Each item is associated with a specific cow sale or milk sale.
    - Items include details like quantity, unit, and price.

- **Sales Record Management:**
    - Each sales record must be linked to a customer.
    - Sales records include sale type, item details, quantity, price, and sale date.

- **Vaccine Management:**
    - Each vaccine record is linked to a specific user and shed.
    - Vaccine records include details like dose, disease, and duration.
    - Vaccines are administered by users.

- **User Management:**
    - Each user has unique identifiers, login credentials, and contact information.
    - Users may have different types (e.g., admin, worker).

- **Shed Management:**
    - Each shed can house multiple cows.
    - Sheds include identifiers and descriptions.

4.3.2 Logical Design

Table below shows the data dictionary of Farm Management System that it will clearly explain the data structure by listing every attribute name, data type, size, constraint, and short description.

4.3.2.1 Data Ditionary

Table 4.1: Category

Attribute Name	Data Type	Size	Format	Range	Required	Unique	PK/FK	FK Reference
category_id	int	11			Yes	Yes	PK	
name	varchar	255			Yes	No		
description	text				Yes	No		

Table 4.2: Cow

Attribute Name	Data Type	Size	Format	Range	Required	Unique	PK/ FK	FK Reference
Cow_Id	int	11			Yes	Yes	PK	
User_Id	int	11			No	No	FK	user(User_Id)
Gender	enum		('male','female')		Yes	No		
Cow_Image	varchar	255			No	No		
Type	varchar	50			No	No		
Shed_Id	int	11			No	No	FK	shed(Shed_Id)
Category_Id	int	11			No	No	FK	cow_category(Category_Id)
Quantity	int	11			No	No		
Last_Update_Timestamp	timestamp		DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP		Yes	No		
Last_Updated_By	int	11			No	No		

Table 4.3: Cow_Category

Attribute Name	Data Type	Size	Format	Range	Required	Unique	PK/FK	FK Reference
Category_Id	int	11			Yes	Yes	PK	
Category_Name	varchar	50			Yes	No		
Description	text				No	No		

Table 4.4: Cow_Sale

Attribute Name	Data Type	Size	Format	Range	Required	Unique	PK/FK	FK Reference
CowSale_Id	int	11			Yes	Yes	PK	
Cow_Id	int	11			Yes	No	FK	cow(Cow_Id)
Category_Id	int	11			Yes	No	FK	cow_category(Category_Id)
Name	varchar	100			Yes	No		
Quantity	decimal	10,2			Yes	No		
Cost_Price_per_kg	decimal	10,2			Yes	No		
Sale_Price_per_kg	decimal	10,2			Yes	No		

Table 4.5: Customer

Attribute Name	Data Type	Size	Format	Range	Required	Unique	PK/FK	FK Reference
customer_id	int	11			Yes	Yes	PK	
name	varchar	255			Yes	No		
due_pay	decimal	10,2			Yes	No		
status_pay	enum		('Paid','Pending')		Yes	No		
created_at	datetime				Yes	No		

Table 4.6: Expense

Attribute Name	Data Type	Size	Format	Range	Required	Unique	PK/FK	FK Reference
expense_id	int	11			Yes	Yes	PK	
category_id	int	11			Yes	No	FK	category(category_id)
date	date	10,2			Yes	No		
total_price	decimal				Yes	No		
prove	varchar	255			Yes	No		

Table 4.7: milk_sale

Attribute Name	Data Type	Size	Format	Range	Required	Unique	PK/FK	FK Reference
MilkSale_Id	int	11			Yes	Yes	PK	
User_Id	int	11			Yes	No	FK	user(User_Id)
Cow_Id	int	11			Yes	No	FK	cow(Cow_Id)
Category_Id	int	11			Yes	No	FK	cow_category(Category_Id)
Litre_Collect	decimal	10,2			Yes	No		
Cost_Price_Per_Litre	decimal	10,2			Yes	No		
Sale_Price_Per_Litre	decimal	10,2			Yes	No		
Date_Collected	date				Yes	No		

Table 4.8: sales_record

Attribute Name	Data Type	Size	Format	Range	Required	Unique	PK/FK	FK Reference
record_id	int	11			Yes	Yes	PK	
customer_id	int	11			Yes	No	FK	customer(customer_id)
sale_type	enum		('milk','cow')		Yes	No		
item_id	int	11			Yes	No		
quantity	decimal	10,2			Yes	No		
price	decimal	10,2			Yes	No		
sale_date	datetime	10,2			Yes	No		

Table 4.9: shed

Attribute Name	Data Type	Size	Format	Range	Required	Unique	PK/FK	FK Reference
Shed_Id	int	11			Yes	Yes	PK	
Name	varchar	50			Yes	No		
Description	text				No	No		

Table 4.10: user

Attribute Name	Data Type	Size	Format	Range	Required	Unique	PK/FK	FK Reference
User_Id	int	11			Yes	Yes	PK	
Username	varchar	50			Yes	No		
Password	varchar	100			Yes	No		
Full_Name	varchar	100			No	No		
Email	varchar	100			No	No		
Contact	varchar	20			No	No		
User_Typ	enum		('admin','farmer','health manager','worker')		Yes	No		

Table 4.11: Vaccine

Attribute Name	Data Type	Size	Format	Range	Required	Unique	PK/FK	FK Reference
Vaccine_Id	int	11			Yes	Yes	PK	
User_Id	int	11			No	No	FK	user(User_Id)
Date_Vaccine	date				No	No		
Shed_Id	int	11			No	No	FK	shed(Shed_Id)
Dose	int	11			No	No		
Disease	varchar	255			No	No		
Duration	varchar	255			No	No		

4.3.2.2 Query design

A query design describes the structure and criteria of a database query, specifying how data should be retrieved, filtered, and organized to meet specific requirements in Farm Management System.

a) **Select Queries:**

```
$result = $conn->query("SELECT Shed_Id, Name FROM shed");
```

Desc: Fetching Shed Data for the Dropdown

b) **Left Join Queries**:

```
$result = $conn->query("SELECT v.*, s.Name as Shed_Name, u.username
FROM vaccine v
        LEFT JOIN shed s ON v.Shed_Id = s.Shed_Id
        LEFT JOIN user u ON v.User_Id = u.User_Id");
```

Desc : Reading Vaccine Data

c) **Aggregate Queries:**

```
$total_staff_stmt = $conn->prepare("SELECT COUNT(*) AS total_staff
FROM user");
```

Desc: Query for Total Staff

d) **Update Queries:**

$stmt = $conn->prepare("UPDATE customer SET name = ?, due_pay = ?, status_pay = ? WHERE customer_id = ?");

Desc: This query updates the customer data in the `customer` table.

e) **Delete Queries:**

$stmt = $conn->prepare("DELETE FROM customer WHERE customer_id = ?");

Desc: This query deletes a customer from the `customer` table.

f) **Insert Queries:**

$stmt = $conn->prepare("INSERT INTO customer (name, due_pay, status_pay) VALUES (name, 'due_pay', 'status_pay')");

Desc : This query inserts a new record into the customer table with the fields name, due_pay, and status_pay.

4.3.3 Physical Design

Physical design in a database refers to the process of translating the logical design of the database into a technical specification for storing and retrieving data efficiently. This involves deciding on the optimal storage structures, such as tables, indexes, and partitions, to support the required data access patterns and performance needs. Additionally, physical design includes utilizing database objects like stored procedures and triggers to automate and optimize data operations, implementing security mechanisms to protect data integrity and confidentiality, and establishing contingency plans for backup and recovery to ensure data availability and resilience in case of failures. Key considerations include indexing strategies, partitioning schemes, and the selection of appropriate hardware and storage configurations to meet the desired performance and scalability goals.

4.3.3.1 Selection of DBMS

The Farm Management System will be using MySQL as a Database Management System (DBMS). MySQL is an open-source DBMS known for its high performance, reliability, scalability, ease of use, strong security features, and extensive community support, making it suitable for a wide range of applications from small projects to enterprise-level systems.

4.3.3.2 Database Object

a) Procedure Operation

```
CREATE PROCEDURE CustomerOperations(
    IN p_action VARCHAR(10),
    IN p_customer_id INT,
    IN p_name VARCHAR(255),
    IN p_due_pay DECIMAL(10, 2),
    IN p_status_pay VARCHAR(10)
)
BEGIN
    IF p_action = 'create' THEN
        INSERT INTO customer (name, due_pay, status_pay)
        VALUES (p_name, p_due_pay, p_status_pay);
    ELSEIF p_action = 'update' THEN
        UPDATE customer
        SET name = p_name, due_pay = p_due_pay, status_pay = p_status_pay
        WHERE customer_id = p_customer_id;
    ELSEIF p_action = 'delete' THEN
```

```
    DELETE FROM customer
    WHERE customer_id = p_customer_id;
  END IF;
END
```

Desc :

Stored procedure for the insert, update, and delete operations on the `customer` table based on your PHP code. This stored procedure handles creating, updating, and deleting customer records

b) Trigger Operation

```
CREATE TRIGGER log_user_update
AFTER UPDATE ON user
FOR EACH ROW
BEGIN
  DECLARE changed_fields TEXT;

  SET changed_fields = '';

  IF OLD.Username <> NEW.Username THEN
```

```
    SET changed_fields = CONCAT(changed_fields, 'Username ');
END IF;

IF OLD.Password <> NEW.Password THEN
    SET changed_fields = CONCAT(changed_fields, 'Password ');
END IF;

IF OLD.Full_Name <> NEW.Full_Name THEN
    SET changed_fields = CONCAT(changed_fields, 'Full_Name ');
END IF;

IF OLD.Email <> NEW.Email THEN
    SET changed_fields = CONCAT(changed_fields, 'Email ');
END IF;

IF OLD.Contact <> NEW.Contact THEN
    SET changed_fields = CONCAT(changed_fields, 'Contact ');
END IF;
```

```
    INSERT INTO profile_updates (user_id, changed_fields)

    VALUES (NEW.User_Id, TRIM(changed_fields));

END;
```

Desc:

This trigger will execute after every update on the user table. It checks each field to see if it has changed and constructs a string of the changed fields, which it then inserts into the profile_updates table.

By implementing this trigger, you ensure that all updates to user profiles are logged automatically, with details about which fields were changed.

4.4 Graphical User Interface (GUI)

Graphical User Interface (GUI) for a farm management system involves designing an intuitive and user-friendly interface that allows users to manage various aspects of farm operations efficiently. Below is a detailed description of the main components and features that should be included in the GUI for a farm management system.

4.4.1 Login GUI

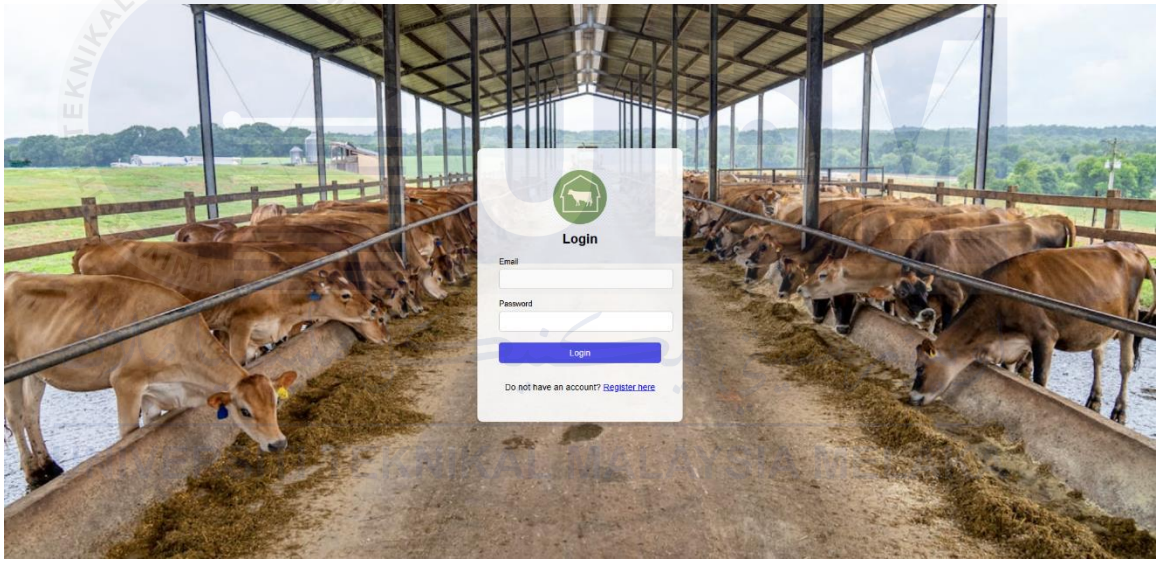

Figure 4.2 Log In Interface

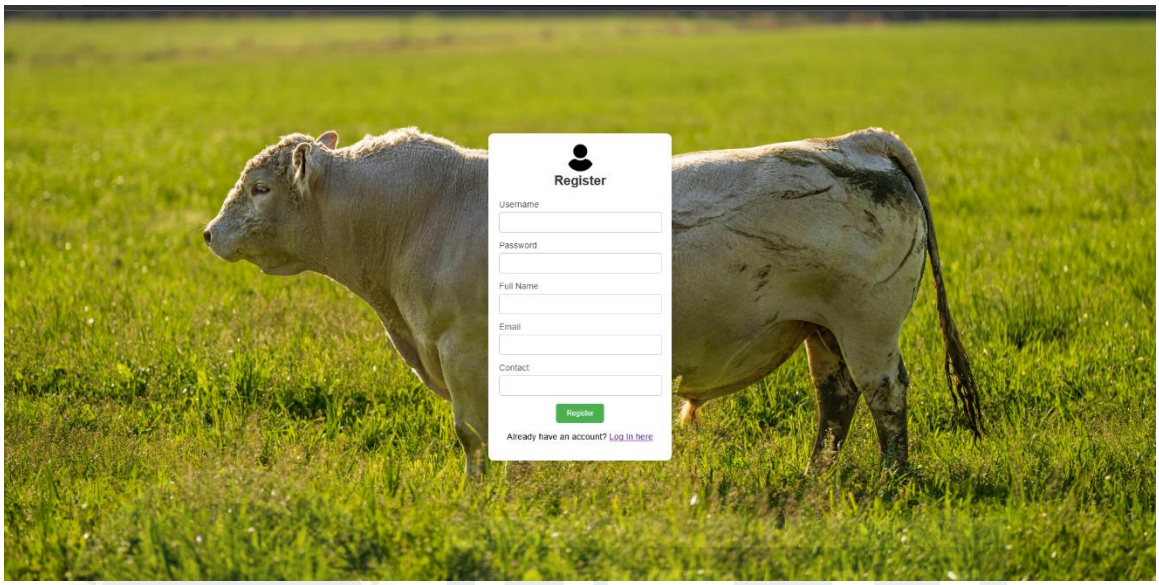

Figure 4.3 Register Interface

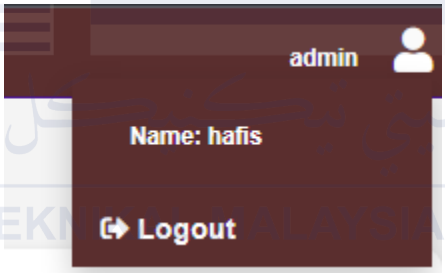

Figure 4.4 Log Out Button

4.4.2 Admin GUI

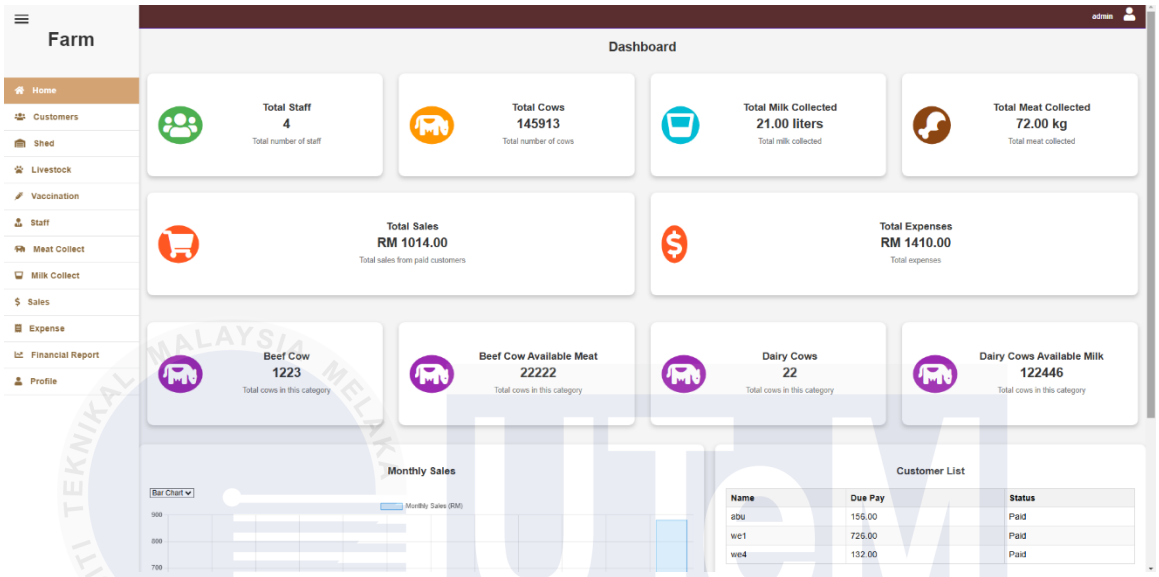

Figure 4.5 Admin Dashboard Interface

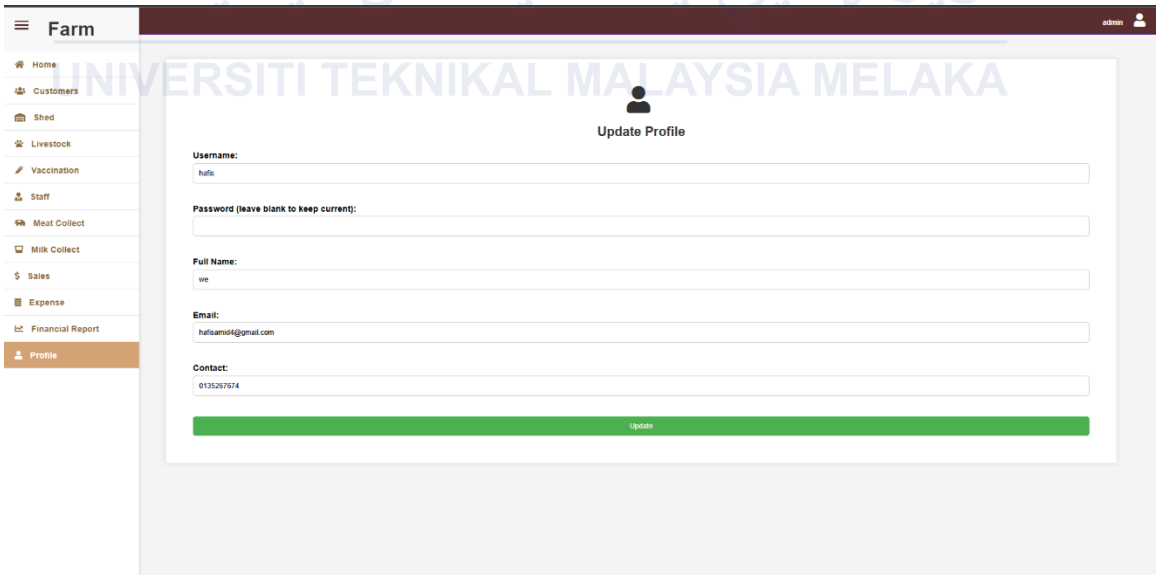

Figure 4.6 Profile Interface

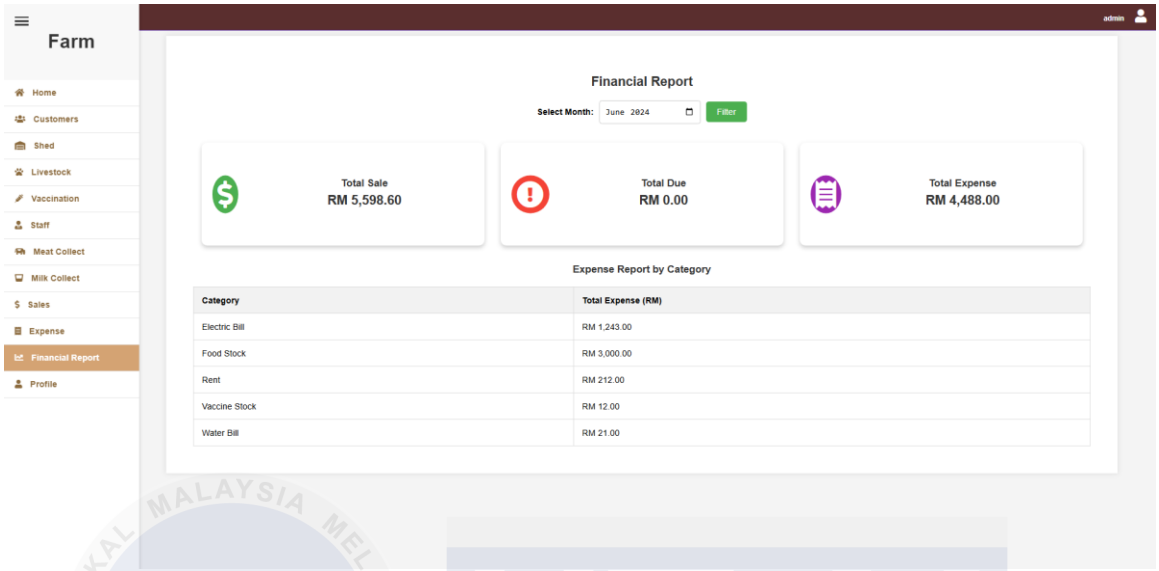

Figure 4.7 Financial Interface

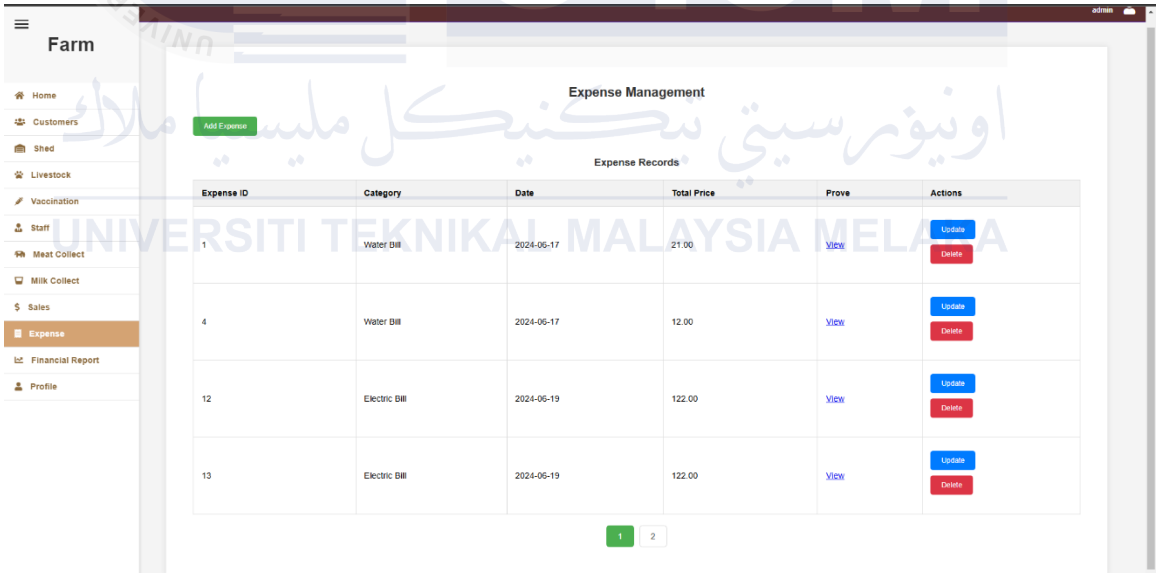

Figure 4.8 Expense Interface

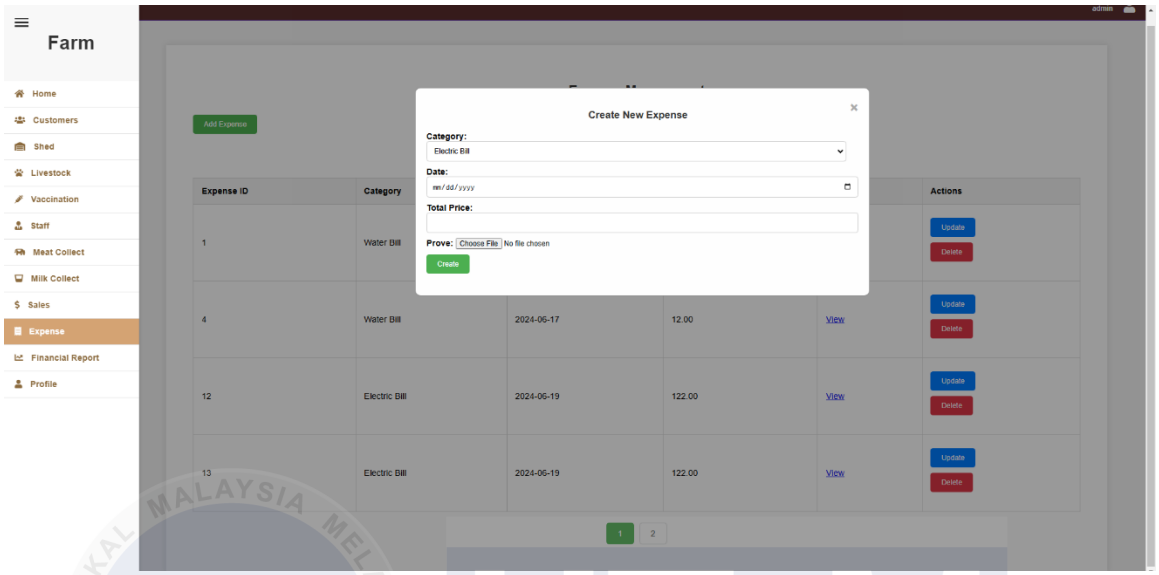

Figure 4.9 Add Expense Interface

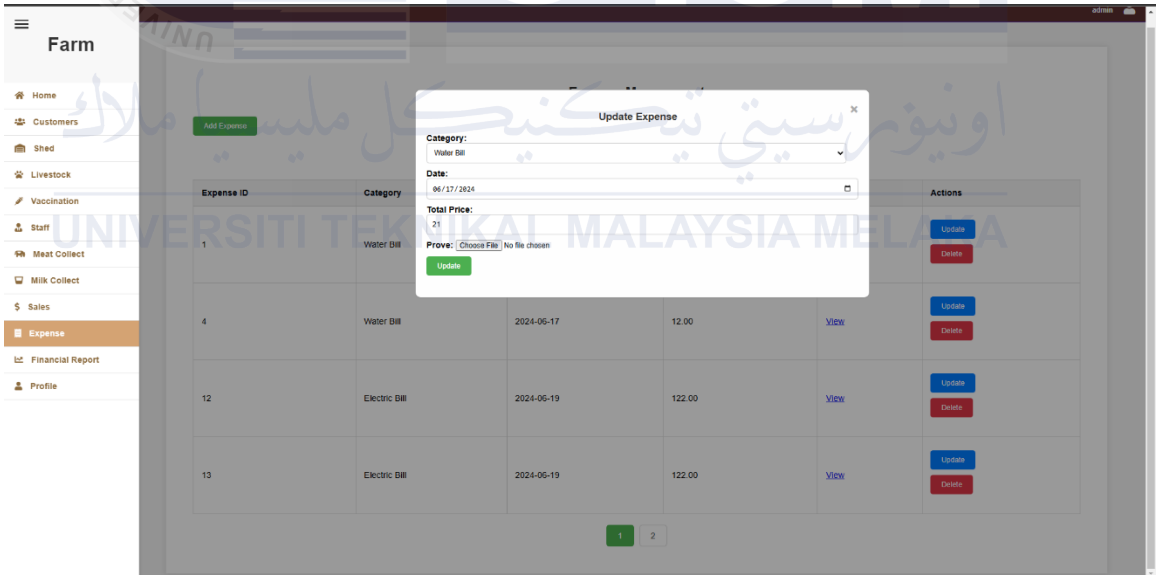

Figure 4.10 Update Expense Interface

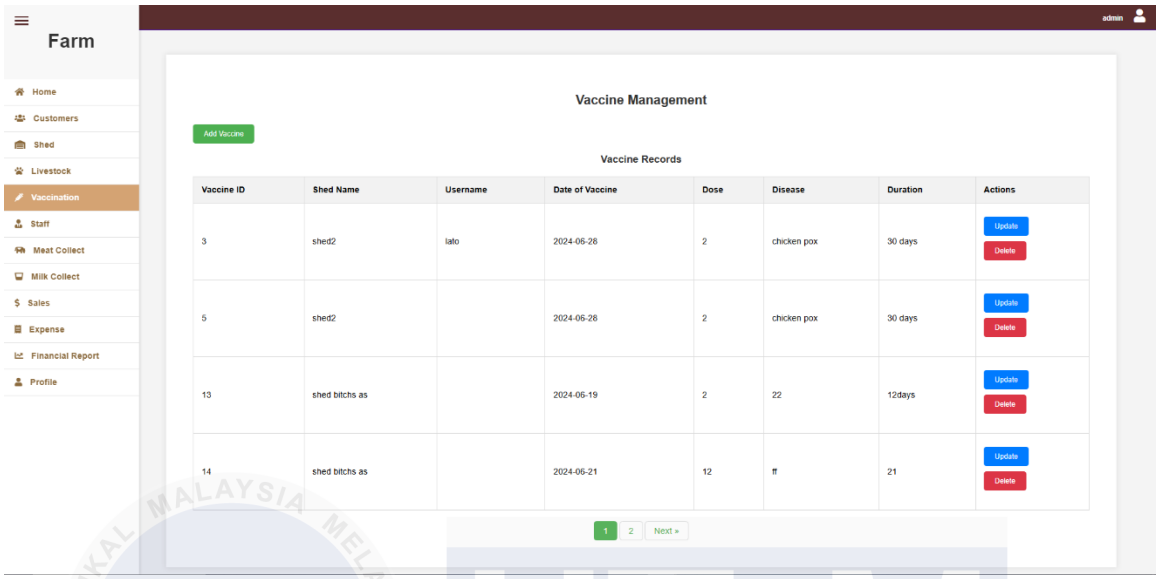

Figure 4.11 Vaccine Interface

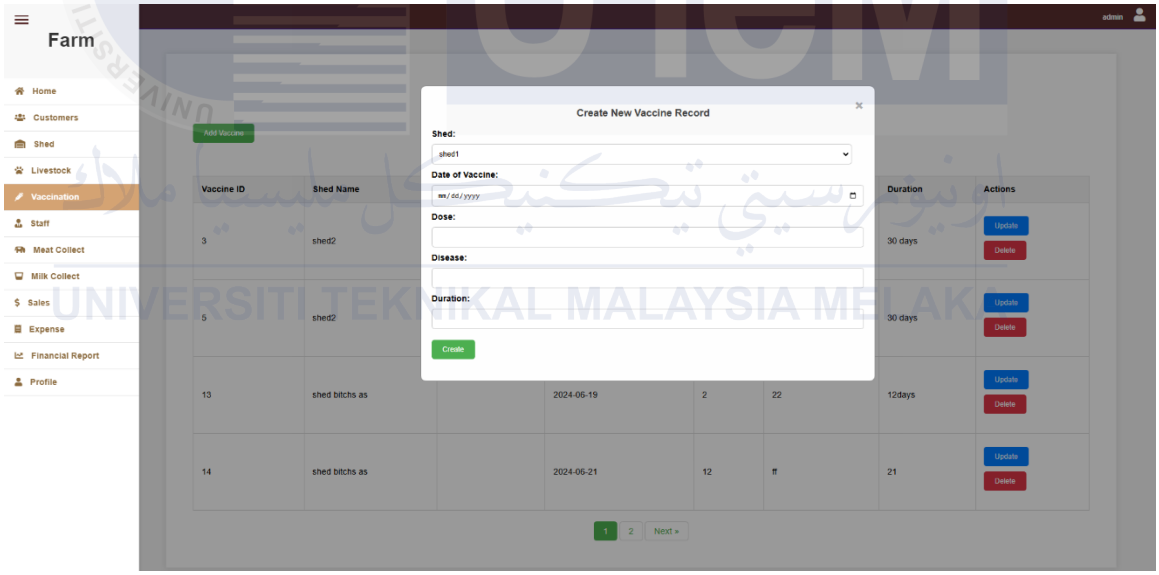

Figure 4.12 Add Vaccine Interface

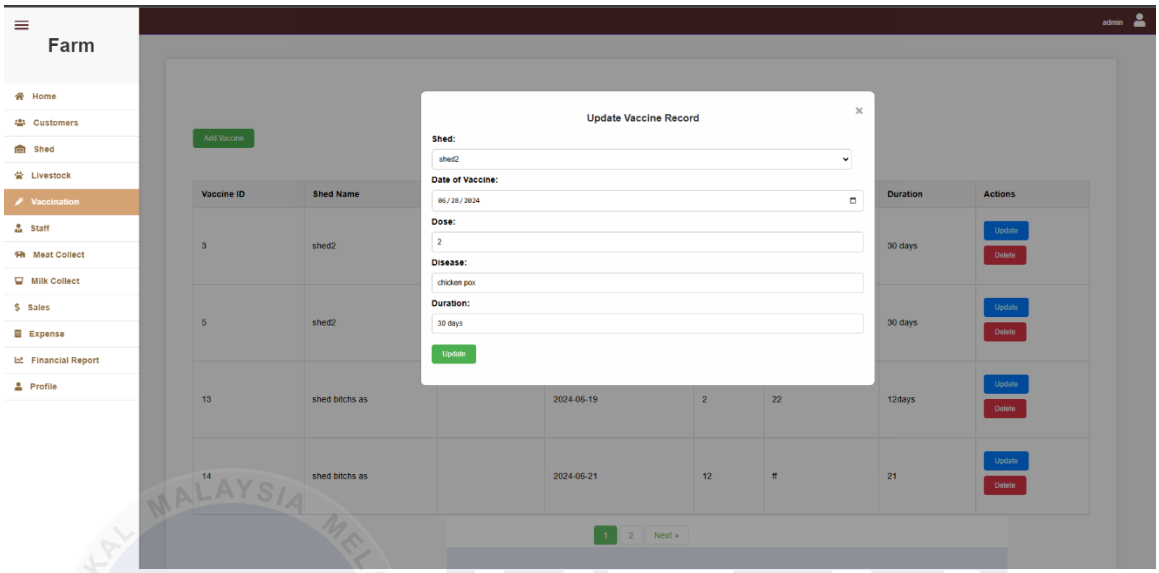

Figure 4.13 Update Vaccine Interface

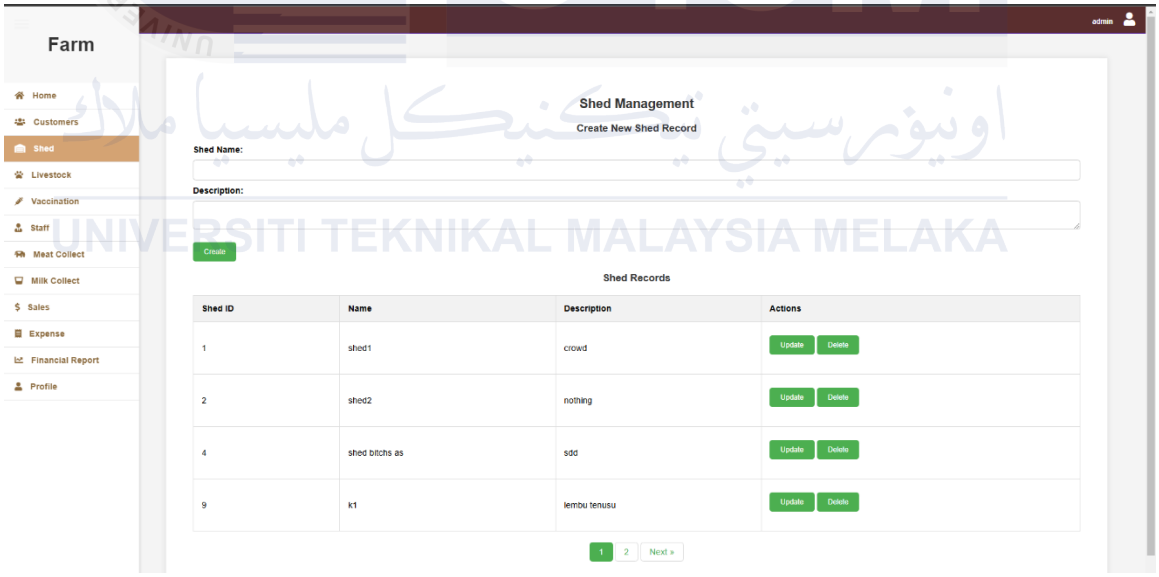

Figure 4.14 Add Shed Interface

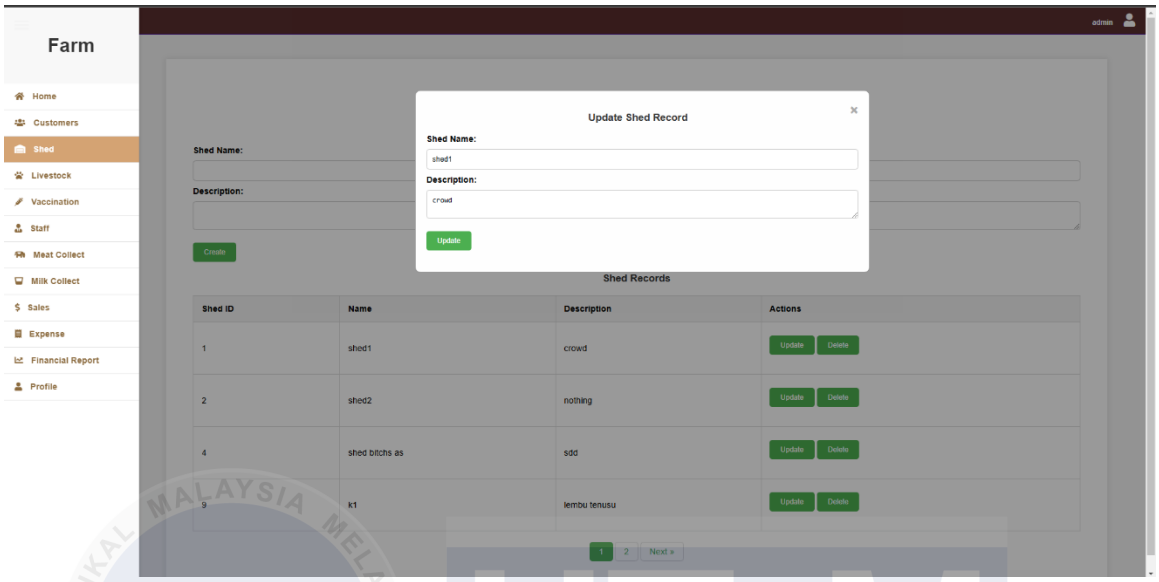

Figure 4.15 Update Shed Interface

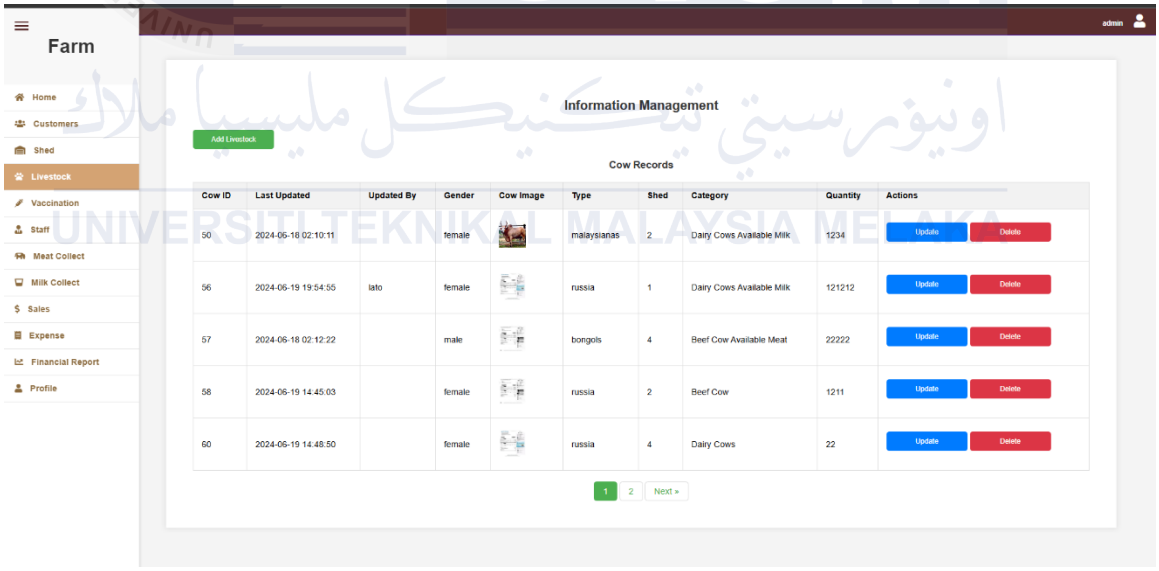

Figure 4.16 Livestock Interface

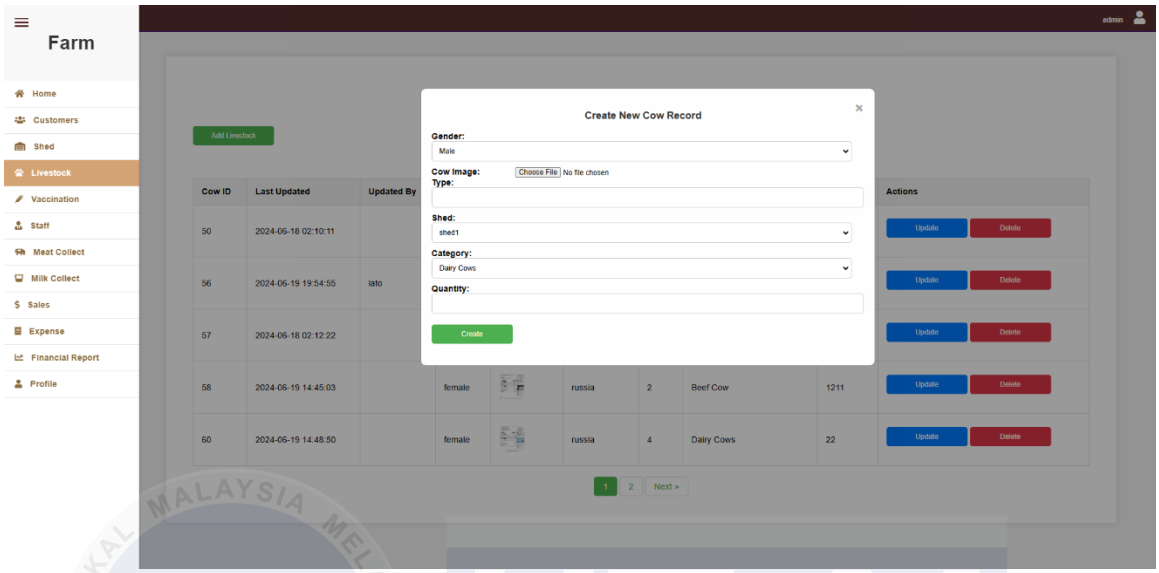

Figure 4.17 Add Livestock Interface

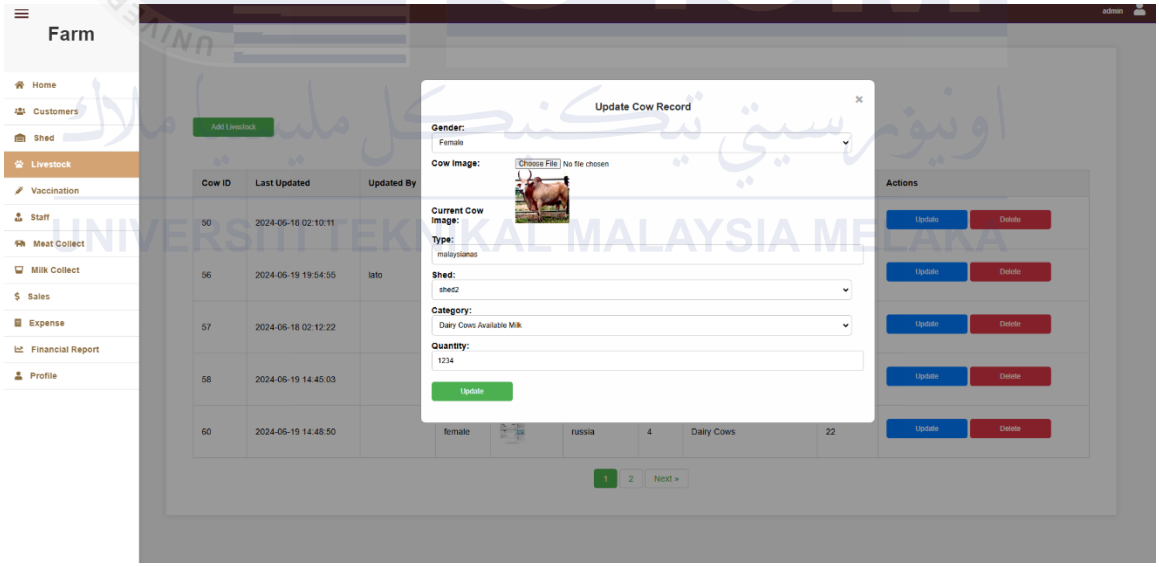

Figure 4.18 Update Livestock Interface

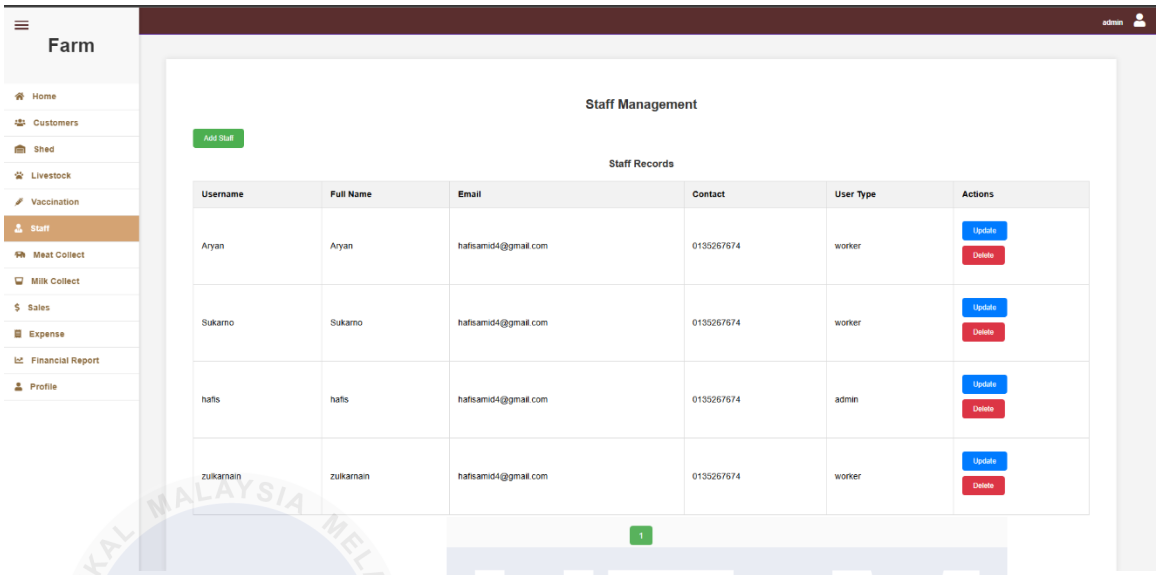

Figure 4.19 Staff Interface

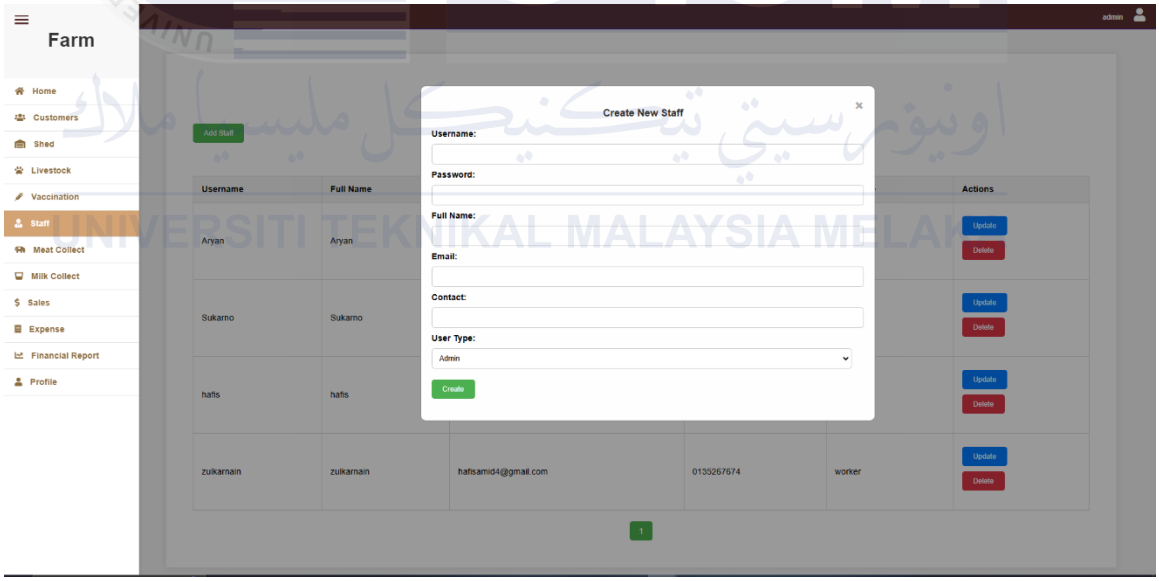

Figure 4.20 Add Staff Interface

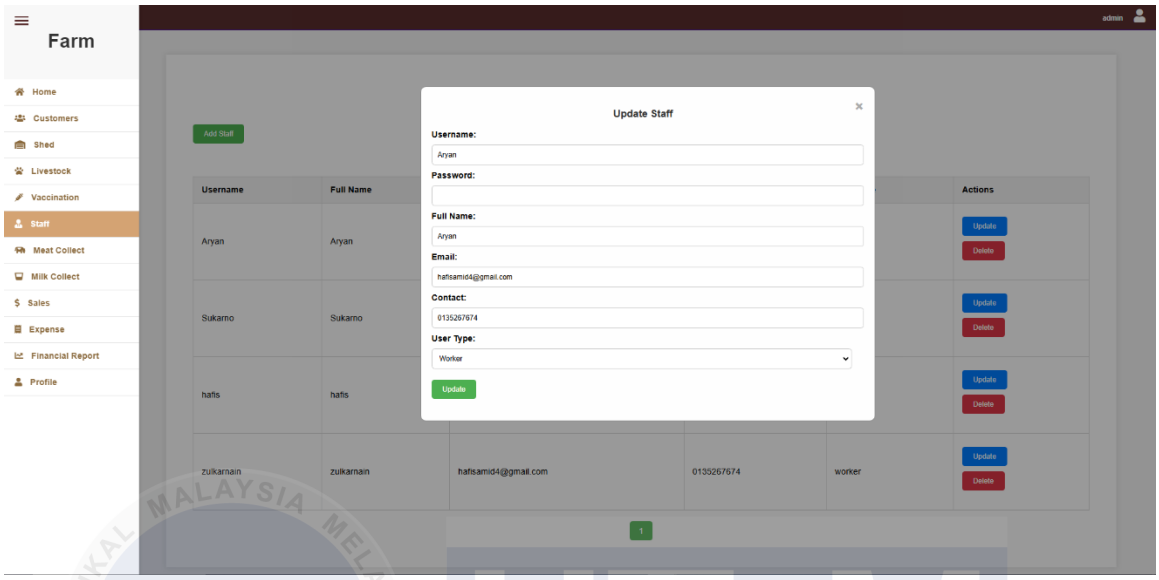

Figure 4.21 Update Staff Interface

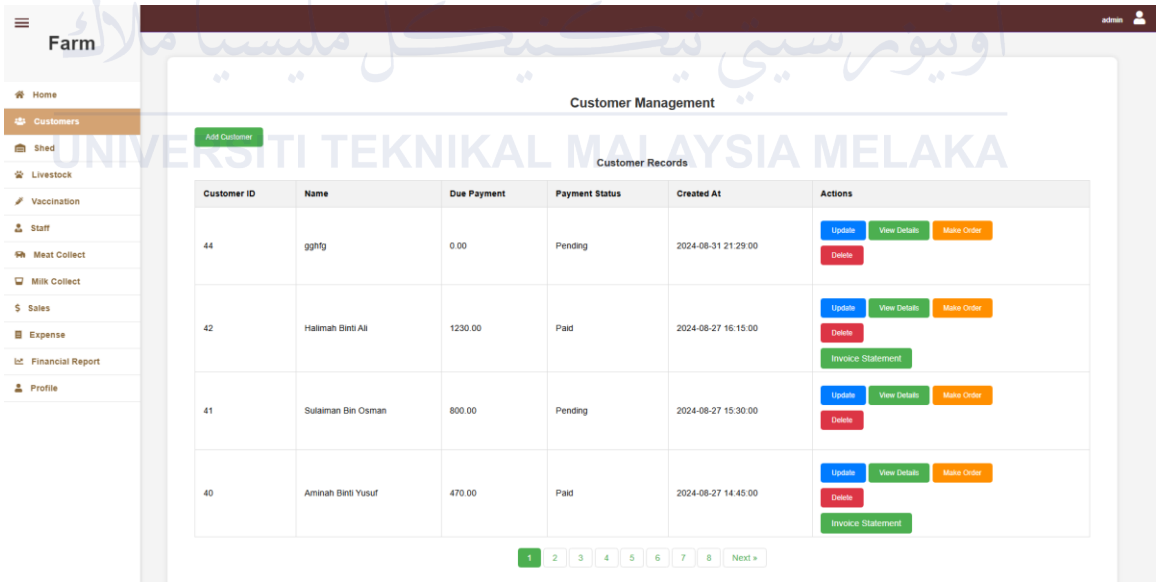

Figure 4.22 Customer Interface

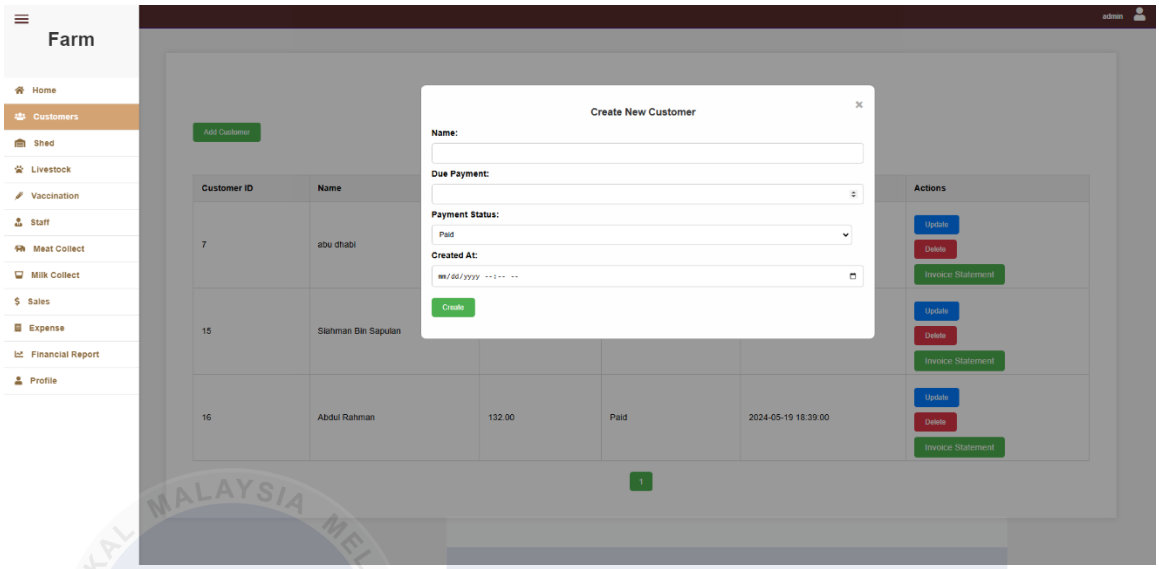

Figure 4.23 Add Customer Interface

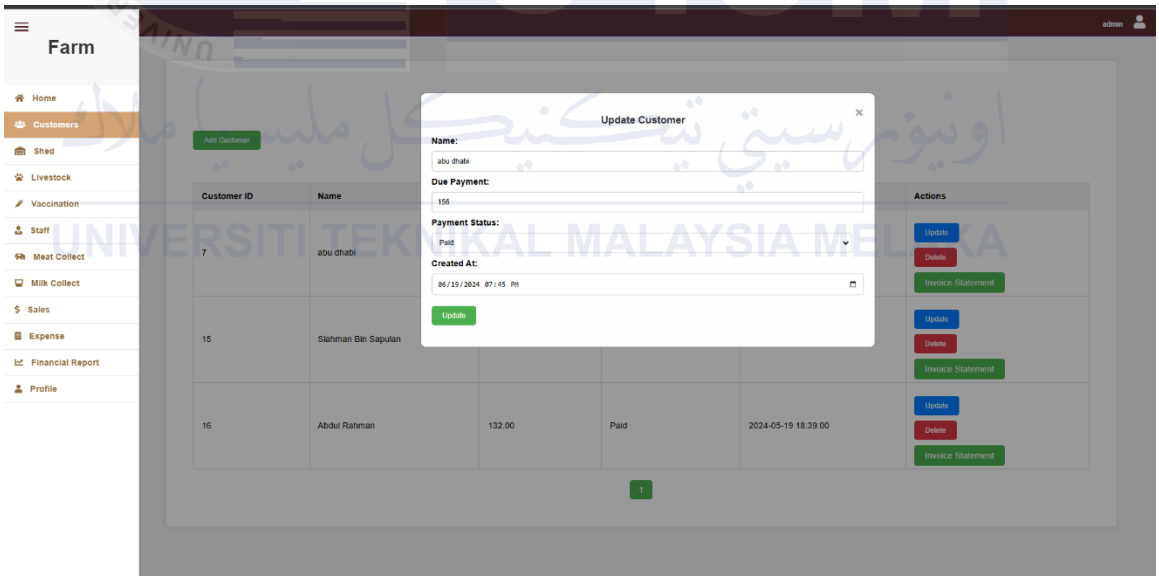

Figure 4.24 Update Customer Interface

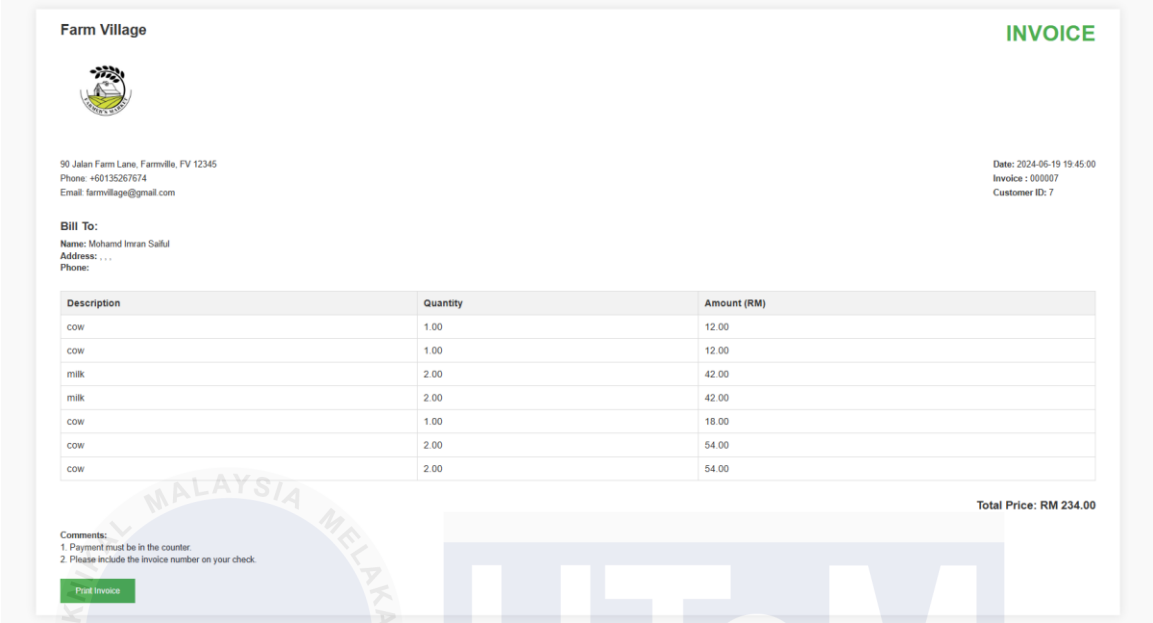

Figure 4.25 Invoice Customer Interface

Figure 4.26 Meat Collect Interface

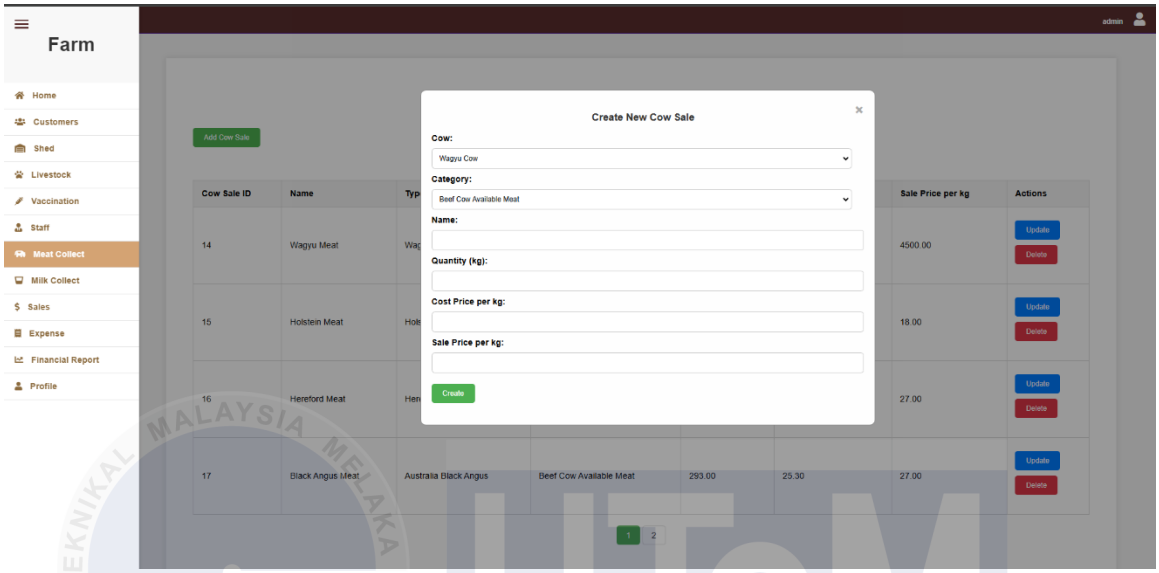

Figure 4.27 Add Meat Collect Interface

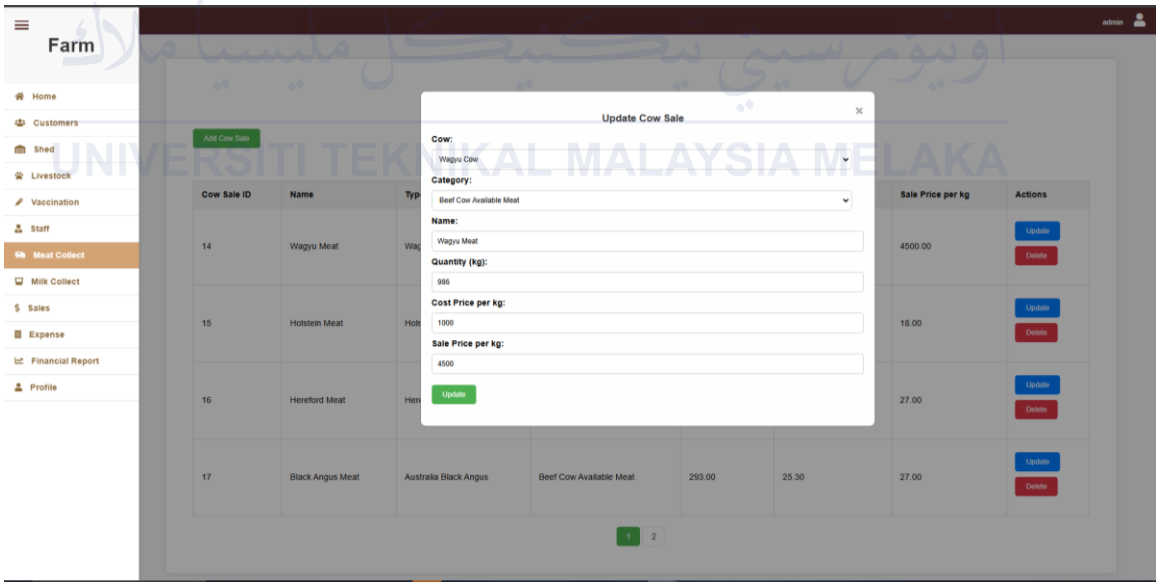

Figure 4.28 Update Meat Collect Interface

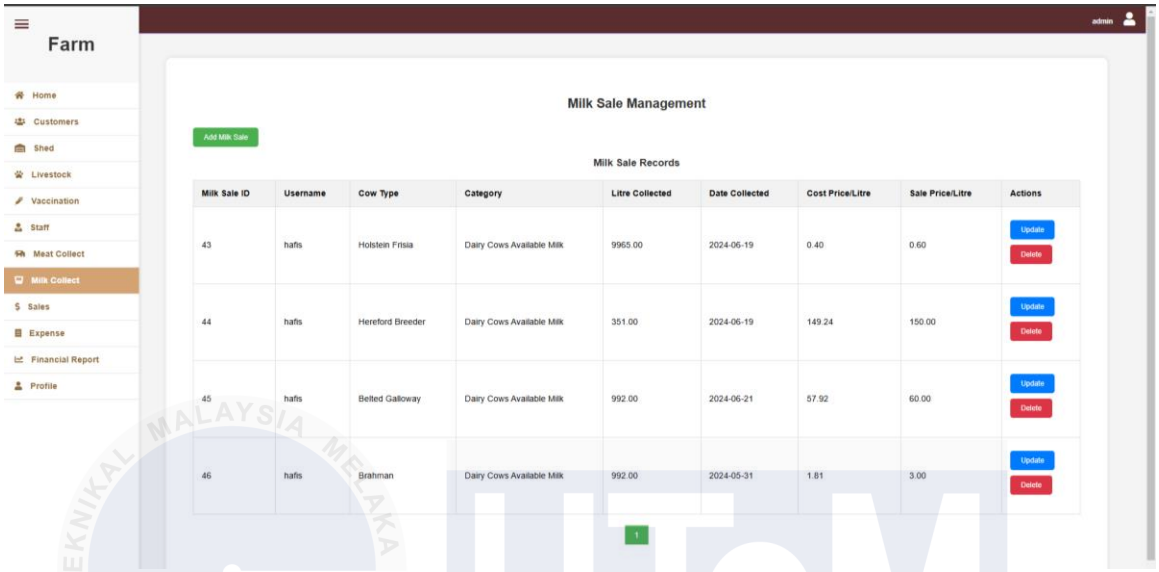

Figure 4.29 Milk Collect Interface

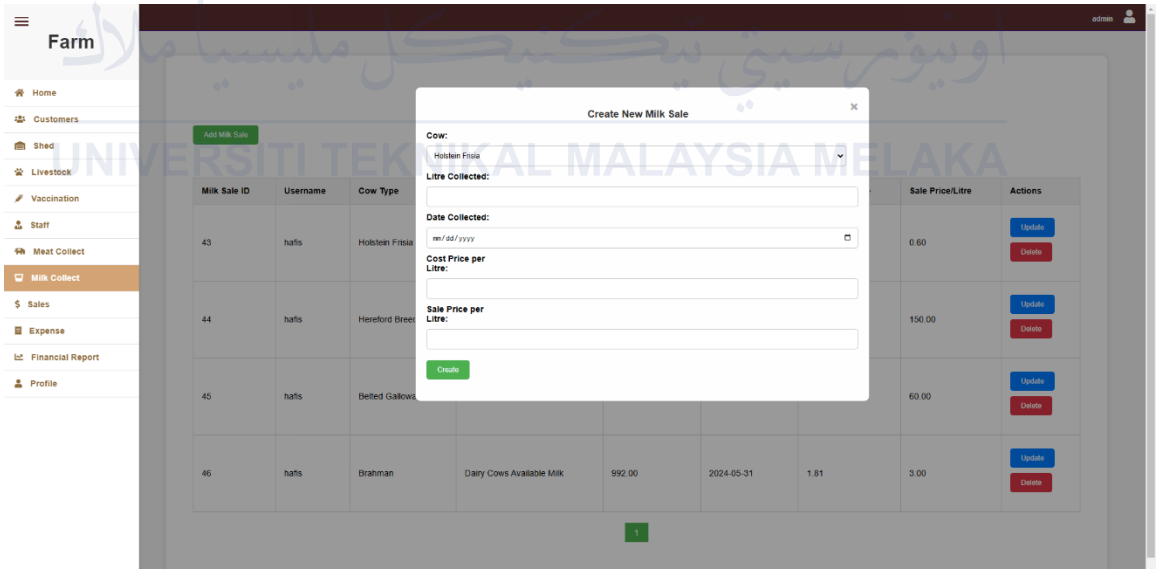

Figure 4.30 Add Milk Collect Interface

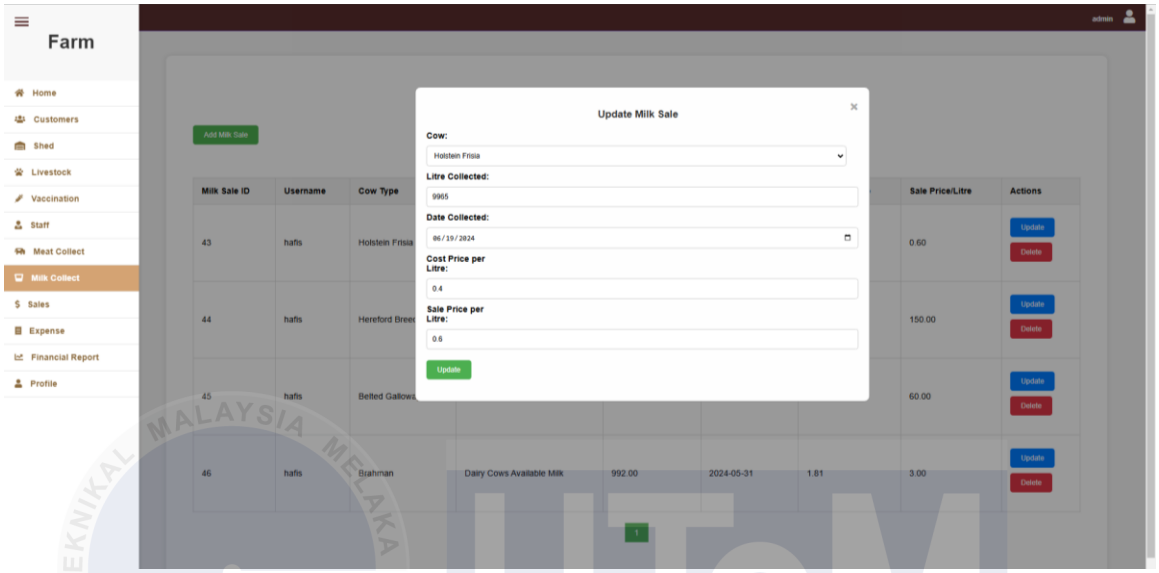

Figure 4.31 Update Milk Collect Interface

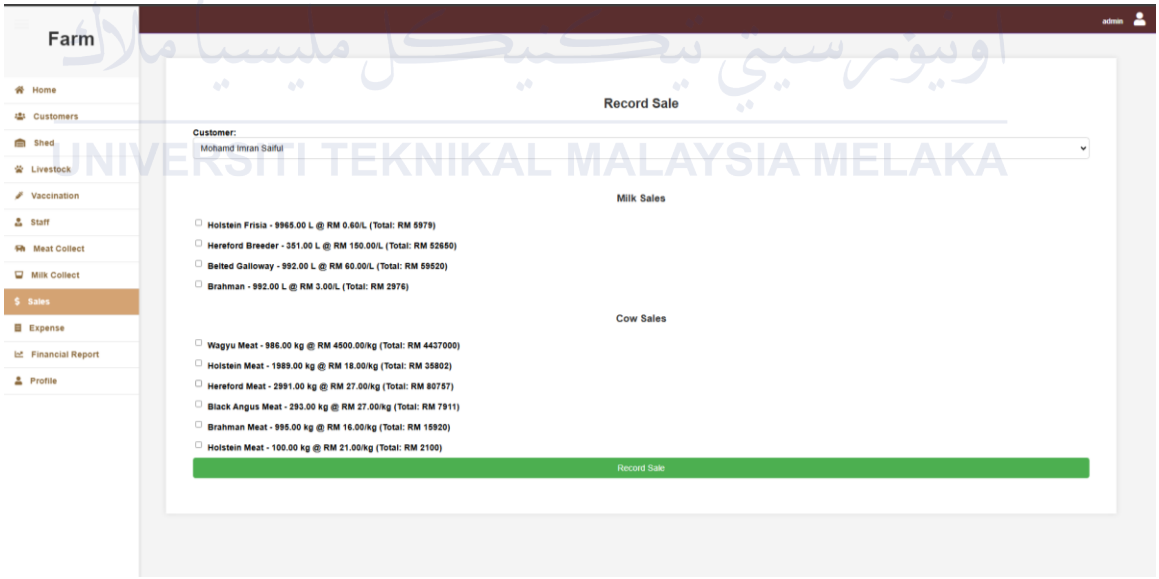

Figure 4.32 Sale Interface

4.4.3 Worker (GUI)

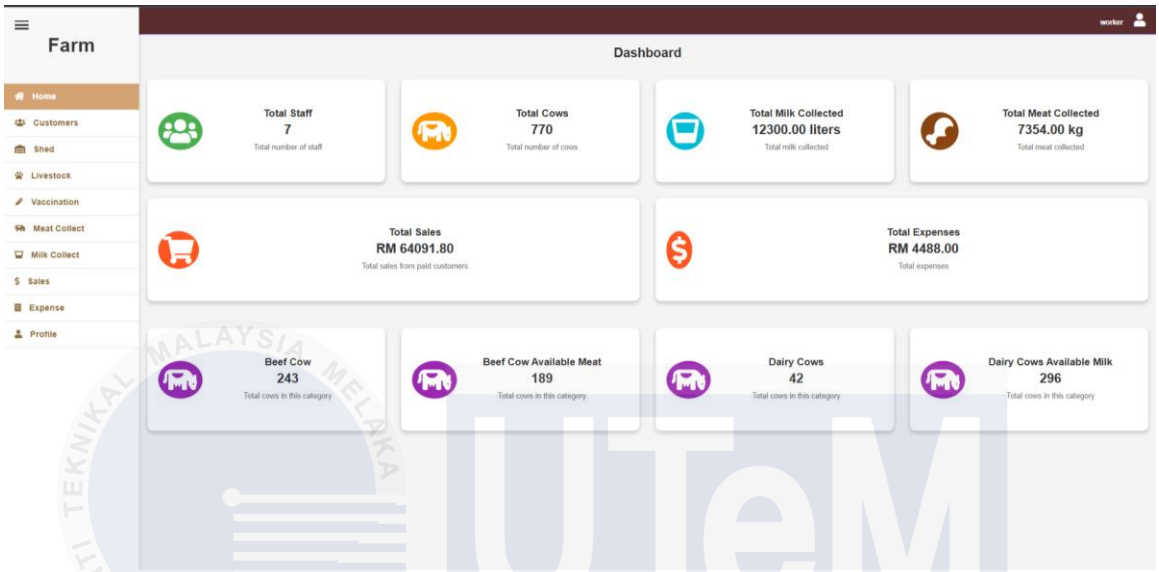

Figure 4.33 Worker Dashboard Interface

4.5 CONCLUSION

The design phase of the Farm Management System is pivotal as it transitions the logical system design from the analysis phase into a detailed physical system framework. This comprehensive process includes defining the input/output requirements, processing specifications, and database schemas, which are crucial for ensuring the system's functionality and efficiency. The design incorporates various elements such as Entity-Relationship Diagrams (ERD), business rules, data dictionary, normalization, and the Database Management System (DBMS). Additionally, the design emphasizes the importance of Graphical User Interface (GUI) to ensure an intuitive and user-friendly experience for managing farm operations. This meticulous design process lays a solid foundation for the subsequent development and implementation phases, ensuring the system meets specific business needs and operates seamlessly.

CHAPTER 5: IMPLEMENTATION

5.1 Introduction

In this chapter it is necessary to set up the development environment and create the database are covered in detail in this chapter, which concentrates on the farm management system's implementation phase. During the implementation phase, the software development environment, such as setting up the required software, establishing the database, and creating table, view, stored procedure, and trigger database objects. By After this stage is over, the farm management site database will be completely operational and prepared for connection.

5.2 Software Development Environment Setup

System and Database Environment Setup:

Configured on a local server in PHP 8.2.12, MariaDB 10.4.32 as the (DBMS).

The development environment setup :

1. Installation : Installing Xampp and Apache as the local server
2. Configuration : Configure as admin and using username hafis and the password is hafis.
3. Database Service : Starting the MySQL or MariaDB.

Database creation and object:

Creates and structures the farm_management database with tables, indexes, and constraints for managing farm-related data, including categories, cows, sales, customers, expenses, users, sheds, and vaccines.

5.3 Database Implementation

DDL statement :

- **Table `customer` example :**

```
CREATE TABLE `customer` (
  `customer_id` int(11) NOT NULL,
  `name` varchar(255) NOT NULL,
  `due_pay` decimal(10,2) NOT NULL,
  `status_pay` enum('Paid','Pending') NOT NULL,
  `created_at` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

- Table `sales_record` :

```
CREATE TABLE `sales_record` (
  `record_id` int(11) NOT NULL,
  `customer_id` int(11) NOT NULL,
  `sale_type` enum('milk','cow') NOT NULL,
  `item_id` int(11) NOT NULL,
  `quantity` decimal(10,2) NOT NULL,
  `price` decimal(10,2) NOT NULL,
  `sale_date` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

- Store procedure and trigger:

Procedure :

- **GetCustomerDetails Procedure:**

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `GetSalesRecords` (IN `p_customer_id` INT)
BEGIN
    SELECT
        sale_type,
        quantity,
        price
    FROM
        sales_record
    WHERE
        customer_id = p_customer_id;
END;
```

Figure 5.1 Procedure GetCustomerDetails

This procedure is used to retrieve detailed information about a specific customer, including both basic customer information and additional details like address, city, state, postal code.

Trigger:

- after_login_attempt :

```
DELIMITER $$
CREATE TRIGGER `after_login_attempt`
AFTER INSERT ON `user`
FOR EACH ROW
BEGIN
    IF NEW.User_Id IS NOT NULL THEN
        INSERT INTO login_attempts (username, attempt_time, success)
        VALUES (NEW.Username, NOW(), 'yes');
    ELSE
        INSERT INTO login_attempts (username, attempt_time, success)
        VALUES (NEW.Username, NOW(), 'no');
    END IF;
END$$
DELIMITER ;
```

Figure 5.2 Trigger After Log In Attempt

This trigger is designed to automatically insert a record into the login_attempts table whenever a new user is created in the user table. The trigger checks if the User_Id is not null, and based on that, it logs a successful or failed login attempt.

- Data loading : Process of inserting data into a database table.

  - Data loading in `customer` table.

```
INSERT INTO `customer` (`customer_id`, `name`, `due_pay`, `status_pay`, `created_at`) VALUES
(7, 'Mohamd Imran Saiful', '282.00', 'Paid', '2024-06-19 19:45:00'),
(15, 'Irfan danial', '726.00', 'Paid', '2024-06-19 18:24:00'),
(16, 'Nurul Saadiah Binti Rahiman', '132.00', 'Paid', '2024-05-19 18:39:00'),
(17, 'Saidatul Insyirah Binti Kasmin', '1994.00', 'Paid', '2024-05-21 02:30:00'),
(18, 'Suresh A/L Rajit', '13555.80', 'Paid', '2024-04-20 08:35:00'),
(19, 'Salman Bin Hadi', '1.20', 'Pending', '2024-04-20 08:36:00'),
(20, 'Hadi Rashid Bin Samsul', '0.00', 'Pending', '2024-04-20 08:40:00'),
(21, 'Sulaiman Bin Sarifudin', '102.00', 'Paid', '2024-03-20 08:37:00'),
(22, 'Suzana Binti Kamsuri', '4668.60', 'Paid', '2024-02-20 08:37:00'),
(23, 'Suraya Binti Said', '4500.00', 'Pending', '2024-02-20 08:37:00'),
(24, 'Sukar Bin Joekarno', '9000.00', 'Paid', '2024-01-20 08:38:00'),
(25, 'Salah Bin Salahudin', '158.00', 'Paid', '2023-12-20 08:38:00'),
(26, 'Rizalman Bin Ruzaidi', '4801.60', 'Paid', '2023-11-20 08:39:00'),
(27, 'Manaf Bin Sayaf', '427.20', 'Paid', '2023-10-20 08:39:00'),
(28, 'Henry William', '13713.60', 'Paid', '2023-09-20 08:40:00'),
(29, 'Sabri Bin Yunus', '513.20', 'Paid', '2023-08-20 00:43:00'),
(30, 'Susanti A/P Ah Seng', '9427.20', 'Paid', '2023-07-20 08:41:00'),
(31, 'hafis', '4590.60', 'Paid', '2024-06-20 11:20:00');
```


Figure 5.3 Data Loading Customer Table

5.4 Conclusion

In conclusion, the implementation of the farm management system involved setting up a local development environment with XAMPP, configuring MariaDB, and creating the necessary database objects such as tables, stored procedures, and triggers. The database was structured to manage various farm-related data, ensuring that it is fully operational and ready for integration with the farm management site. The process included data loading into key tables, like the customer table, to populate the database with essential information for system functionality.

CHAPTER 6: TESTING

6.1 Introduction

The testing phase is essential to ensure the quality and reliability of the Farm Management System. This phase focuses on validating the system's functionality, performance, and usability by conducting various tests. Testing identifies defects, verifies that the system meets the specified requirements, and ensures it operates smoothly in the intended environment. The purpose of this chapter is to outline the test plan, the strategy used, the environment setup, and the analysis of the test results, leading to the final conclusions about the system's readiness for deployment.

6.2 Test Plan

6.2.1 Test Organization

The test organization involves the roles and responsibilities of the testing team.

Table 6.1 Test Oraganization

Test ID	Name	Position	Task
T1	Mohamad Hafiszudin Bin Amid	System Developer and Test Manager	Prepare test plan and test scripts; perform unit testing and integration testing
T2	Suresh A/L Rajit	Worker	Testing customer management module
T3	Saidatul Insyirah Binti Kasmin	Worker	Testing livestock management, sales management, and financial reporting modules

6.2.2 Test Organization

The test environment simulates the production environment to ensure the system's behavior is consistent and predictable. The setup includes the hardware, software, network configurations, and other tools necessary for testing.

- **Server Setup:** Configuring a local server using XAMPP to host the Farm Management System.
- **Database Setup:** Setting up MariaDB for testing purposes, including loading sample data relevant to farm operations.
- **Client Workstations:** Setting up client machines that simulate different types of users, such as admins and workers.

6.2.2.2 Software Application

The software application setup includes the installation and configuration of all the applications and tools required for testing:

- **XAMPP:** Used to create a local server environment for hosting the application.
- **MariaDB:** Serves as the database management system for storing and managing farm-related data.
- **PHP:** The scripting language used to develop the Farm Management System.
- **Sublime Text:** A text editor used for modifying and debugging code during testing.

6.2.2.3 System Software

The system software refers to the underlying software components that support the operation of the Farm Management System:

- **Operating System:** Windows 10 for both server and client machines.
- **Web Browser:** Google Chrome, Mozilla Firefox, and Microsoft Edge for cross-browser testing.
- **Database Management System:** MariaDB, version 10.4.32.
- **Web Server:** Apache, provided by XAMPP.

6.2.2.4 System Hardware

The hardware setup includes the physical devices necessary to perform testing:

- **Server Hardware:**
  - Processor: AMD Ryzen 5 5600G with Radeon Graphics @ 3.90 GHz
  - Memory: 16 GB RAM
  - Storage: 500 GB SSD
  - Graphics: NVIDIA GeForce RTX 3060 Ti
- **Client Workstations:**
  - Processor: Intel Core i5
  - Memory: 8 GB RAM
  - Storage: 256 GB SSD
  - Display: 24-inch monitor with a resolution of 1920x1080

6.2.3 Test Schedule

The test schedule outlines the timeline for different testing activities:

Table 6.2 Test Schedule

Module	Test Type	Start Date	End Date	Tester
User Management Module	Unit Testing	01/08/2024	02/08/2024	T1, T3
Livestock Management Module	Unit Testing, Integration Testing	03/08/2024	05/08/2024	T3
Sales Management Module	Unit Testing	06/08/2024	07/08/2024	T3
Financial Reporting Module	Integration Testing	08/08/2024	09/08/2024	T3
Customer Management Module	Integration Testing	10/08/2024	12/08/2024	T2

6.3 Test Strategy

The test strategy defines the approach and methods used to test the "Farm Management System." The strategy involves both black-box and white-box testing techniques, ensuring that all system functionalities are thoroughly evaluated.

6.3.1 Classes of Test

a) Error Handling Test:

Error handling tests evaluate how the system manages and responds to user errors, such as entering invalid customer details or incorrect payment information.

b) Security Test:

Security tests ensure that the "Farm Management System" is protected from unauthorized access and that sensitive data, such as financial records, is securely handled.

c) Integration Test:

Integration tests focus on verifying the interactions between different modules, such as the integration between the sales management module and the financial reporting module.

6.4 Test Design

The test design phase involves creating detailed test cases and scenarios to validate the "Farm Management System" against its defined requirements. This phase is broken down into test descriptions and test data preparation.

6.4.1 Test Description

Test Module: Customer Management Module

Test Type: Unit Testing

Test Strategy: Black Box Testing

Test Description: Adding a new customer to the system.

Table 6.3 Test Customer Management Module

Test Case ID	Test Case Description	Test Step	Expected Result
TC1_01	Verify that a new customer can be added with valid data.	1. Go to the customer management page. 2. Input customer name, due payment, payment status, date created, address, contact details and additional info. 3. Click "Add Customer" button.	Customer added successfully. Display the new customer data.
TC1_02	Verify that adding a new customer fails if required fields are left empty.	Leave any input field empty.	Customer not added. Display message "Please fill out this field."

Test Module: Integration between Sales Management Module and Financial Reporting Module

Test Type: Integration Testing

Test Strategy: Black Box Testing

Test Description: Generating a financial report after recording a sale.

Table 6.4 Test Integration between Sales Management Module and Financial

Test Case ID	Test Case Description	Test Step	Expected Result
TC10_01	Verify that the financial report is updated after recording a new sale.	1. Record a new sale in the sales management module. 2. Generate a financial report.	Financial report updated successfully in total sales.
TC10_02	Verify that the financial report in total sales nothing to display if there are no sales.	Attempt to display report without recording any sales.	Report display in total sales is empty.

Test Module: Livestock Management Module

Test Type: Unit Testing

Test Strategy: Black Box Testing

Test Description: Adding a new livestock record to the system.

Table 6.5 Test Livestock Management Module

Test Case ID	Test Case Description	Test Step	Expected Result
TC2_01	Verify that a new livestock record can be added with valid data.	1. Go to the livestock management page. 2. Input livestock gender, image, type, category, shed ID, and quantity. 3. Click "Add Livestock" button.	Livestock record added successfully.
TC2_02	Verify that adding a new livestock record fails if required fields are left empty.	Leave any input field empty.	Livestock record not added. Display message "Please fill out this field."

Test Module: Expense Management Module

Test Type: Unit Testing

Test Strategy: Black Box Testing

Test Description: Recording an expense in the system.

Table 6.6 Test Expense Management Module

Test Case ID	Test Case Description	Test Step	Expected Result
TC3_01	Verify that an expense can be recorded with valid data.	1. Go to the expense management page. 2. Input category, date, and total price. 3. Upload proof of expense. 4. Click "Record Expense" button.	Expense recorded successfully. Display message "Expense recorded successfully."
TC3_02	Verify that recording an expense fails if the total price is not a valid number.	Enter an invalid total price (e.g., a text string).	Expense not recorded. Display message "Please enter a valid number for total price."
TC3_03	Verify that recording an expense fails if the proof of expense is not uploaded.	Do not upload any proof of expense.	Expense not recorded. Display message "Proof of expense is required."

Test Module: Shed Management Module

Test Type: Unit Testing

Test Strategy: Black Box Testing

Test Description: Adding a new shed record to the system.

Table 6.7 Test Shed Management Module

Test Case ID	Test Case Description	Test Step	Expected Result
TC4_01	Verify that a new shed can be added with valid data.	1. Go to the shed management page. 2. Input shed name and description. 3. Click "Add Shed" button.	Shed added successfully. Display message "Shed added successfully."
TC4_02	Verify that adding a new shed fails if the shed name is not provided.	Leave the shed name field empty.	Shed not added. Display message "Shed name is required."
TC4_03	Verify that adding a new shed fails if a shed with the same name already exists.	Enter a shed name that already exists in the system.	Shed not added. Display message "Shed name already exists."

6.4.2 Test Data

Test Data for Customer Management:

Table 6.8 Test Data for Customer Management

Test Data ID	Name	Address	Phone Number
TD1_01	Ahmad Faizal	123 Jalan Merdeka	0123456789
TD1_02		456 Jalan Tun Razak	0134567890
TD1_03	Suraya Binti Said	789 Jalan Ampang	0145678901

Test Data for Sales Management:

Table 6.9 Test Data for Sales Management

Test Data ID	Sale Type	Item ID	Quantity	Price	Sale Date
TD10_01	cow	14	2.00	9000.00	2024-08-27 09:13:43
TD10_02	milk	43	20.00	12.00	2024-08-27 09:13:43

Test Data for Livestock Management:

Table 6.10 Test Data for Livestock Management

Test Data ID	Gender	Image	Livestock Type	Shed ID	Category	Quantity
TD2_01	Male	uploads/Cow.jpg	Wagyu Cow	17	Beef Cow	30
TD2_02	Female	uploads/Cow.jpg	Holstein Frisia	18	Dairy Cows	20

Test Data for Expense Management:

Table 6.11 Test Data for Expense Management

Test Data ID	Category	Date	Total Price	Proof of Expense
TD3_01	Water Bill	2024-07-01	450.00	uploads/electric_bill_july.jpeg
TD3_02	Rent	2024-07-03	"Three Thousand"	uploads/rent_july.jpeg
TD3_03	Food Stock	2024-07-06	1500.00	

Test Data for Shed Management:

Table 6.12 Test Data for Shed Management

Test Data ID	Shed Name	Description
TD4_01	Shed1	Main storage shed
TD4_02	Shed2	Dairy cows only
TD4_03	Shed1	Temporary shed

6.5 Test Results and Analysis

Customer Management Module:

Table 6.13 Test Result Customer Management Module

Test Case ID	Test Data ID	Expected Result	Actual Result	Pass/Fail
TC1_01	TD1_01	Customer added successfully. Display the new customer data.	As expected	Pass
TC1_02	TD1_02	Customer not added. Display message "Please fill out this field."	As expected	Pass

Sales Management Module:

Table 6.14 Test Result Sales Management Module

Test Case ID	**Test Data ID**	**Expected Result**	**Actual Result**	**Pass/Fail**
TC10_01	TD10_01	Financial report updated successfully in total sales	As expected	Pass
TC10_02	TD10_02	Report display in total sales is empty.	As expected	Pass

Livestock Management Module:

Table 6.15 Test Result Livestock Management Module

Test Case ID	Test Data ID	Expected Result	Actual Result	Pass/Fail
TC2_01	TD2_01	Livestock record added successfully.	As expected	Pass
TC2_02	TD2_02	Livestock record not added. Display message "Please fill out this field."	As expected	Pass

Expense Management Module:

Table 6.16 Test Result Expense Management Module

Test Case ID	Test Data ID	Expected Result	Actual Result	Pass/Fail
TC3_01	TD3_01	Expense recorded successfully. Display message "Expense recorded successfully."	As expected	Pass
TC3_02	TD3_02	Expense not recorded. Display message "Please enter a valid number for total price."	As expected	Pass
TC3_03	TD3_03	Expense not recorded. Display message "Proof of expense is required."	As expected	Pass

Shed Management Module:

Table 6.17 Test Result Shed Management Module

Test Case ID	Test Data ID	Expected Result	Actual Result	Pass/Fail
TC4_01	TD4_01	Shed added successfully. Display message "Shed added successfully."	As expected	Pass
TC4_02	TD4_02	Shed not added. Display message "Shed name is required."	As expected	Pass
TC4_03	TD4_03	Shed not added. Display message "Shed name already exists."	As expected	Pass

6.5.1 User Acceptance Testing (UAT)

This section describes the User Acceptance Test (UAT) conducted for the "Farm Management System." UAT is the final phase of testing where real users of the system verify that the developed system meets their needs and is ready for deployment.

List of UAT Questionnaire:

1. How easy was it to navigate the Farm Management System interface?

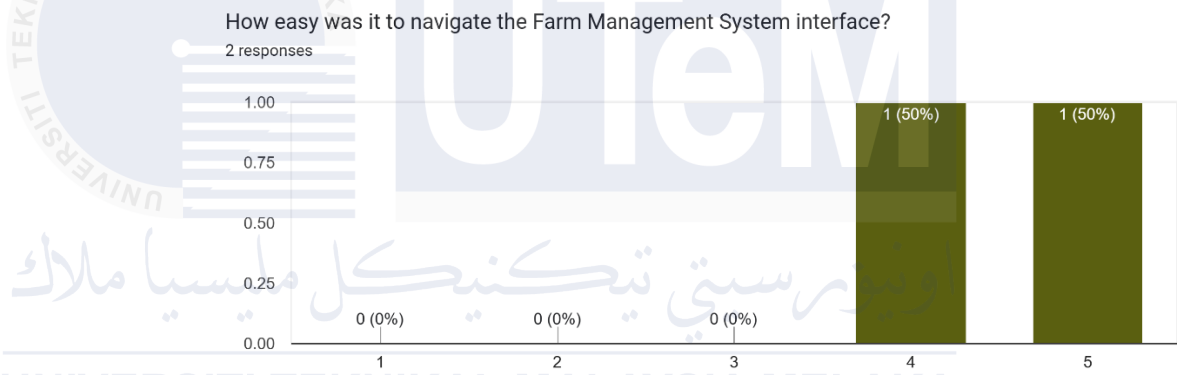

Figure 6.1 Feedback question 1

2. How would you rate the overall user experience of Farm Management System ?

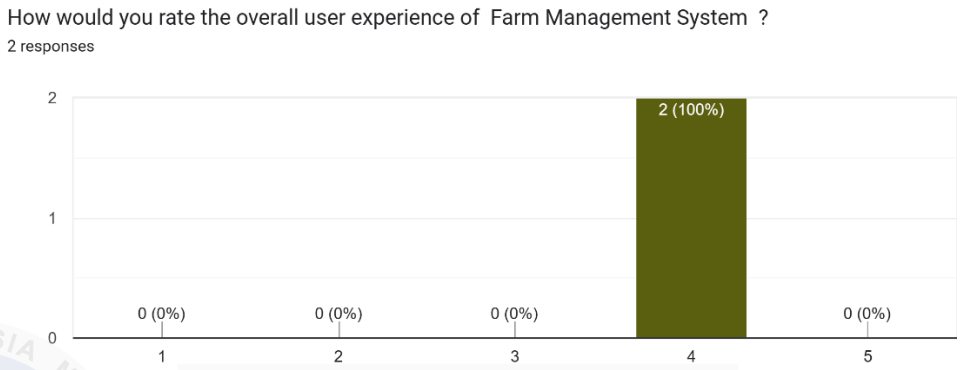

Figure 6.2 Feedback question 2

3. How would you rate the speed and responsiveness of the system?

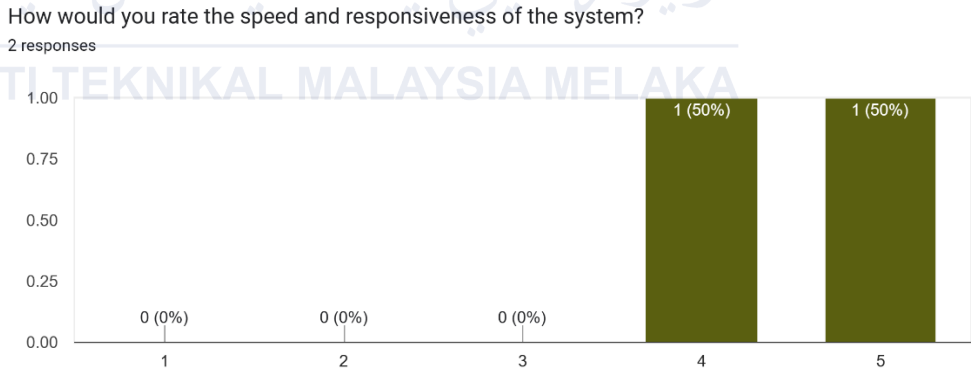

Figure 6.3 Feedback question 3

4. How do you feel about the design and layout of the system?

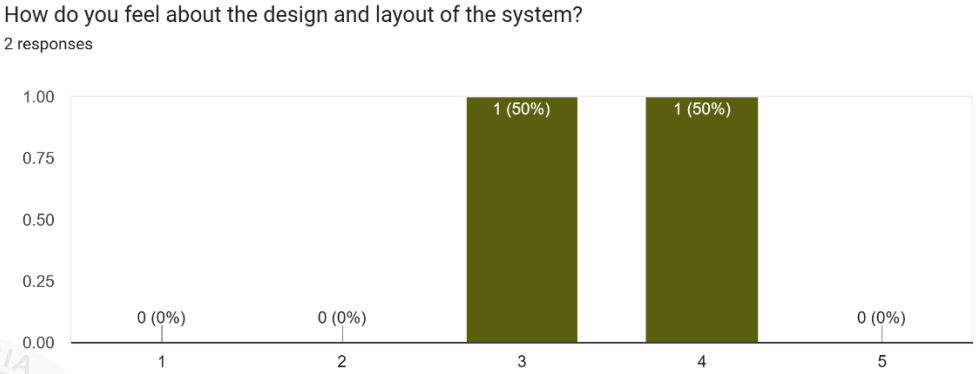

Figure 6.4 Feedback question 4

5. How satisfied are you with your overall experience using the Farm Management System?

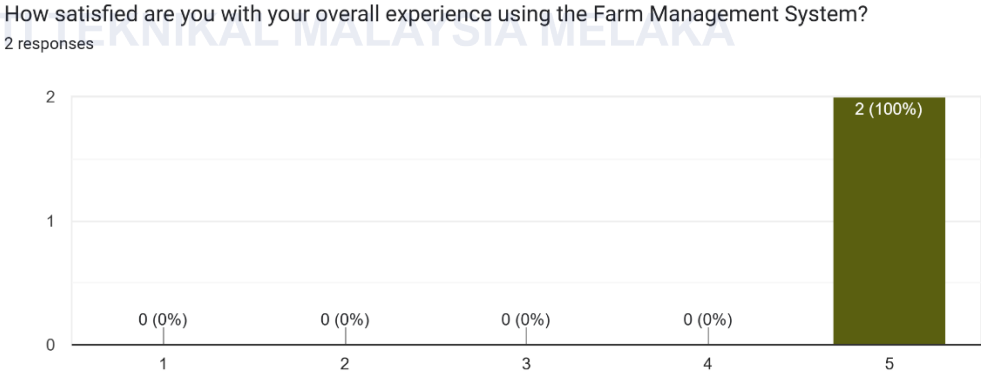

Figure 6.5 Feedback question 5

6. How well do the system's features meet your needs as a user in Admin role?

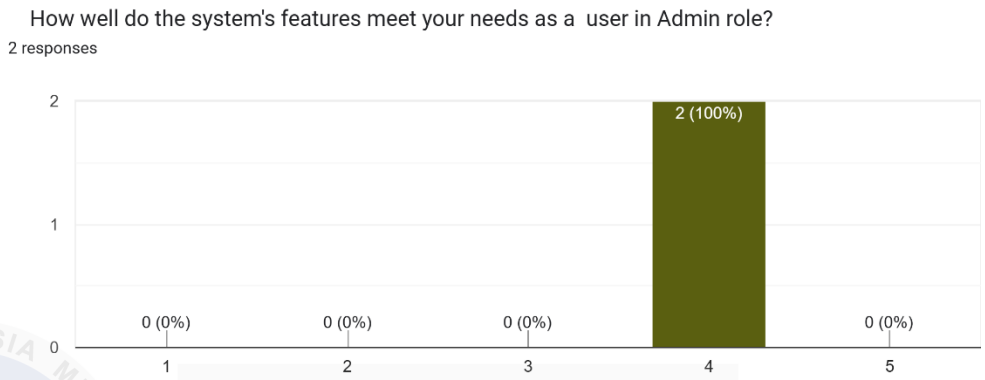

Figure 6.6 Feedback question 6

7. How well do the system's features meet your needs as a user in Worker role?

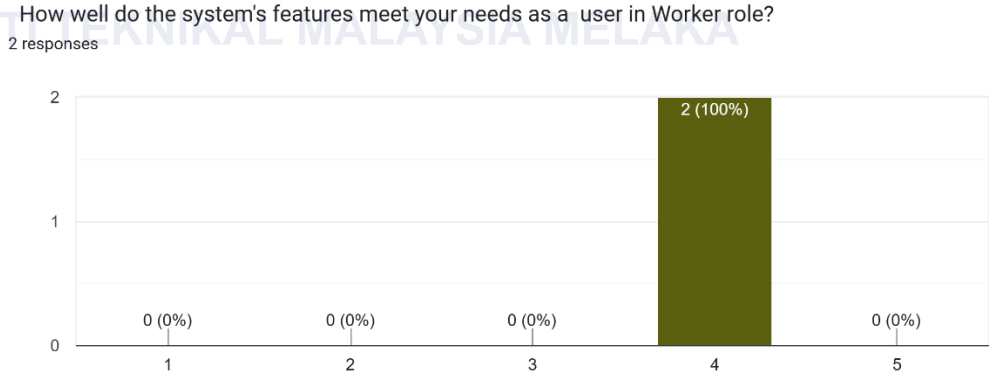

Figure 6.7 Feedback question 7

8. I think using Farm Management System is a valuable tool for managing my farm.

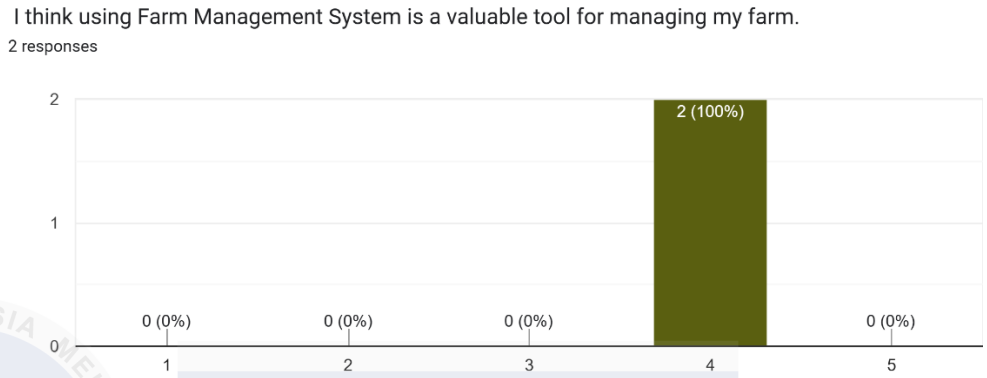

Figure 6.8 Feedback question 8

9. I believe that Farm Management System improves my management experience positively.

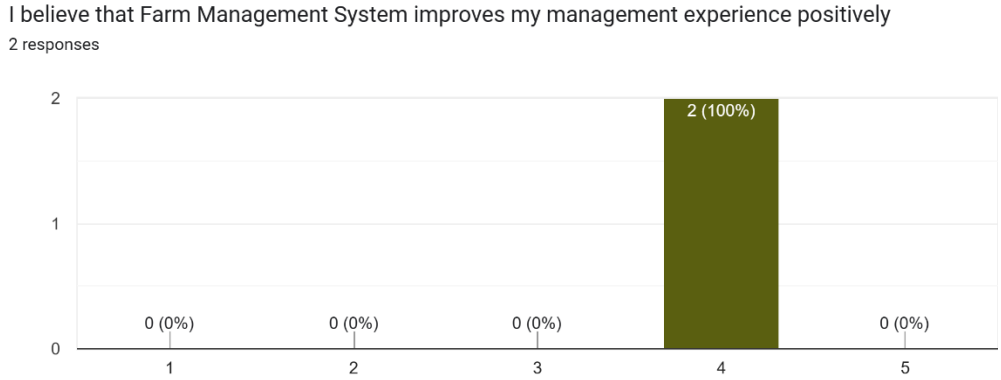

Figure 6.9 Feedback question 9

10. Did you encounter any issues or bugs while using the system? If yes, please describe them.

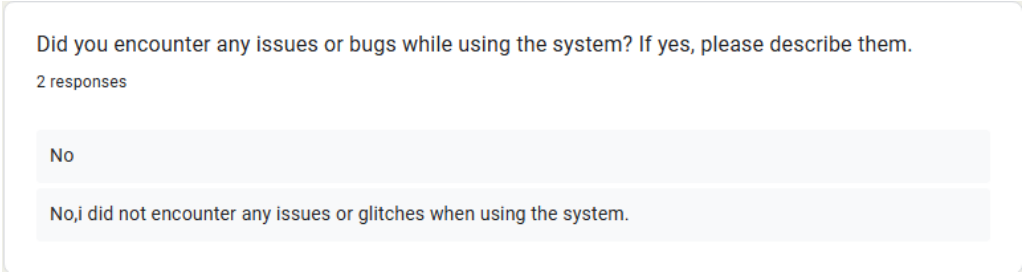

Figure 6.10 Feedback question 10

11. Were there any features that you found particularly useful? Please specify.

Figure 6.11 Feedback question 11

12. Are there any features that you believe should be added or improved? Please specify.

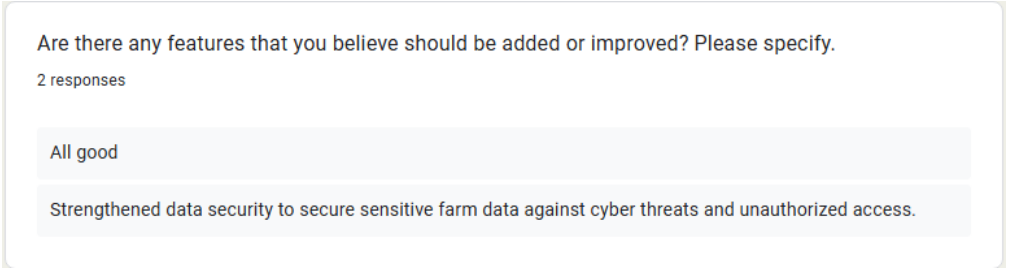

Figure 6.12 Feedback question 12

13. Any additional comments or suggestions for improvement?

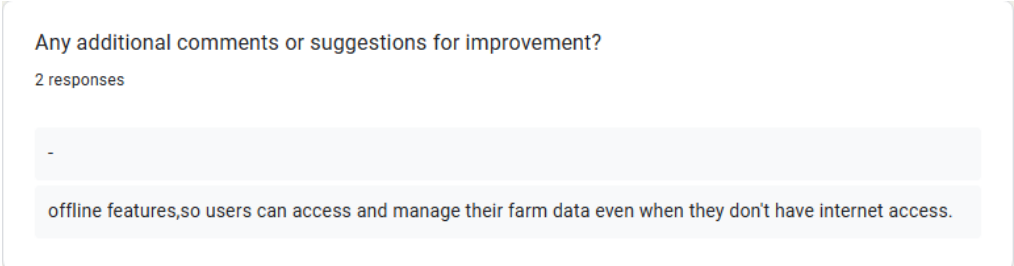

Figure 6.13 Feedback question 13

6.6 Conclusion

In conclusion, the testing phase of the "Farm Management System" successfully validated the system's functionality, performance, and usability. The system passed all test cases, and user feedback during UAT was positive, confirming that the system is ready for deployment. The next steps involve monitoring system performance in a live environment and making any necessary adjustments based on user feedback.

CHAPTER 7: CONCLUSION

7.1 Introduction

7.2 Observations on Weaknesses and Strengths

Strengths:

- Comprehensive Functionality: The Farm Management System offers a wide range of features that address various aspects of farm operations, including livestock management, sales tracking, expense management, and financial reporting. This comprehensive approach ensures that farmers have access to all necessary tools within a single platform.

- User-Friendly Interface: The system is designed with an intuitive and easy-to-navigate interface, making it accessible to users with varying levels of technical expertise. This enhances user adoption and reduces the learning curve.

- Scalability: The system is built to be scalable, allowing for the addition of new modules and handling increased data volumes as the farm's operations expand. This ensures that the system can grow alongside the farm's needs.

- Security: Strong security measures, including user authentication and data encryption, are implemented to protect sensitive information and ensure that only authorized users have access to critical functions.

Weaknesses:

- Limited Customization Options: While the system offers a good set of features, it has limited customization options for farmers with unique operational needs. This may require additional development work to make the system to specific farm setups.

- Initial Setup Complexity: The initial setup of the system, including environment configuration and database initialization, may be challenging for users with limited technical expertise. This could be a barrier to adoption for smaller farms without dedicated IT resources.

7.3 Propositions for Improvement

- Enhanced Customization: Introducing more customizable features within the system would allow farmers to tailor the software to their specific operational needs. This could include customizable dashboards, reports, and workflows that can be adjusted without requiring significant technical intervention.

- Simplified Setup Process: Developing a more streamlined installation and setup process, possibly with automated configuration scripts or a guided setup wizard, would reduce the complexity of getting the system up and running. This would make the system more accessible to smaller farms with limited technical expertise.

- Mobile Accessibility: Implementing a mobile-friendly version of the system or a dedicated mobile app would provide farmers with greater flexibility in managing their operations on the go. This is particularly important for tasks that require real-time data entry and monitoring in the field.

7.4 Contribution

- Centralized Farm Management: The system provides a unified platform that streamlines various farm operations, including livestock management, sales tracking, and financial reporting, reducing the need for multiple disparate tools.

- Data-Driven Decision Making: By integrating data analytics capabilities, the system empowers farmers to make informed decisions, optimizing resource allocation and enhancing overall farm productivity.

- User-Friendly Interface: The system's intuitive design makes it accessible to users with varying levels of technical expertise, improving adoption rates and reducing the learning curve.

- Scalable Solution**:** The system is designed to scale with the farm's needs, accommodating increased data volumes and additional modules as the farm grows.

REFERENCES

(1) MYLembu Sdn. Bhd. (2021, October 14). *MyLembu - No.1 leading cattle farm in Malaysia*. MYLembu. https://mylembu.com/

(2) informa market. (2024, February 19). *Livestock Malaysia 2025*. Livestock Malaysia 2025. https://www.livestockmalaysia.com/

(3) GeeksforGeeks. (2024, June 6). *SQL trigger Student database*. GeeksforGeeks. https://www.geeksforgeeks.org/sql-trigger-student-database/

(4) *W3Schools.com*. (n.d.). https://www.w3schools.com/sql/sql_stored_procedures.asp

(5) Codecademy. (n.d.). *Thinking about errors in your code differently*. Codecademy. https://www.codecademy.com/article/thinking-about-errors-in-your-code-differently

(6) Starling Bank. (n.d.). *What is cost of sales and how is it calculated? - Starling Bank*. https://www.starlingbank.com/resources/business-guides/what-is-cost-of-sales-and-how-is-it-calculated/

(7) *W3Schools.com*. (n.d.-b). https://www.w3schools.com/howto/howto_css_login_form.asp

(8) *What are the most profitable livestock for small farms?* (2024, March 1). Agrilinks. https://agrilinks.org/post/what-are-most-profitable-livestock-small-farms

(9) Farmbrite. (2023, May 2). Top 15 Cattle Breeds for your Farm. *Farmbrite*. https://www.farmbrite.com/post/top-15-types-of-cattle-for-your-farm

(10) *Font awesome*. (n.d.). Font Awesome. https://fontawesome.com/