



Faculty of Computer Science and Information Technology

**PoultryPlus:
Systematic System for Inventory and Health Tracking**

Tan Zi Siah

**Bachelor of Computer Science with Honours
(Computational Science)**

2025

**POULTRYPLUS:
SYSTEMATIC SYSTEM FOR INVENTORY AND HEALTH TRACKING**

TAN ZI SIAH

This project is submitted in partial fulfilment of the
requirements for the degree of
Bachelor of Computer Science with Honours

Faculty of Computer Science and Information Technology
UNIVERSITI MALAYSIA SARAWAK

2025

**POULTRYPLUS:
SYSTEMATIC SYSTEM FOR INVENTORY AND HEALTH TRACKING**

TAN ZI SIAH

Projek ini merupakan salah satu keperluan untuk Ijazah
Sarjana Muda Sains Komputer dan Teknologi Maklumat

Faculty of Computer Science and Information Technology
UNIVERSITI MALAYSIA SARAWAK

2025

CERTIFICATION OF ORIGINALITY

I hereby declare that this project together with all its contents is no other than those of my own work, except for some information taken and extracted from other resources that have been quoted respectively.



.....
Tan Zi Siah
81355
Bachelor of Computer Science with Honors
(Computational Science)
Universiti Malaysia Sarawak

Date: 4/2/2025

ACKNOWLEDGEMENT

First and foremost, I would like to express my heartfelt gratitude to my supervisor, Dr. Lee Nung Kion, for his invaluable guidance, unwavering support, and insightful advice throughout the development of this project. His expertise and encouragement have been instrumental in shaping the direction and success of this work.

I extend my sincere thanks to Unigold Sdn Bhd for providing the opportunity and resources necessary to undertake this project. I am grateful to the team at Unigold's poultry farms for their cooperation, particularly in sharing operational insights and data that were crucial for the system's design and implementation.

I would also like to acknowledge my family and friends for their constant encouragement, understanding, and motivation throughout this journey. Their unwavering support has been a source of strength during challenging moments.

Finally, I wish to thank everyone who directly or indirectly contributed to the successful completion of this project. Your efforts and contributions are deeply appreciated, and this accomplishment would not have been possible without your support.

This project marks a significant milestone in my academic journey, and I am truly grateful for the learning opportunities it has provided.

TABLE OF CONTENTS

Chapter 1 : INTRODUCTION	3
1.1 Introduction	3
1.2 Problem Statement	3
1.3 Objectives	4
1.4 Brief Methodologies	7
1.5 Scope of Project	7
1.6 Significant of the Project	8
1.7 Project Schedule	10
1.8 Expected Outcome	11
1.9 Outline of Project	12
1.9.1 Chapter 2: Literature Review	12
1.9.2 Chapter 3: Requirement Analysis and Design	12
1.9.3 Chapter 4: System Development and Implementation	12
1.9.4 Chapter 5: Testing and Evaluation	13
1.9.5 Chapter 6: Conclusion and Future Work	13
Chapter 2 : LITERATURE REVIEW	14
2.1 Introduction	14
2.2 Background Study	14
2.3 General Concept of Poultry Broiler Management System	15
2.4 Existing Poultry Management System	17
2.4.1 BroilerX App	17
2.4.2 NAVFarm	19
2.4.3 Farmbrite	21
2.5 Comparison Features of Existing Systems	24
2.6 Proposed Solution	27
2.7 Summary	27
Chapter 3 : REQUIREMENT ANALYSIS AND DESIGN	29
3.1 Introduction	29
3.2 Methodology	29
3.2.1 Requirement Collection and Analysis	29
3.2.2 System Design	31
3.2.3 System Development	31

3.2.4 Testing	32
3.3 Requirement Collection and Analysis	32
3.3.1 Identify Users and Characteristics	32
3.3.2. Functional Requirements	36
3.3.3. Non-Functional Requirements	37
3.3.4. Software Requirement	39
3.3.5. Hardware Requirement	40
3.4 System Design	41
3.4.1. Use Case Diagram	41
3.4.2. Activity Diagram	56
3.4.3. Sequence Diagram	60
3.4.4. Data Dictionary	65
3.4.5. Entity Relationship Diagram (ERD)	73
3.4.6. Graphical User Interface Design	74
3.5 Summary	76
Chapter 4 : SYSTEM IMPLEMENTATION	78
4.1 Introduction	78
4.2 System Implementation	78
4.2.1. Installation	78
4.2.2. Project File Structure	80
4.2.3. Coding — Front-to-Back Data Flow	81
4.2.4. Implementation of PoultryPlus	84
4.2.4.1 Authentication Module	84
4.2.4.2 Dashboard Module	87
4.2.4.3 Users Module	88
4.2.4.4 Setup Module	89
4.2.4.5 Alert Module	95
4.2.4.6 Daily Record Module	97
4.2.4.7 Feed Inventory Module	100
4.2.4.8 Vaccine Inventory Module	101
4.2.4.9 Report Module	102
4.3 Database Implementation	103
4.3.1. Django Models and Migrations	103

4.3.2. Django Admin Panel	104
4.3.3. Database Setup for Production	105
4.4 Summary	105
Chapter 5 SYSTEM TESTING	107
5.1 Introduction	107
5.2 Functionality Testing	107
5.3 Usability Testing	128
5.4 Summary	131
Chapter 6 CONCLUSION AND FUTURE WORK	133
6.1 Introduction	133
6.2 Project Achievement	133
6.3 Project Limitation	133
6.4 Future Work	134
6.5 Conclusion	135

List of Figures

Figure 1.1 Overview of Agile Methodology by Sajna (2022)	7
Figure 1.2 Gantt Chart of the Project Schedule	10
Figure 2.1 Flowchart of data input steps, management approach, and flock evaluation	15
Figure 2.2 Overview of BroilerX App Features	18
Figure 2.3 NAVFarm Main Page	20
Figure 2.4 NAVFarm Report Overview	21
Figure 2.5 Farmbrite Interface for Tracking Weight and FCR	22
Figure 3.1 Overview of Agile Methodology by Sajna (2022)	29
Figure 3.2 Current Data Recording Form	34
Figure 3.3 Example of Data Recording	34
Figure 3.4 Weight Record of Catch Chicken	35
Figure 3.5 Current Workflow for Unigold	36
Figure 3.6 Overview of PoultryPlus System with Use Case Scenario	41
Figure 3.7 Use Case Scenario 1: User Authentication	42
Figure 3.8 Use Case Scenario 2: Manipulate Farm Information	44
Figure 3.9 Use Case Scenario 3: Add New Vaccine and Feed	46
Figure 3.10 Use Case Scenario 4: Record Chicks Arrival	48
Figure 3.11 Use Case Scenario 5: Chicken Management Activities	50
Figure 3.12 Use Case Scenario 6: View Poultry Data Metrics	52
Figure 3.13 Use Case Scenario 7: Track Inventory	54
Figure 3.14 Use Case Scenario 8: Generate Report	55
Figure 3.15 Activity Diagram for Use Case 1: User Authentication	57
Figure 3.16 Activity Diagram for Use Case 2: Manipulate Farm Information	57
Figure 3.17 Activity Diagram for Use Case 3: Add New Vaccine and Feed	57
Figure 3.18 Activity Diagram for Use Case 4: Record Chick Arrival	58
Figure 3.19 Activity Diagram for Use Case 5: Chicken Management Activities	59
Figure 3.20 Activity Diagram for Use Case 6: View Poultry Data Metrics	59
Figure 3.21 Activity Diagram for Use Case 7: Track Inventory	59
Figure 3.22 Activity Diagram for Use Case 8: Generate Report	60
Figure 3.23 Sequence Diagram for Use Case 1: User Authentication	60
Figure 3.24 Sequence Diagram for Use Case 2: Manipulate Farm Information	61
Figure 3.25 Sequence Diagram for Use Case 3: Add New Vaccine and Feed	62

Figure 3.26 Sequence Diagram for Use Case 4: Record Chick Arrival	62
Figure 3.27 Sequence Diagram for Use Case 5: Chicken Management Activities	63
Figure 3.28 Sequence Diagram for Use Case 6: View Poultry Data Metrics	63
Figure 3.29 Sequence Diagram for Use Case 7: Track Inventory	64
Figure 3.30 Sequence Diagram for Use Case 8: Generate Report	64
Figure 3.31 ER Diagram for PoultryPlus System	74
Figure 3.32 GUI Design 1: Register Page	75
Figure 3.33 GUI Design 2: Login Page	75
Figure 3.34 GUI Design 3: Dashboard	76
Figure 4.1 Django ORM and REST API data flow (Adapted from Garner, 2021)	79
Figure 4.2 Backend File Structure as seen in Visual Studio Code	80
Figure 4.3 Frontend File Structure as seen in Visual Studio Code	80
Figure 4.4 Frontend Design File Structure as seen in Visual Studio Code	81
Figure 4.5 React <Register> component code	82
Figure 4.6 Users App urls.py	82
Figure 4.7 Response Handling in Register.js	83
Figure 4.8 Sequence Diagram for The Registration Flow	83
Figure 4.9 Login Frontend Code Snippet (React)	85
Figure 4.10 Login Page	85
Figure 4.11 Register Page Input Form	86
Figure 4.12 PoultryPlus Dashboard	87
Figure 4.13 User profile interface showing edit/view modes	88
Figure 4.14 Manager view with worker list and password reset controls	89
Figure 4.15 Farm Listing Page	90
Figure 4.16 Farm Modal (Add/Edit)	91
Figure 4.17 Delete Confirmation Alert	92
Figure 4.18 Shed List Page	93
Figure 4.19 Feed Type Page	93
Figure 4.20 Vaccine Type Page	94
Figure 4.21 Chicks and Mortality Page	95
Figure 4.22 Standard Data saved in the system	96
Figure 4.23 Alert Thresholds Management	96
Figure 4.24 Active Shed	97

Figure 4.25 Daily Flock Record	98
Figure 4.26 Modal Form to Input Data	98
Figure 4.27 FCR Performance Dashboard	99
Figure 4.28 Feed inventory interface with calendar and transaction log	100
Figure 4.29 Feed Balance Summary	100
Figure 4.30 Vaccine inventory interface with calendar and transaction log	101
Figure 4.31 Vaccine Balance Summary	101
Figure 4.32 Report Interface with Filter Controls and Tab Navigation	102
Figure 4.33 Flock Performance Charts	103
Figure 4.34 Example of a Django Model Definition	104
Figure 4.35 Example of a Django Admin Registration	104
Figure 4.36 Django Admin Interface Showing Flock Table	104
Figure 4.37 Database Configuration Architecture	105

List of Tables

Table 1 Comparison between Existing System and Proposed PoultryPlus System	24
Table 2 Comparison Pros and Cons among Existing System	25
Table 3 Specification of PoultryPlus Software Requirements	39
Table 4 Specification of PoultryPlus Hardware Requirements	40
Table 5 Data Dictionary for Farm Worker	1
Table 6 Data Dictionary for Farm	1
Table 7 Data Dictionary for Shed	1
Table 8 Data Dictionary for Flock	1
Table 9 Data Dictionary for Chick Type	1
Table 10 Data Dictionary for Feed	1
Table 11 Data Dictionary for Feed Record	1
Table 12 Data Dictionary for Vaccine	1
Table 13 Data Dictionary for Vaccine Record	1
Table 14 Data Dictionary for Weight Record	1
Table 15 Data Dictionary for FCR Record	1
Table 16 Data Dictionary for Mortality Record	1
Table 17 Data Dictionary for Feed Inventory	1
Table 18 Data Dictionary for Vaccine Inventory	1
Table 19 Data Dictionary for Alert Threshold	1
Table 20 Data Dictionary for Mortality Reason	1
Table 21 Django Backend Installation	1
Table 22 React Frontend Installation	1
Table 23 Test Case for Login Module	1
Table 24 Test Case for Farm Module	1
Table 25 Test Case for Shed Module	1
Table 26 Test Case for Feed Module	1
Table 27 Test Case for Vaccine Module	1
Table 28 Test Case for Chicks Module	1
Table 29 Test Case for Mortality Module	1
Table 30 Test Case for Alert Module	1
Table 31 Test Case for Flock Module	1
Table 32 Test Case for Daily Flock Activity Module	1

Table 33 Test Case for User Module	1
Table 34 Test Case for Report Module	1
Table 35 Test Case for Feed Inventory Module	1
Table 36 Test Case for Vaccine Inventory Module	1
Table 37 Test Scenario 1	1
Table 38 Test Scenario 2	1
Table 39 Limitations of PoultryPlus	1

List of Abbreviations

UNIMAS	Universiti Malaysia Sarawak
FCR	Feed Conversion Ratios
KPI	Key Performance Indicator
SDLC	Software Development Life Cycle
SQL	Structured Query Language
API	Application Programming Interface
UAT	User acceptance testing
UML	Unified Modeling Language
BMS	Broiler Management System
DOC	Day-Old Chicks
ORM	Object-Relational Mapping
JWT	JSON Web Token

Abstract

The poultry industry plays a critical role in meeting the global demand for protein, with broiler farms focusing on maximizing productivity, minimizing costs, and ensuring flock health. However, many mid-sized farms, including Unigold Sdn Bhd, still rely on manual record-keeping methods, which lead to inefficiencies, data inaccuracies, and challenges in proactive decision-making. This project aims to develop a digital inventory and mortality tracking system, PoultryPlus, specifically tailored to the operational needs of Unigold's poultry farms. The system, designed using an Agile Software Development Life Cycle (SDLC) methodology, integrates a React-based frontend, a Django backend, and a SQL database to create a user-friendly, scalable, and secure solution. Key functionalities include real-time tracking of critical performance indicators such as Feed Conversion Ratio (FCR) and mortality rates, streamlined record-keeping for chick arrivals, feed usage, and vaccination schedules, and robust data analytics to support proactive resource planning and decision-making. By replacing manual processes with a centralized digital platform, PoultryPlus aims to enhance operational efficiency, improve data accuracy, and empower Unigold's management and workers with actionable insights. Beyond addressing the challenges faced by Unigold, this system serves as a model for mid-sized broiler farms seeking to adopt technology for sustainable poultry farming practices.

Abstrak

Industri ternakan ayam memainkan peranan penting dalam memenuhi permintaan global terhadap protein, dengan ladang ayam daging memberi tumpuan kepada memaksimumkan produktiviti, mengurangkan kos, dan memastikan kesihatan ternakan. Namun, banyak ladang bersaiz sederhana, termasuk Unigold Sdn Bhd, masih bergantung kepada kaedah penyimpanan rekod manual, yang membawa kepada ketidakcekapan, ketidaktepatan data, dan cabaran dalam membuat keputusan secara proaktif. Projek ini bertujuan untuk membangunkan sistem penjejakan inventori dan kadar kematian secara digital, PoultryPlus, yang direka khusus untuk memenuhi keperluan operasi ladang ternakan ayam Unigold. Sistem ini, yang dibangunkan menggunakan metodologi Kitar Hayat Pembangunan Perisian (SDLC) Agile, mengintegrasikan frontend berasaskan React, backend Django, dan pangkalan data SQL untuk mencipta penyelesaian yang mesra pengguna, boleh diskalakan, dan selamat. Fungsi utama termasuk penjejakan masa nyata untuk penunjuk prestasi kritikal seperti nisbah penukaran makanan (FCR) dan kadar kematian, penyimpanan rekod yang diselaraskan untuk ketibaan anak ayam, penggunaan makanan, dan jadual vaksinasi, serta analitik data yang kukuh untuk menyokong perancangan sumber dan pembuatan keputusan secara proaktif. Dengan menggantikan proses manual dengan platform digital berpusat, PoultryPlus bertujuan untuk meningkatkan kecekapan operasi, meningkatkan ketepatan data, dan memperkasakan pengurusan dan pekerja Unigold dengan pandangan yang boleh diambil tindakan. Selain menangani cabaran yang dihadapi oleh Unigold, sistem ini berfungsi sebagai model untuk ladang ayam daging bersaiz sederhana yang ingin menggunakan teknologi untuk amalan penternakan ayam yang mampan.

Chapter 1 : INTRODUCTION

1.1 Introduction

The poultry industry is essential for meeting global protein demand, with broiler farms playing a critical role by raising chickens specifically for meat production. These farms focus on maximizing growth, minimizing feed costs, and ensuring flock health. Key performance indicators (KPIs) such as feed conversion ratio (FCR) and mortality rate are vital for measuring efficiency and sustainability. FCR tracks how effectively chickens convert feed into body weight, while mortality rates indicate flock health and management effectiveness.

Despite their importance, many mid-sized broiler farms still rely on manual data recording, which is prone to errors and inefficiencies. This limits the ability to accurately monitor KPIs and make informed decisions. Although digital management systems have advanced in agriculture, they often remain inaccessible or unsuitable for smaller broiler farms (Dhillon & Moncur, 2023).

This project aims to bridge this gap by developing an accessible, user-friendly digital solution tailored to broiler farming. By improving data accuracy and providing real-time insights, the system seeks to enhance operational efficiency and support sustainable poultry management.

1.2 Problem Statement

At Unigold's poultry broiler farms, crucial operational data—including chick arrivals, daily mortality rates, feed consumption, and vaccination schedules—are recorded manually. This paper-based approach leads to frequent data loss, delays in tracking inventory and performance indicators, and reactive supply ordering, which disrupts farm productivity. Current agricultural management software often fails to meet the needs of mid-sized poultry

farms; many available systems are either too generic, lacking poultry-specific features, or too complex for practical use on farms like Unigold's. As a result, Unigold struggles with inefficient data management, which hinders effective decision-making and limits the farm's ability to proactively manage resources and monitor flock health. This project seeks to address these issues by developing a user-friendly, web-based poultry management system tailored to Unigold's operational needs, aiming to improve data accuracy, track KPIs in real-time, and enhance overall farm efficiency.

1.3 Objectives

1. Develop a Poultry Management System

The objective is to create a comprehensive web-based application specifically tailored to the operational needs of Unigold Sdn Bhd. This system will enable farm workers to efficiently record and manage essential data, including:

- **Chick Arrivals:** Streamlining the entry process for new chicks to ensure accurate tracking from day one.
- **Mortality Rates:** Implementing a mechanism for timely recording of daily mortality, allowing for immediate analysis of flock health.
- **Feed Usage:** Tracking feed consumption to assess efficiency and optimize feed management strategies.
- **Vaccination Schedules:** Maintaining detailed records of vaccination activities to ensure compliance with health protocols and improve flock immunity.

The focus will be on user experience, ensuring that the interface is intuitive and accessible for all workers, regardless of their technical expertise.

2. Implement Secure and Efficient Data Processing

This objective involves utilizing Django to build a robust and secure back-end infrastructure.

Key features will include:

- **Scalable RESTful API:** Developing an API that supports seamless data processing, storage, and retrieval, ensuring that the system can handle increasing volumes of data as the farm grows.
- **Data Security:** Implementing stringent security measures to protect sensitive operational data from unauthorized access and breaches.
- **Real-Time Processing:** Ensuring that data entries and updates are processed in real-time, allowing for immediate access to the latest information and quick response to emerging issues.

This will enhance the reliability of data used for decision-making and operational planning.

3. Enable Real-Time Tracking of Key Performance Indicators (KPIs)

The goal is to facilitate the real-time tracking and analysis of critical KPIs essential for farm management, including:

- **Feed Conversion Ratio (FCR):** Monitoring FCR to evaluate the efficiency with which feed is converted into weight gain, helping to identify areas for improvement in feeding strategies.
- **Mortality Rates:** Analysing mortality data to detect trends and potential health issues within the flock, enabling timely interventions.
- **Data Analytics Integration:** Integrating analytical tools that provide actionable insights based on real-time data, empowering management to make informed decisions about resource allocation, health management, and operational adjustments.

By focusing on real-time analytics, the system will support proactive decision-making, ultimately enhancing productivity and sustainability in poultry farming practices.

1.4 Brief Methodologies

This project adopts an Agile Software Development Life Cycle (SDLC) methodology to ensure flexibility, iterative progress, and regular feedback. Agile is particularly suitable for projects where user requirements evolve, and quick feedback is essential to success (Schwaber & Beedle, 2002). The process begins with requirement gathering through interviews and surveys to identify key metrics and essential features. The system design involves creating a user-friendly frontend using React, a secure backend with Django, and a SQL database for operational data. Development phases include building the frontend and backend, followed by integration via a RESTful API. Finally, comprehensive testing—including unit, integration, and user acceptance testing—ensures the system meets user needs and functions effectively. The detail explanation of the methodology is discussed in the Chapter 3.

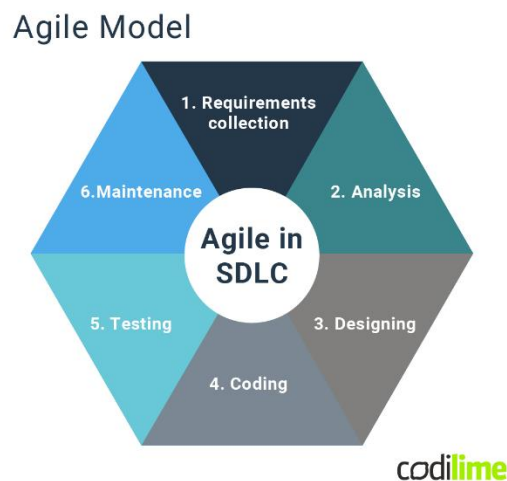


Figure 1.1 Overview of Agile Methodology by Sajna (2022)

1.5 Scope of Project

The project aims to develop a digital inventory and mortality tracking system for Unigold's poultry farms (Farms C, E, and G). This system will replace manual record-keeping with a

streamlined digital solution, enhancing efficiency and data accuracy. Key features will include a user-friendly web interface using React, data entry forms for logging chick arrivals, mortality rates, and feed consumption, as well as basic tracking of KPIs like FCR. Secure data storage will be implemented using Django and MySQL, along with simple report generation for insights into inventory levels. User acceptance testing (UAT) will ensure the system meets the needs of farmers and workers.

To maintain focus, certain features will be excluded from this phase, such as advanced analytics, mobile application development, and integration with external systems. The project will operate within a specified timeframe, delivering essential functionalities first and allowing for future enhancements based on feedback from farmers and workers. The immediate goal is to deliver a functional web application that improves data recording and tracking processes, effectively supporting Unigold's poultry operations.

1.6 Significant of the Project

The development of a digital inventory and mortality tracking system for Unigold's poultry farms is significant for several reasons. First, it directly addresses inefficiencies in the current manual record-keeping practices, enhancing operational efficiency and improving the accuracy of tracking KPIs like FCR and mortality rates. This digital solution benefits Unigold and at the same time serves as a model for other mid-sized poultry farms facing similar challenges.

Academically, the project fulfils degree requirements by applying theoretical knowledge in software development, database management, and user experience design to a real-world context. It refines existing knowledge on digital solutions in agriculture, demonstrating how tailored systems can improve data management in poultry operations.

Additionally, the findings could inform educational programs in agricultural management by suggesting curriculum enhancements incorporate digital tools. The results may also influence policy decisions related to technology adoption in the agricultural sector, advocating for digital solutions to boost productivity and sustainability.

In summary, this project aims to solve practical problems at Unigold's farms and contribute to scholarly research, inform educational practices, and influence policy decisions in the poultry industry, leading to improved management practices across the sector.

1.7 Project Schedule

The project schedule outlines a phased approach from November 2024 to June 2025, covering key stages to develop a digital inventory and mortality tracking system for Unigold's poultry farms. It begins with data collection and system design, followed by iterative software development in sprints. The schedule concludes with system evaluation, user feedback integration, and final documentation, ensuring the project meets the farm's operational needs effectively.

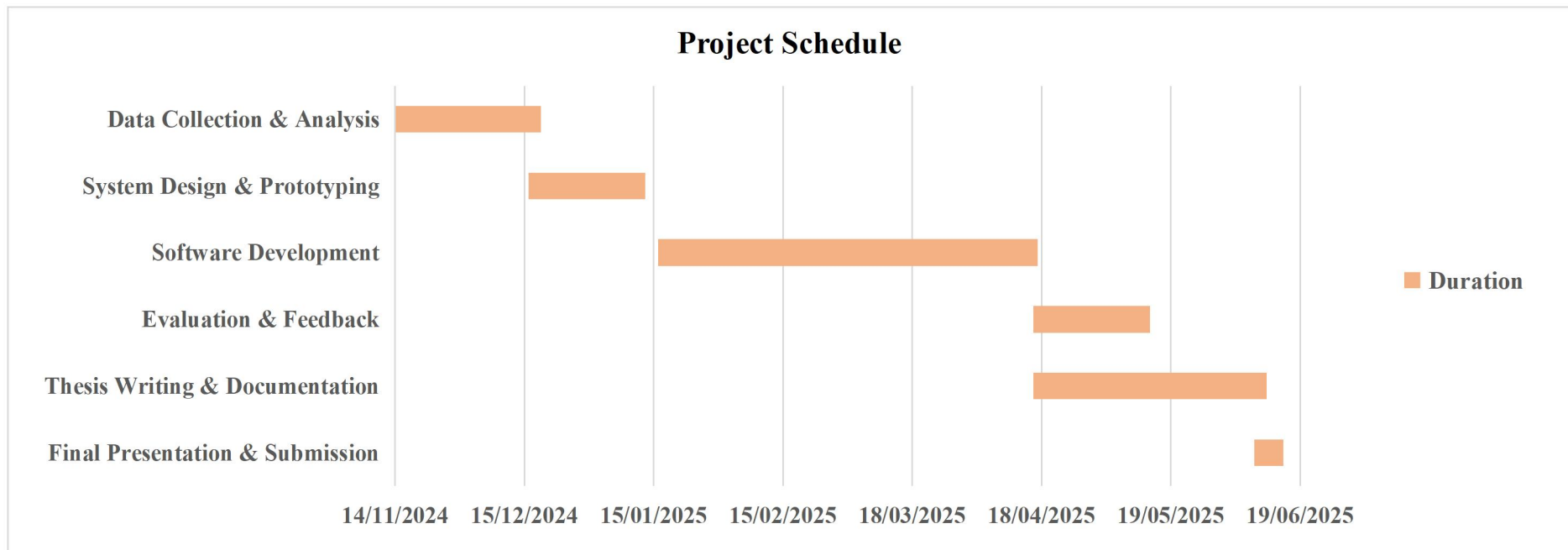


Figure 1.2 Gantt Chart of the Project Schedule

1.8 Expected Outcome

Upon completion, this project will deliver a functional, user-friendly digital management system that replaces Unigold's manual record-keeping processes with an efficient, web-based solution. The system will empower farm workers to easily record and access essential operational data—such as chick arrivals, mortality rates, feed usage, and vaccination schedules—through an intuitive interface.

Expected outcomes include:

1. **Enhanced Data Accuracy:** Centralizing records digitally will reduce errors in data entry and minimize the risk of data loss.
2. **Improved Decision-Making:** Real-time tracking of KPIs will enable farm managers to make proactive, data-driven decisions.
3. **Operational Efficiency:** Streamlined record-keeping and automated calculations will reduce the time spent on administrative tasks, allowing workers to focus on higher-value activities.
4. **Proactive Resource Management:** The system will facilitate predictive resource planning, helping the farm anticipate and fulfill supply needs (e.g., feed and vaccines) in advance, thus preventing last-minute orders.
5. **Scalable Digital Framework:** A flexible backend built with Django and SQL will allow for future enhancements, such as more advanced analytics or mobile accessibility.
6. **Model for Industry Application:** The system will serve as a case study for other mid-sized poultry farms, showcasing the benefits of adopting digital tools for improved productivity and sustainability in the poultry industry.

Overall, this project will improve farm management at Unigold, enhancing productivity and animal welfare through effective data management and real-time insights.

1.9 Outline of Project

The project is categorized into two phases: Final Year Project 1 (FYP 1) and Final Year Project 2 (FYP 2). The development spans one academic year (two semesters) and progresses from project conception to completion. For FYP 1, the project begins on 14th November 2024 and is expected to conclude by 12th January 2025. The comprehensive project task schedule for FYP is detailed in the Gantt chart (Figure 1.2).

1.9.1 Chapter 2: Literature Review

Chapter 2 provides a detailed review of existing digital inventory and poultry management systems, focusing on their strengths, limitations, and relevance to mid-sized farms like Unigold's. The review includes the study of current manual systems used in poultry farms and insights into the challenges faced. Additionally, the chapter discusses KPIs such as FCR and mortality rates, emphasizing their importance in poultry farm management. The chapter concludes with proposed strategies to address the identified gaps.

1.9.2 Chapter 3: Requirement Analysis and Design

Chapter 3 outlines the requirements and design specifications for the PoultryPlus system. It elaborates on the Agile SDLC methodology, highlighting its flexibility in accommodating evolving requirements. Unified Modeling Language (UML) diagrams are used to illustrate the system's architecture, including data flow and functional relationships. This chapter serves as the foundational guide for the system's development.

1.9.3 Chapter 4: System Development and Implementation

Chapter 4 describes the development and implementation of the PoultryPlus system. This includes details of frontend development using React, backend development with Django, and the integration of these components through a RESTful API. The chapter documents the methodologies and coding standards adopted during development, serving as a reference for future enhancements.

1.9.4 Chapter 5: Testing and Evaluation

Chapter 5 focuses on the testing and evaluation process. It includes unit testing, integration testing, and User Acceptance Test (UAT) to ensure the system meets functional and non-functional requirements. Errors and defects are identified and resolved through systematic debugging. The chapter validates the system's usability, accuracy, and efficiency based on user feedback and predefined criteria.

1.9.5 Chapter 6: Conclusion and Future Work

The final chapter summarizes the project's contributions and highlights its significance in improving poultry farm operations. It reflects on the overall development process, emphasizing lessons learned and key achievements. Additionally, the chapter explores potential future enhancements, such as integrating advanced analytics or mobile application functionality, to further improve the system's scalability and impact.

Chapter 2 : LITERATURE REVIEW

2.1 Introduction

Poultry farms have drastically improved recently, with factories turning to digital solutions for operating poultry farms. Broiler chicken farming, for example, requires efficient systems to keep checks on inventory, mortality rates, and productivity. The chapter reviews systems related to poultry management and discussed their characteristics and drawbacks. This analysis serves as the foundation for proposing a solution to help broiler farms, such as Unigold Sdn Bhd, meet the demands of their unique needs.

2.2 Background Study

Traditionally, poultry farms, like countless other businesses, have relied on manual methods, specifically paperwork, to document inventory, death rates, feed consumption, and various operational metrics. These methods experience mistakes, inefficiencies, and loss of data. Technological advancements have created opportunities for farms to automate their record-keeping, enhancing both accuracy and productivity.

One instance of this transformation is Chop Cheng Bee Sdn Bhd (CCB), a poultry distributor supplying live chickens throughout Malaysia and Singapore. Even though CCB was acknowledged as the ‘Outstanding Broiler Farm’ at the Malaysian Livestock Industry Awards 2018, it encountered difficulties as its operations became increasingly complex. Ineffectiveness in gathering field data, overseeing farm activities in real-time, and managing orders impeded their expansion. Moreover, their current software solely handled financial accounts, restricting business insights. Analyzing data manually took a lot of time and was prone to human mistakes.

Through the implementation of a tailored digital platform driven by SAP Business One, CCB successfully digitized every facet of their broiler farm activities, encompassing the care of day-old chicks to their sale. The case of CCB illustrates the advantages of implementing digital tools in poultry farming, emphasizing the necessity for systems that unify different elements of farm management into a single platform.

2.3 General Concept of Poultry Broiler Management System

The Poultry Broiler Management System (BMS) is an all-in-one solution that helps streamline broiler farm operations and increase efficiency. They bring together many key applications such as flock performance, sales, expenses, purchasing, feed manufacturing, feed formulation, accounting, etc. BMS constitutes an essential component of modern poultry farming as technological solutions are increasingly applied to support efficient decision-making processes (Pradeep & Pradeep, 2020).

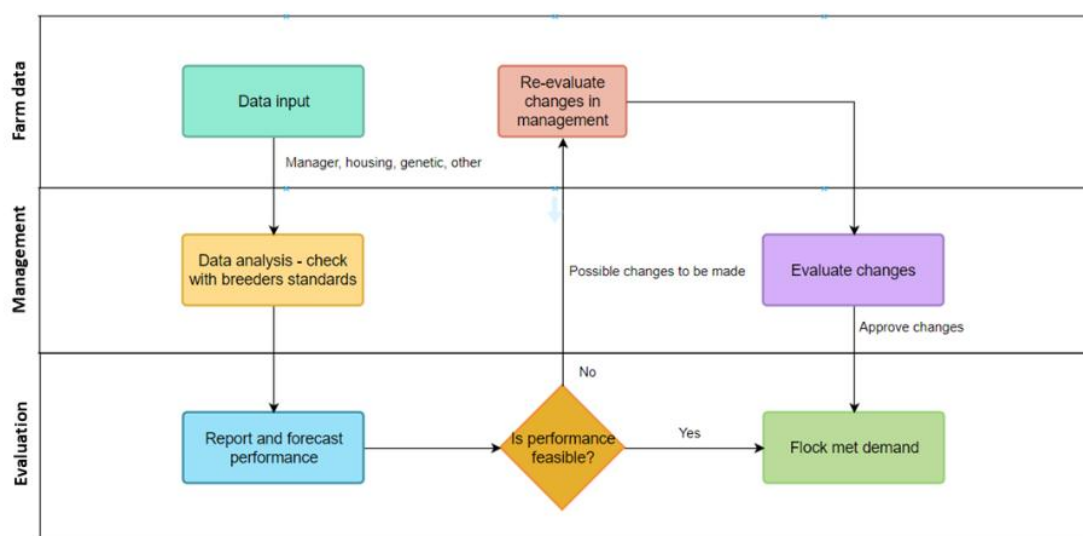


Figure 2.1 Flowchart of data input steps, management approach, and flock evaluation

Key Concepts of a Broiler Management System

1. Flock Management:

- BMS tracks the entire lifecycle of broilers, from housing Day-Old Chicks (DOC) to the sale of mature birds. It includes monitoring daily and weekly depletion reports, mortality rates, and culling records, facilitating accurate performance assessment.

2. Feed and Vaccine Management:

- Feed formulation is optimized based on geographical, climatic, and material availability to ensure cost-effective feeding strategies. Vaccination schedules are meticulously recorded and managed by the flock, helping maintain bird health and mitigate disease outbreaks.

3. Data Analytics and Reporting:

- The system generates detailed reports, including individual flock performance, vaccination records, and weekly depletion statistics. These insights enable farm managers to analyse trends and make informed decisions based on data-driven evidence.

4. Automation and Remote Connectivity:

- BMS enhances operational efficiency with automated feed and vaccination schedules. Systems like Livine offer mobile app connectivity for remote data entry, ensuring seamless operations across geographically dispersed farm locations.

5. Enhanced Decision-Making:

- By integrating real-time data collection and processing, BMS provides tools that improve resource allocation and profitability. It supports broiler farm owners in achieving operational excellence through informed strategic decisions.

According to Pradeep and Pradeep (2020), broiler management systems not only facilitate the recording of farm activities but also serve as powerful decision-making tools. By automating record-keeping and providing actionable insights, these systems significantly enhance productivity, contributing to the overall profitability and efficiency of broiler farms.

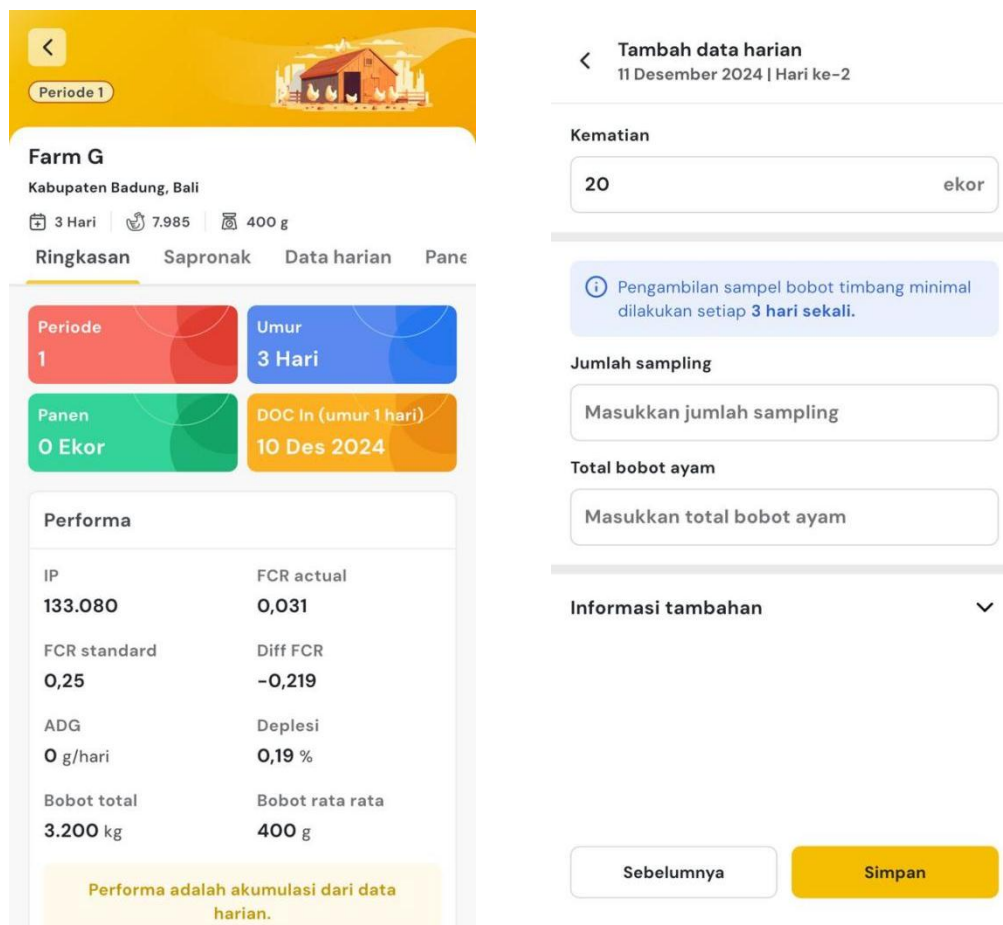
In summary, a well-implemented Poultry Broiler Management System is essential for modern broiler farming, enabling farmers to optimize their operations, manage resources effectively, and enhance the health and productivity of their flocks.

2.4 Existing Poultry Management System

Several poultry management systems have been developed to help farmers streamline their operations, improve productivity, and ensure better control over their farms. Three notable systems are BroilerX, NAVFarm, and Farmbrite, each offering unique features tailored to different aspects of poultry management.

2.4.1 BroilerX App

BroilerX is a poultry farm management platform that uses technology to help poultry farmers increase profits and efficiency (Rohmawan et al., 2024). It is an IoT-based broiler farm monitoring and management system. With this technology, broiler farms will be able to monitor their hens daily and respond quickly to any abnormalities (Salma, 2023).



Features of BroilerX App

Figure 2.2 Overview of BroilerX App Features

Data Management

BroilerX efficiently stores important coop data, cultivation periods, and daily records related to the management of broiler chickens. This feature allows farmers to maintain a comprehensive history of flock performance and environmental conditions.

Production Tracking

The system records daily production metrics, such as weight gain and feed consumption, which are essential for monitoring the overall health and productivity of the flock. This helps farmers identify trends and make timely adjustments to their management practices.

User Interface

BroilerX is designed with a user-friendly interface that simplifies data entry and navigation. This accessibility is particularly beneficial for farm workers who may have limited technical experience.

Health and Treatment Records

The application provides basic tracking of vaccination schedules and health treatments, allowing farmers to ensure timely interventions for disease prevention and management.

2.4.2 NAVFarm

NAVFarm uses the newest technology to automate crucial business processes, providing a creative approach to poultry management. By implementing real-time monitoring of variables such as feed conversion ratio (FCR), mortality rates, and environmental controls, NAVFarm enables farmers to maintain ideal circumstances for chick growth (Navfarm, 2019).

Considering today's world, where high ambient temperature and poor ventilation can considerably affect the performance and health of poultry, NAVFarm has given a wide-based platform to implement effective farm management. At NAVFarm, data integration starts from feed production to animal management down to the level of food distribution, which will ensure that farmers can enhance their productivity and profitability without compromising on quality.

Features of NAVFarm

Real-Time Monitoring

NAVFarm utilizes IoT technology to monitor important parameters like FCR and environmental conditions in real time, storing this data for easy access and analysis.

Integrated Data Management

The platform provides a centralized database for all farm-related data, allowing farmers to track performance across different operations and make informed decisions. Figure 2.3 displays the main page of the NAVFarm system, providing a comprehensive snapshot of critical data related to poultry management. This interface is designed to help farmers effectively monitor and manage their operations in real time.

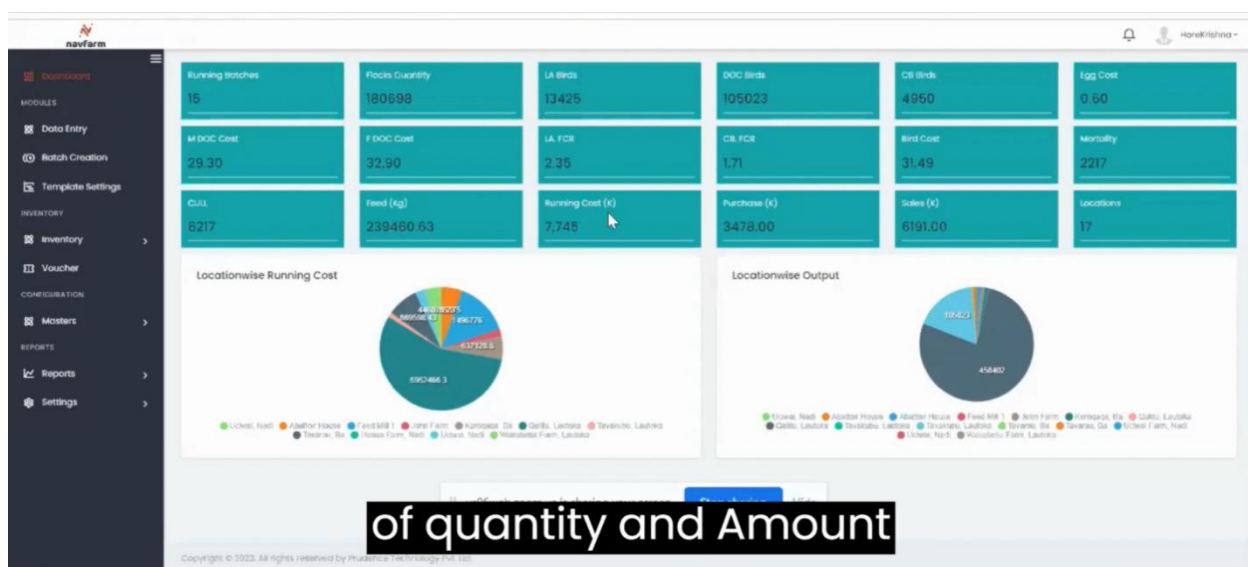


Figure 2.3 NAVFarm Main Page

Analytics and Reporting

NAVFarm generates detailed reports based on stored data, helping farmers analyze production trends and resource usage. Figure 2.5 presents a graphical report generated by the NAVFarm system, illustrating KPIs related to poultry management, including weight, FCR, and other essential data.



Figure 2.4 NAVFarm Report Overview

2.4.3 Farmbrite

Farmbrite was founded in 2012 by Ian and Janine Russell, a husband-and-wife team who developed the software based on their own experiences managing a small farm. Their goal was to create a comprehensive farm management solution that addresses the real-world challenges faced by farmers. Today, Farmbrite helps users optimize their operations, improve productivity, and enhance profitability through its user-friendly platform and integrated features.

Features of Farmbrite

Data Management

Farmbrite offers centralized storage for all farm data, including livestock, crop, and financial information. This feature enables farmers to easily access and manage their records from one location.

Production Tracking

The system measures key performance indicators, such as yield and growth rates, helping farmers monitor productivity and make data-driven decisions. Figure 2.3 shows the interface for recording animal weights and feed conversion ratios.

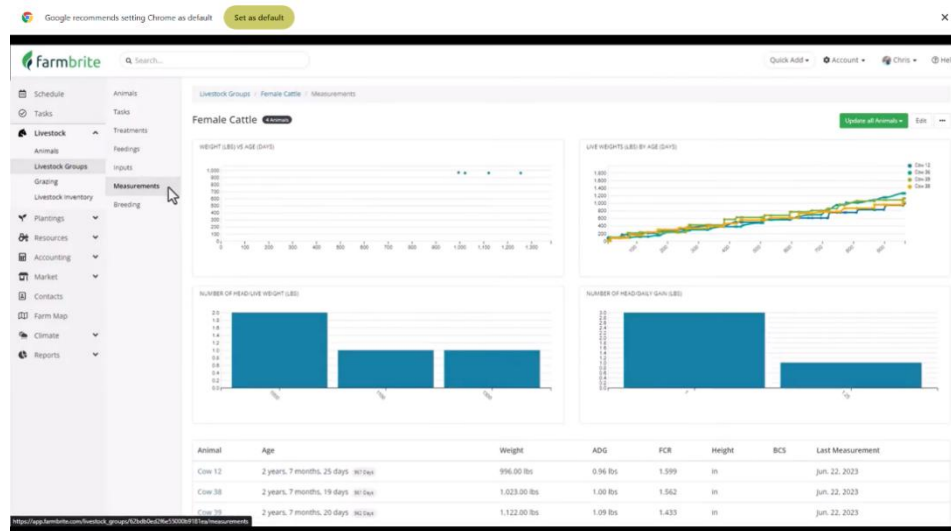


Figure 2.5 Farmbrite Interface for Tracking Weight and FCR

Inventory Management

Farmbrite tracks inventory levels for feed, supplies, and equipment, ensuring that farmers have the necessary resources available when needed.

2.5 Comparison Features of Existing Systems

Table 1 Comparison between Existing System and Proposed PoultryPlus System

Features	BroilerX	NAVFarm	Farmbrite	PoultryPlus (Proposed System)
User Interface	✓	✓	✓	✓
FCR Calculation	✓	✓	✓	✓
Inventory Tracking	✓	✓	✓	✓
Expense Calculation	✓	✓	✓	✓
Report Generation	✗	✓	✓	✓
Comparison with KPIs	✓	✗	✗	✓
Focus on Broilers	✓	✗	✗	✓
Data Accuracy and Security	✗	✓	✓	✓
Proactive Decision-Making	✗	✓	✓	✓
Mortality Track	✓	✓	✓	✓

Based on Table 2.1, PoultryPlus addresses the limitations of existing systems by integrating their strengths while introducing tailored features for broiler farming. Unlike BroilerX, which lacks advanced analytics and report generation, PoultryPlus provides comprehensive reporting and KPI tracking for better decision-making. It also simplifies operations compared to NAVFarm, making it more accessible for smaller farms while maintaining robust data accuracy and security. Additionally, PoultryPlus enhances broiler-specific management, which is less emphasized in Farmbrite, by offering real-time tracking of critical metrics and proactive analytical tools. This makes PoultryPlus a secure, efficient, and specialized solution for modern broiler farm management.

Table 2 Comparison Pros and Cons among Existing System

	Types of Poultry Management System			
Characteristic	BroilerX	NAVFarm	Farmbrite	PoultryPlus (Proposed System)
Advantages	• User-friendly interface for basic farm management. • Specifically tailored for broiler farms.	• Comprehensive interface with advanced data visualization. • Detailed report generation for operational insights.	• Centralized system with broad farm management capabilities. • Offers productivity reports for farm operations.	• Secure and efficient data processing with real-time tracking of KPIs. • Analytical tools for proactive decision-making and KPI comparison with industry standards.
Disadvantages	• Lacks report generation capabilities. • Limited analytics and	• Overwhelming for smaller farms due to feature complexity.	• Less emphasis on optimizing broiler-specific operations.	N/A

	insights for proactive decision-making.	• Not specifically optimized for broiler farming.		
--	---	---	--	--

2.6 Proposed Solution

The proposed PoultryPlus system addresses the unique needs of Unigold Sdn Bhd's broiler farm operations through:

1. **Broiler-Specific Features:** Focused on mortality tracking, feed usage, and vaccine administration tailored to broiler farming.
2. **User-Friendly Interface:** A simple web-based platform for easy data entry and navigation.
3. **Centralized Data Management:** Secure and efficient data storage using Django and SQL, replacing manual records.
4. **Real-Time KPI Monitoring:** Enables tracking of FCR, mortality rates, and other metrics for actionable insights.
5. **Proactive Tools:** Offers analytics for resource optimization and performance comparison against standard KPIs.
6. **Streamlined Focus:** Excludes unnecessary features like financial management to keep the system efficient.

2.7 Summary

The existing work reviewed current poultry management systems, highlighting their disadvantages, such as limited analytics, lack of tailored features for broiler farming, and insufficient real-time data tracking. These shortcomings hinder effective decision-making and operational efficiency at Unigold. To bridge these gaps, the proposed PoultryPlus system offers a tailored, user-friendly, and efficient platform for mortality tracking, feed and vaccine management, and proactive decision-making tools.

PoultryPlus addresses issues like implementation complexity, data migration challenges, and resistance to change by ensuring ease of use and comprehensive training. It also enhances data accuracy and security, which are often lacking in existing systems. The next steps will involve finalizing system requirements with Unigold, developing the system using React, Django, and SQL, and conducting user testing to gather feedback. This will be followed by the deployment of the system, accompanied by necessary training to optimize farm operations and enhance decision-making.

Chapter 3 : REQUIREMENT ANALYSIS AND DESIGN

3.1 Introduction

To meet the unique requirements of Unigold Sdn Bhd, this chapter provides a thorough study of the requirements and design specifications for the proposed PoultryPlus system. As covered in Chapter 2, we have evaluated current poultry management methods and determined their key features, benefits, and limitations. This information guides the creation of a customized solution that improves operational efficiency while satisfying the demands of farm managers and employees.

3.2 Methodology

This project adopts an Agile SDLC methodology to ensure flexibility, iterative progress, and regular feedback. Agile is particularly suitable for projects with evolving user requirements and the need for quick feedback (Schwaber & Beedle, 2002).

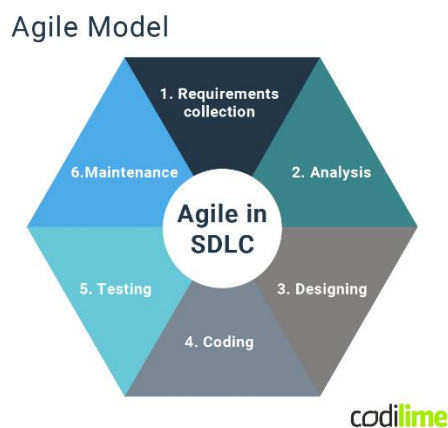


Figure 3.1 Overview of Agile Methodology by Sajna (2022)

The methodology is summarized in the following phases:

3.2.1 Requirement Collection and Analysis

Interviews and Surveys

Conduct interviews with stakeholders, including the farm manager, Mr. William and 6 farm workers from 3 different farm (Farm C, Farm E, Farm G), to identify challenges in current data management practices.

Sample Interview Questions:

1. What challenges do you face with the current data management system?
2. How do you currently track mortality rates and feed usage?
3. What key performance indicators (KPIs) do you consider most important for your operations?
4. Are there any specific features you wish you had in the current system?
5. How do you handle vaccination schedules and record-keeping?
6. What improvements would you like to see in reporting capabilities?

The full set of interview questions can be found in Appendix B.

Identifying Key Metrics

To effectively identify the key performance indicators (KPIs) relevant to Unigold Sdn Bhd's operations, the following approach will be implemented:

1. **Stakeholder Workshops:** Hold brief discussions with farm managers and staff to identify key metrics they currently track and what they find important.
2. **Data Review:** Examine existing records to see which metrics have been used and their significance.
3. **Benchmarking:** Research standard KPIs in the poultry industry, such as:
 - **Feed Conversion Ratio (FCR):**

$$FCR = \frac{\text{Total Feed Intake}}{\text{Total Weight Gain}}$$

- **Mortality Rate:**

$$\text{Mortality Rate} = \left(\frac{\text{No. of Deaths}}{\text{Total Number of Chicks}} \right) \times 100$$

- **Vaccination Schedule:** Keep a simple list of vaccination dates and dosages for tracking.
4. **Feedback Loop:** Discuss the calculated KPIs with stakeholders to ensure they align with operational needs.

Mapping Features

Define essential features like data entry forms, KPI tracking, secure data storage, and reporting capabilities.

3.2.2 System Design

- **Frontend Design:** Create a user-friendly interface using React to ensure ease of use for farm workers.
- **Backend Design:** Develop the backend using Django, implementing a RESTful API for real-time data processing and retrieval.
- **Database Design:** Use SQL to design the database schema for storing operational data such as mortality rates and feed usage.

3.2.3 System Development

- **Frontend Development:** Build a responsive interface with features for data input, KPI visualization, and report generation.
- **Backend Development:** Implement backend logic using Django for secure data handling.

- **Integration:** Integrate the frontend and backend through a RESTful API for seamless communication.

3.2.4 Testing

- **Unit Testing:** Test individual components to ensure functionality.
- **Integration Testing:** Verify that the entire system works together correctly, ensuring seamless communication between the frontend and backend.
- **User Acceptance Testing (UAT):**
 - **Target Respondents:** Involve farm workers and farm managers in the testing process to gather diverse feedback from end-users who will interact with the system daily.
 - **Focus Areas:**
 1. **User Friendliness:** Assess the ease of use and intuitive navigation of the interface.
 2. **Functionality:** Ensure that all features perform as intended and meet the specified requirements.
 3. **Performance:** Evaluate the system's speed and responsiveness under typical usage conditions.
 4. **Reliability:** Test the system's stability and error handling to confirm it can operate without failures during real-world usage.

3.3 Requirement Collection and Analysis

3.3.1 Identify Users and Characteristics

In this phase, the key users of the PoultryPlus system and their characteristics are identified to ensure the solution effectively meets their needs. The primary users include:

Farm Manager

- Role: Oversees the overall operations of the farm.
- Authorization: Has full access to all system features, including reports on productivity and KPIs.
- Responsibilities: Monitors farm performance makes strategic decisions, and reviews operational data to ensure efficient management.

Farm Worker

- Role: Responsible for daily operational tasks, including feeding, monitoring livestock, and recording data.
- Authorization: Has limited access to data entry functionalities to input relevant information regarding livestock and farm activities.
- Responsibilities: Enter and update data related to daily operations, monitor animal health, and report any issues to the farm owner or manager.

To understand the existing data management practices, current method used by farm workers to record essential operational data is analysed. Below is an image of the form they used:

M/S		No. 316120		日期 Date 11/1/25	
NO	數量 QUANTITY	貨品 PARTICULARS	單價 RATE	總額 AMOUNT	
1	50	E5 53 52	55		
2	50	52 53	56		
3	50	52 52	57		
4	51	52 52	54		
5	50	53 56 x 10	55		
6	52	53 560	54		
7	52	52	55		
8	52	52	56		
9	53	53	56		
10	52	52	55		
11	53	53	20 x 10		
12	53	52	200		
				統計 TOTAL RM	
收貨人 COLLECTOR			經手人 ISSUED BY		

以上貨物完整無誤。
The goods are complete and correct.

Figure 3.4 Weight Record of Catch Chicken

The current workflow for poultry management at Unigold Sdn Bhd involves a manual, paper-based system where farm workers are responsible for daily operational tasks such as feeding, monitoring livestock, and recording essential data. Upon the arrival of chicks, workers document the date and the total number of chickens. Each day, they conduct mortality counts twice—once in the morning at 11:30 AM and again in the evening at 5:30 PM—recording the number of dead chickens for each shed. Additionally, farm workers feed the chickens twice daily, noting the amount of feed used in "guni" and calculating the total for the day. When the chickens reach the appropriate weight for culling, workers catch and cull them, recording the weight of the culled chickens and the total number caught each day.

For inventory management, farm workers verbally inform the Farm Manager when stocks of feed or vaccines are running low, often leading to last-minute orders. This reactive approach can result in delays and potential shortages, emphasizing the need for a more

proactive inventory management system. Once the shed is emptied, these records are submitted to the Farm Manager, who manually reviews and calculates KPIs such as FCR and mortality rates from the recorded data. This manual process is time-intensive and prone to errors, highlighting the need for a more efficient and streamlined data management solution, such as the proposed PoultryPlus system.

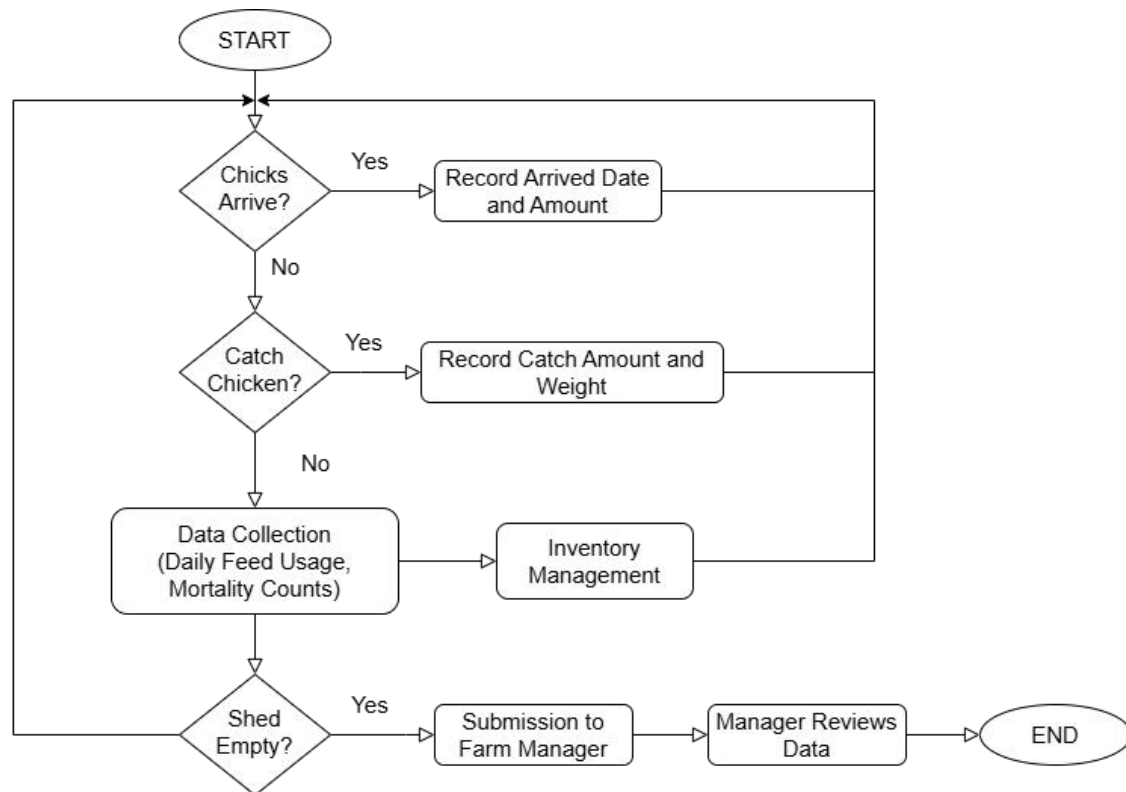


Figure 3.5 Current Workflow for Unigold

3.3.2. Functional Requirements

1. **User Authentication:** The system must allow users to log in using their passport number and a unique password.
2. **Farm and Shed Selection:** Users must be able to select the specific farm (C, E, G) and shed (e.g., E1, E2) they are working in before entering data.
3. **Data Entry:**

Farm workers must be able to record:

- a. Chick arrivals and their average weight.
 - b. Daily feed usage and stock levels.
 - c. Mortality rates for both morning and afternoon.
 - d. Vaccine administration and schedule adherence.
 - e. Weekly chicken weight measurements.
4. **Automated Report Generation:** The system must automatically generate reports that compare calculated KPIs and FCR against standard benchmarks for review by farm managers.
5. **Inventory Tracking:** The system must enable farm managers to monitor real-time levels of feed and vaccines, highlighting low stock alerts for proactive restocking.
6. **Simplified Metrics for Workers:** Provide farm workers with limited access to real-time performance metrics, such as daily mortality rates or feed efficiency, to assist in their tasks.

3.3.3. Non-Functional Requirements

Non-functional requirements specify how the PoultryPlus system should operate and describe the quality attributes it must possess. Unlike functional requirements, which detail specific behaviors or tasks the system performs (like logging in or recording data), non-functional requirements focus on the overall performance, security, reliability, usability, scalability, and maintainability of the system. These attributes are crucial for ensuring that PoultryPlus meets user expectations and functions effectively in a real-world poultry management setting.

3.3.3.1 Performance Requirements

- **Response Time:** The system should complete user actions, such as logging in and recording chick arrivals, within 2 seconds under normal load conditions to ensure efficiency in daily operations.

- **Concurrency:** PoultryPlus must support at least 50 simultaneous users, allowing multiple farm workers and managers to operate without slowdowns, which is vital during peak operational hours.
- **Data Processing:** Reports on KPIs and FCR should be generated within 5 seconds after a request, enabling managers to make timely decisions based on accurate data.

3.3.3.2 Security Requirements

Password Management

Passwords are essential for ensuring that only authorized users can access the PoultryPlus system.

- 1 . The PoultryPlus system shall allow users to manually set their password, which must be a combination of letters, numbers, and special characters with a minimum length of 12 characters.
- 2 . The system shall authenticate the valid Login ID with its associated password to grant access.
- 3 . The system shall display an alert box requesting users to re-login if the entered Login ID or password is invalid, with a link to the password recovery process for assistance.

3.3.3.3 Reliability Requirement

1. The PoultryPlus system shall be capable of avoiding failures due to software defects through rigorous testing and validation.
2. The PoultryPlus system shall operate smoothly without experiencing system hangs, crashes, or unexpected downtime.
3. The PoultryPlus system shall be designed to recover data and restore functionality within a specified time frame in the event of a system failure.

4. The PoultryPlus system shall maintain acceptable response times, ensuring that user requests are processed efficiently, as detailed in Section 3.3.3.1.

3.3.3.4 Usability Requirement

The PoultryPlus system shall feature an intuitive user interface that enhances user experience and facilitates task completion. The usability requirements include:

1. The PoultryPlus system shall use consistent and readable fonts throughout the application to enhance legibility.
2. The PoultryPlus system shall employ standardized icons and graphics to improve user navigation and recognition.
3. The PoultryPlus system shall include clearly distinguishable clickable buttons, separate from labels, to facilitate user interactions.
4. The PoultryPlus system shall maintain consistent layout structures across all screens to provide a cohesive user experience.

3.3.4. Software Requirement

The software requirements focus on the operating system, database server, development language, and development tools necessary for the efficient operation of the PoultryPlus system. The minimum software requirements are outlined below:

Table 3 Specification of PoultryPlus Software Requirements

System/Software	Description
Operating System	Windows 10 and Windows 11

Database Server	SQL
Development Language	Python
Development Tools	Django, Angular, Visual Studio Code, Jupyter Notebook, Git, Docker

3.3.5. Hardware Requirement

The hardware requirements focus on the processor, processing speed, computer memory, and storage. Users must have a complete computer setup with peripherals such as a keyboard and mouse. The minimum hardware requirements for the PoultryPlus system are categorized and specified below:

Table 4 Specification of PoultryPlus Hardware Requirements

Hardware	Description
Processor	Intel® Core i5 or i7
Processing Speed	1.8 GHz
Computer Memory	8 GB RAM or higher
Storage	HDD 500 GB and above

3.4 System Design

In this section, the UML diagrams, including the use case diagram, activity diagram, and class diagram, are utilized to illustrate the design and functionality of the PoultryPlus System. These diagrams help visualize the interactions between users and the system, the flow of activities, and the structural design of the system to ensure efficient poultry management.

3.4.1. Use Case Diagram

3.4.1.1 Overview of PoultryPlus System Use Case Diagram

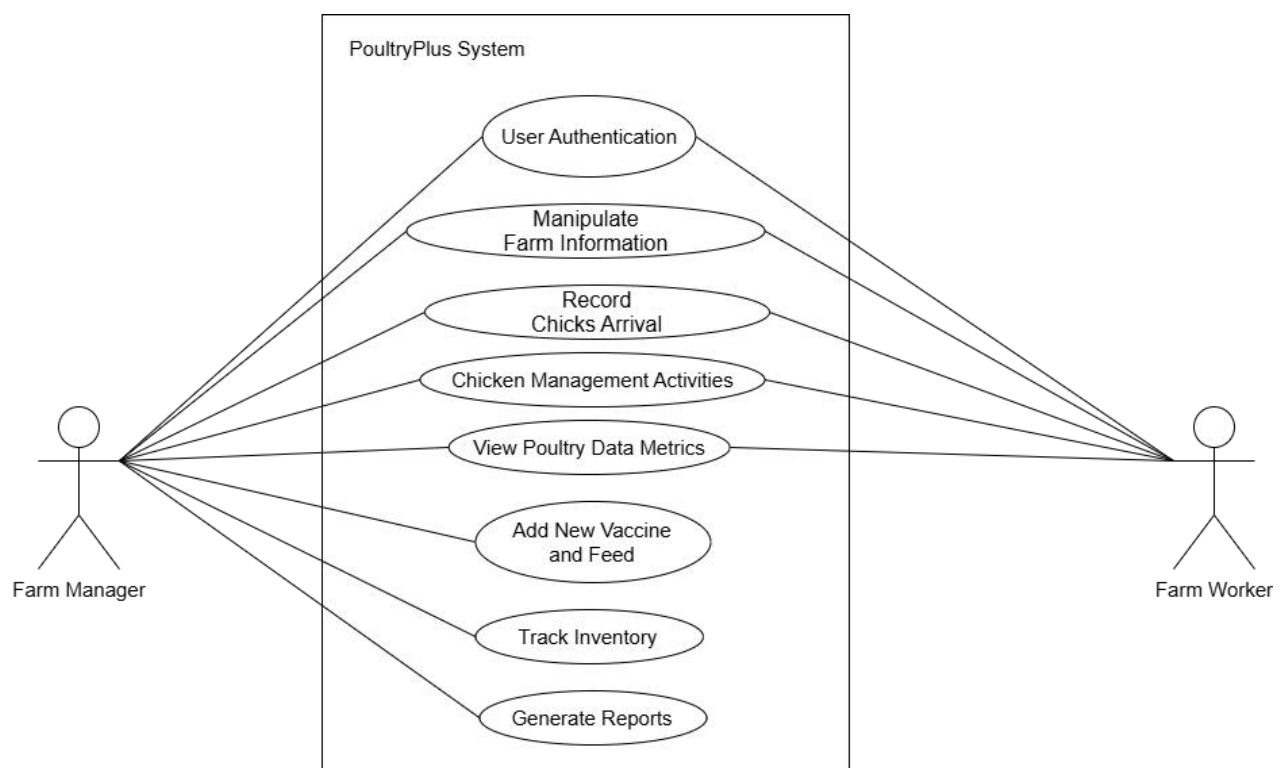


Figure 3.6 Overview of PoultryPlus System with Use Case Scenario

Figure 3.1 shows the Overview of the PoultryPlus System Use Case Diagram, illustrating the interactions between various actors, including Farm Managers and Farm Workers, and the system's functionalities. This diagram highlights key processes such as user authentication, farm and shed management, vaccine and feed tracking, chick arrival recording, and performance metrics visualization. By providing a clear representation of these use cases, the

diagram serves as a valuable tool for understanding the system's operational structure and the roles of different users in managing poultry operations effectively.

3.4.1.2 Use Case Scenario 1: User Authentication System

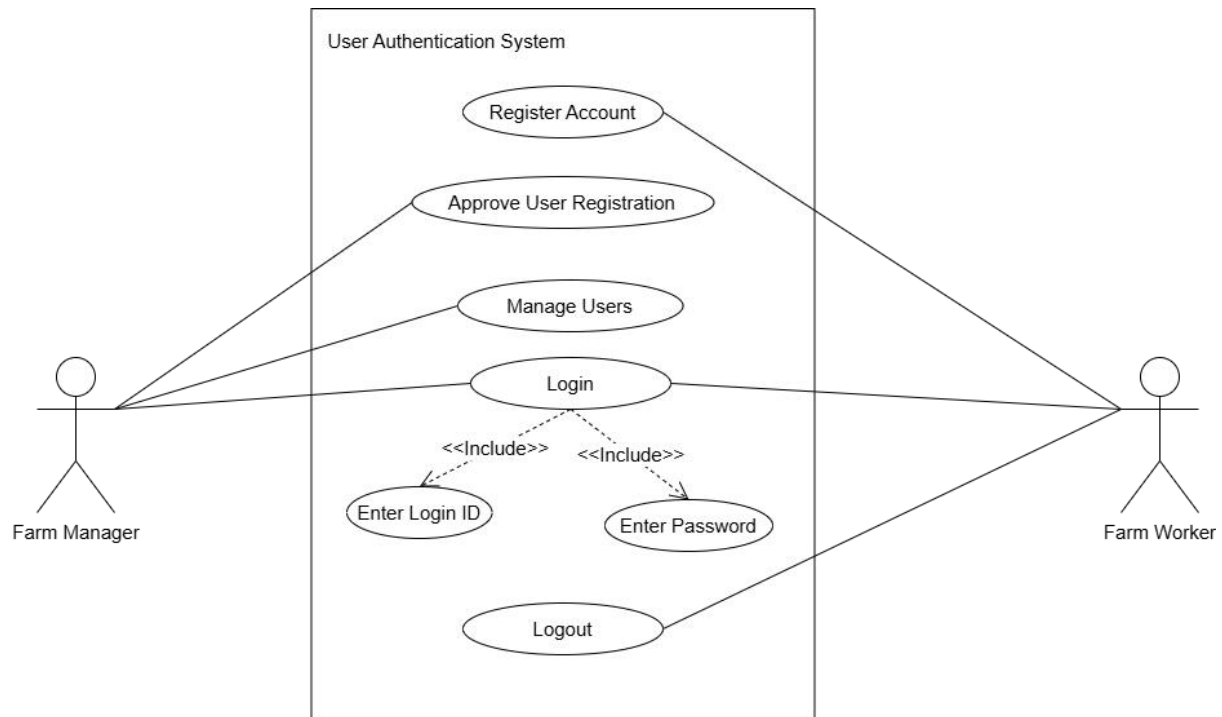


Figure 3.7 Use Case Scenario 1: User Authentication

Use Case ID:	UC1		
Use Case Name:	User Authentication		
Created By:	Tan Zi Siah	Date Last Updated:	30/12/2024
Actor:	1. Farm Manager 2. Farm Worker		
Description:	Farm worker can register for an account in the system. After registration, a manager must approve the user's account before they can log in. If a farm worker attempts to log in before approval, the system will indicate that their account is pending approval. Once approved, farm worker can log in and are directed to their respective main pages. Access levels vary based on user roles.		
Preconditions:	1. Farm worker must provide necessary information to register. 2. Farm worker must have a valid Login ID, password and also been approve by the manager. 3. Managers must log in to review and approve registrations.		
Post-conditions:	1. Users are successfully logged into the system. 2. Users access their respective dashboards.		

Normal Course of Events:	1. Farm Worker clicks “Register” button. 2. Farm Worker fills in the registration form. 3. Farm Worker submits the form. 4. System sends registration details to the Manager for approval. 5. Farm Worker enters Login ID and password on the login page. 6. Farm Worker clicks “Login” button. 7. System verifies credentials and displays the Farm Worker’s main page (if approved).
Alternative Courses:	AC. 1. Manager Approval: 1. Manager reviews pending registrations. 2. Manager approves or rejects the registration. AC. 2. View Profile Information: 1. Manager click “View Profile Information” button. 2. System displays the worker’s profile information AC. 3. Update Profile Information: 1. Manager click “Update” button. 2. System allows the Manager to edit worker profile fields. 3. Manager submits the updated information. 4. System saves the changes and confirms the update. AC. 4. Delete User: 1. Manager clicks “Delete User” button. 2. System prompts for confirmation to delete the user account. 3. Manager confirms deletion. 4. System removes the user account and notifies the Manager of the successful deletion.
Exceptions:	EX. 1. Invalid Login ID Alert prompts Farm Worker to enter a valid Login ID. EX. 2. Invalid Password Alert prompts Farm Worker to enter a valid password. EX. 3. Account Pending Approval Alert notifies Farm Worker that their account is pending manager approval. EX. 4. Registration Error Alert prompts Farm Worker if registration details are invalid or if the Login ID already exists.
Business Rule:	1. Each Farm Worker must have a unique Login ID. 2. Each Farm Worker must have a password. 3. Farm Workers may share passwords, but Login IDs must remain unique. 4. All registration details must be validated before account creation.

	5. Managers must approve registrations before Farm Workers can log in.
--	--

3.4.1.3 Use Case Scenario 2: Manipulate Farm Information

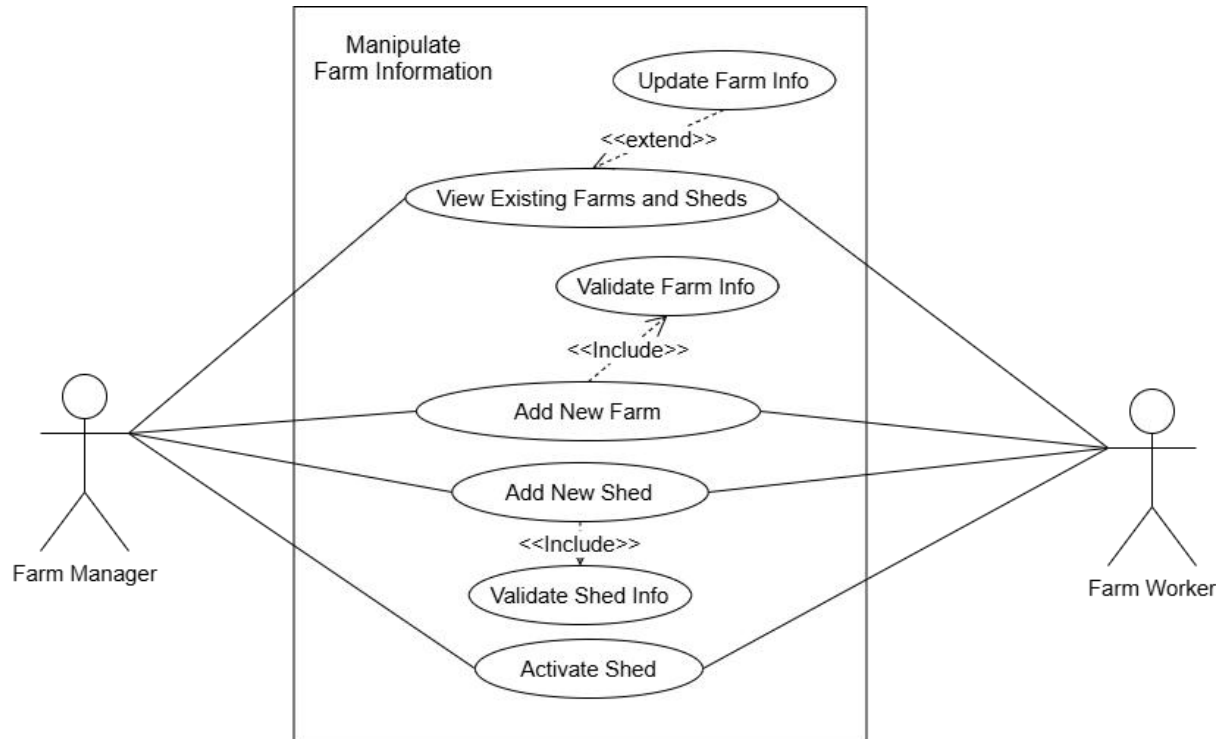


Figure 3.8 Use Case Scenario 2: Manipulate Farm Information

Use Case ID:	UC2		
Use Case Name:	Manipulate Farm Information		
Created By:	Tan Zi Siah	Date Last Updated:	30/12/2024
Actor:	Farm Worker, Farm Manager		
Description:	Users can add a new farm or a new shed for a particular farm. Users can also activate a shed when chicks arrive.		
Preconditions:	User must be logged in.		
Post-conditions:	1. New farm or shed is added successfully. 2. Shed is activated for use when chicks arrive.		
Normal Course of Events:	Adding a New Farm: 1. User selects “Add New Farm” option. 2. User enters the following information: Farm Name, Number of Sheds, Date of Establishment, Location 3. User submits the information. 4. System confirms the new farm has been added. Adding a New Shed: 1. User selects an existing farm from the list.		

	2. User clicks “Add New Shed” option. 3. User enters the following information: Date, Maximum Number of Chickens 4. User submits the information. 5. System confirms the new shed has been added to the selected farm. Activating a Shed: 1. User chooses the farm and the specific shed from the list. 2. User selects “Activate Shed” option. 3. User selects who will oversee the shed. 4. User submits the activation request. 5. System activates the shed and confirms the action.
Alternative Courses:	AC. 1. View Existing Farms and Sheds: 1. User selects “View” option. 2. User can select a farm to view details or update information. AC. 2. Edit Existing Shed Information: 1. User selects an existing shed. 2. User clicks “Edit Shed” button. 3. User makes the necessary updates. 4. User submits the updated information. 5. System confirms the updates. AC. 3. View History of each shed: 1. User selects an existing shed. 2. User clicks “History” button.
Exceptions:	EX. 1 Invalid Farm Information. - Alert prompts the user to enter a valid farm detail. EX. 2 Invalid Shed Information. - Alert prompts the user to enter valid shed details. EX. 3 Shed Already Active. - Alert notifies the user that the shed cannot be activated as it is already in use.
Business Rule:	1. Each farm must have a unique name. 2. Each shed must have a maximum capacity defined. 3. Sheds can only be activated when they have been used (e.g., feed stocked or on the first day of chickens)

3.4.1.4 Use Case Scenario 3: Add New Vaccine and Feed

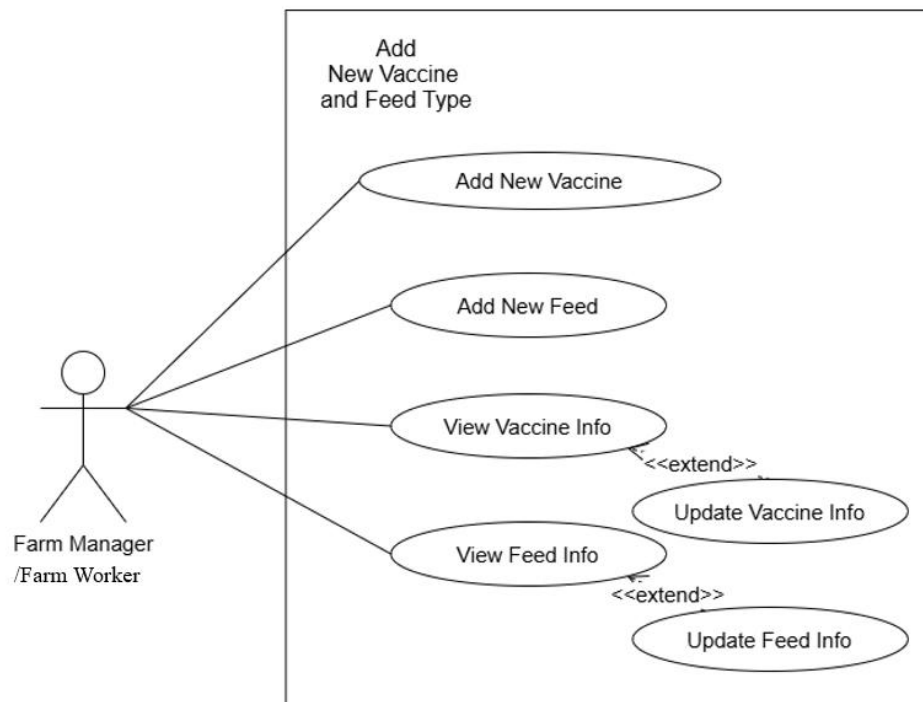


Figure 3.9 Use Case Scenario 3: Add New Vaccine and Feed

Use Case ID:	UC3		
Use Case Name:	Add New Vaccine and Feed Type		
Created By:	Tan Zi Siah	Date Last Updated:	30/12/2024
Actor:	Farm Manager and Farm Worker		
Description:	Allows the users to add new vaccines and feeds to the system.		
Preconditions:	User must be logged in.		
Post-conditions:	New vaccine and feed data are added to the database.		
Normal Course of Events:	Adding a New Vaccine: 1. User selects "Add New Vaccine" option. 2. System prompts for vaccine details (VaccineID, VaccineType, Volume). 3. User enters vaccine information. Adding a New Feed: 1. Manager selects "Add New Feed" option. 2. System prompts for feed details (FeedID, FeedType, Volume Per Package). 3. User enters feed information.		
Alternative Courses:	AC. 1 View Exist Vaccine Info 1. User selects the "View Existing Vaccine" option. 2. System displays a list of all vaccines currently in the system.		

	3. User can select a vaccine from the list to view detailed information. AC. 2 View Exist Feed Info 1. User selects the “View Existing Feed” option. 2. System displays a list of all feed types currently in the system. 3. User can select a feed type from the list to view detailed information. AC. 3 Update Exist Vaccine Info 1. User selects the “Update Vaccine” option. 2. System displays the details of the selected vaccine. 3. User updates the required fields (e.g., volume or vaccine type). 4. User submits the updated information. 5. System confirms the update and reflects the changes in the database. AC. 4 Update Exist Vaccine Info 1. User selects the “Update Feed” option. 2. System displays the details of the selected feed. 3. User updates the required fields (e.g., volume per package or feed type). 4. User submits the updated information. 5. System confirms the update and reflects the changes in the database.
Exceptions:	EX. 1 Invalid Farm Information. - Alert prompts the user to enter a valid feed or vaccine details.
Business Rule:	1. Each vaccine and feed must have a unique identifier (VaccineID and FeedID). 2. Quantity must be a positive integer. 3. VaccineType and FeedType must adhere to predefined categories approved by the farm management.

3.4.1.5 Use Case Scenario 4: Record Chicks Arrival

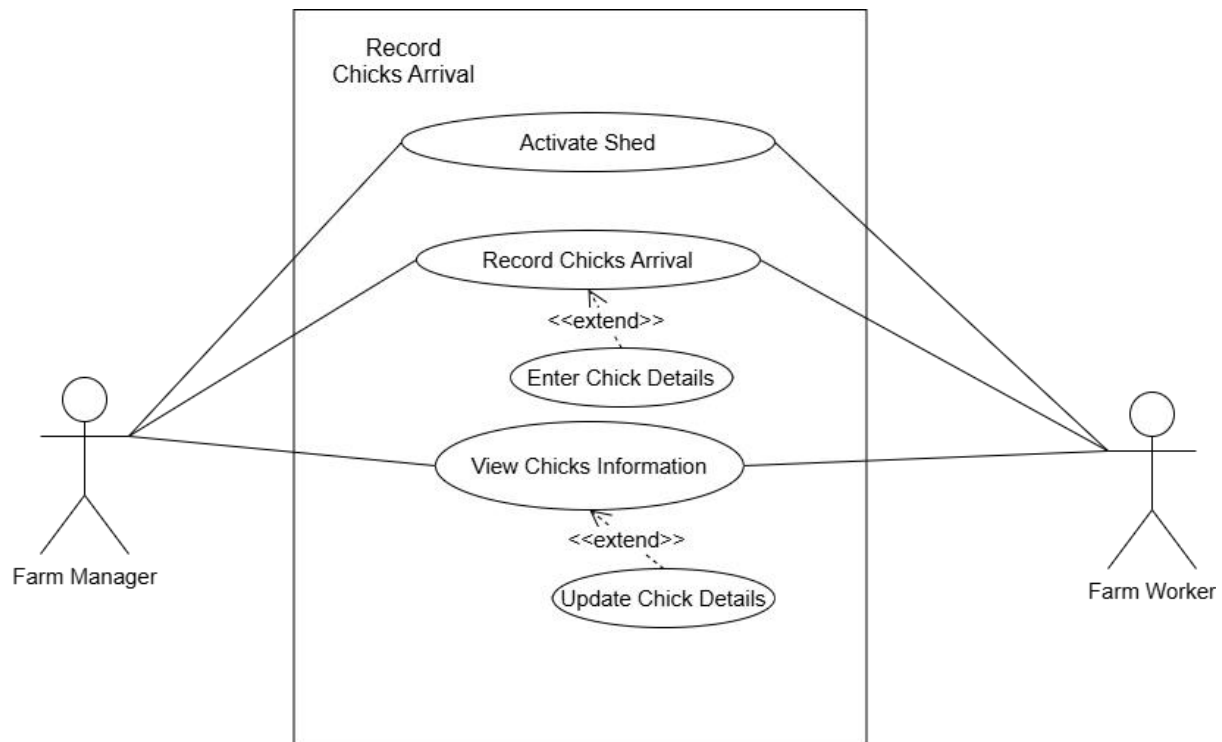


Figure 3.10 Use Case Scenario 4: Record Chicks Arrival

Use Case ID:	UC4		
Use Case Name:	Record Chicks Arrival		
Created By:	Tan Zi Siah	Date Last Updated:	30/12/2024
Actor:	Farm Worker, Farm Manager		
Description:	Users can record the arrival of chicks after activating the specific shed. They enter details regarding the chicks, including type, quantity, mortality during transport, vaccination status, and initial weight.		
Preconditions:	1. User must be logged in. 2. The specific shed must be activated before recording can begin.		
Post-conditions:	1. The record of chicks' arrival is saved in the system. 2. The activated shed's cycle count is updated with the new chicks' information.		
Normal Course of Events:	1. Activate Shed: - User navigates to the "Activated Shed" page. - User selects the specific shed. 2. Start Recording Arrival:		

	• User selects "Record Chicks Arrival" option. • System displays the activated shed's details, including the current cycle count and date. 3. Enter Chick Details: • User selects the type of chick from a dropdown list. • User enters the total number of chicks arriving. • User inputs the number of chicks that died during transport. • User indicates if the chicks have received the IBND vaccine: • If yes, the system records this status. • If no, user specifies how many doses will be given manually. 4. Record Initial Weight: • User weighs 20 chicks and records the weight. • User repeats the weighing process two more times (total of three weighing). • System calculates and displays the average weight. 5. Submit Record: • User submits the chick arrival record. • System confirms the record is saved successfully.
Alternative Courses:	AC 1: View Chicks Information: • User can choose to view the chicks' information of the specific shed. • System displays the chicks record, including their type, arrival count, mortality, vaccination status, and weights. AC 2: Update Chicks Information: • User can update the shed details or chick records as necessary. • System saves the updates and confirms the changes.
Exceptions:	EX 1: Weight Measurement Error: If the weight measurements are inconsistent, the system prompts the user to recheck the weights. EX 2: Invalid Data Input: If the user enters invalid data (e.g., negative numbers or non-numeric values where numbers are required), the system displays an error message prompting the user to enter valid data.
Business Rule:	1. A shed must be activated before recording chick arrivals. 2. The type and quantity of chicks must be specified. 3. Vaccination status must be recorded accurately. 4. Weight must be recorded as an average from three measurements. 5. The cycle count for the shed is incremented each time new chicks arrive.

3.4.1.6 Use Case Scenario 5: Chicken Management Activities

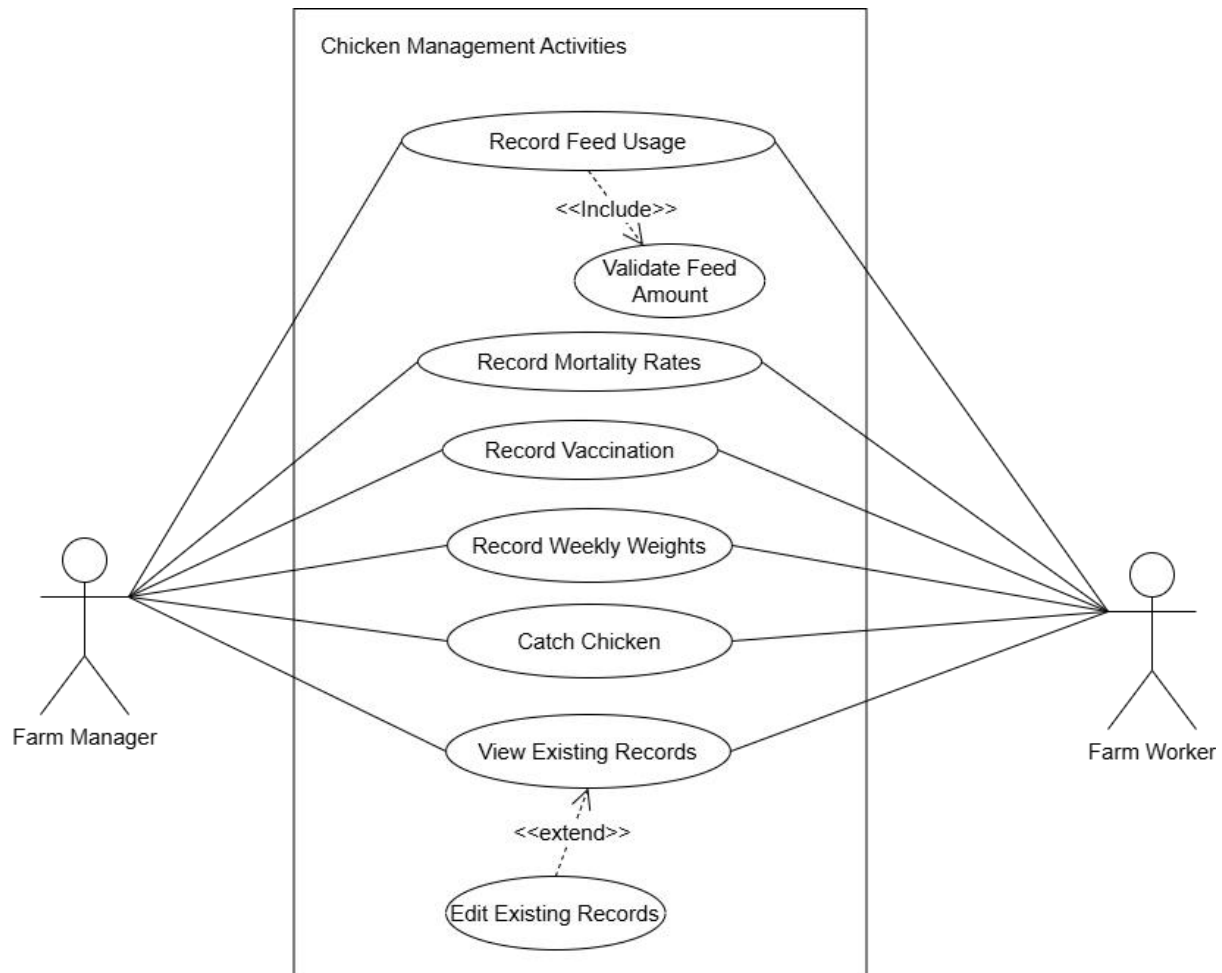


Figure 3.11 Use Case Scenario 5: Chicken Management Activities

Use Case ID:	UC5		
Use Case Name:	Chicken Management Activities		
Created By:	Tan Zi Siah	Date Last Updated:	30/12/2024
Actor:	Farm Worker, Farm Manager		
Description:	This use case allows users to record the daily feed usage, mortality rates, vaccinations, and weekly weights of the chickens. Records are maintained for both morning and afternoon sessions.		
Preconditions:	1. User must be logged in. 2. The specific shed must be activated.		
Post-conditions:	Records of feed usage, mortality, vaccinations, and weights are saved in the system.		
Normal Course of	1. User selects an activated shed.		

Events:	2. User chooses the type of record to add (Feed Arrival, Feed Usage, Mortality, Vaccination, Weight, Catch Chicken). 3. User enters relevant data. 4. User submits the record. 5. System saves the data and confirms entry.
Alternative Courses:	AC 1: View Existing Records: 1. User selects the “View Records” option. 2. System displays a list of existing records for feed usage, mortality, vaccinations, and weights for the selected shed.. AC 2: Edit Existing Records: 1. User selects a record from the list to edit. 2. User updates the necessary fields. 3. User submits the changes. 4. System confirms the updates and saves the modified record. AC 3: Inactivate the shed 1. Users can inactivate the shed once all the chicken have been catch. 2. User submits the changes. 3. System confirms the updates and saves the modified record.
Exceptions:	EX 1: Invalid Data Input: • If the user enters invalid data (e.g., negative numbers or non-numeric values), the system displays an error message prompting the user to enter valid data. EX 2: Missing Required Fields: • If any required fields are left blank, the system alerts the user to complete all mandatory fields before submission. EX 3: Weight Measurement Error: • If the weight measurements are inconsistent, the system prompts the user to recheck the weights.
Business Rule:	1. Each record type must have specific fields filled out correctly before submission. 2. Daily records must be maintained separately for morning and afternoon sessions. 3. The system should automatically timestamp entries with the date and time of submission. 4. Each chicken must have its vaccination status recorded accurately, following the vaccination schedule.

3.4.1.7 Use Case Scenario 6: View Poultry Data Metrics

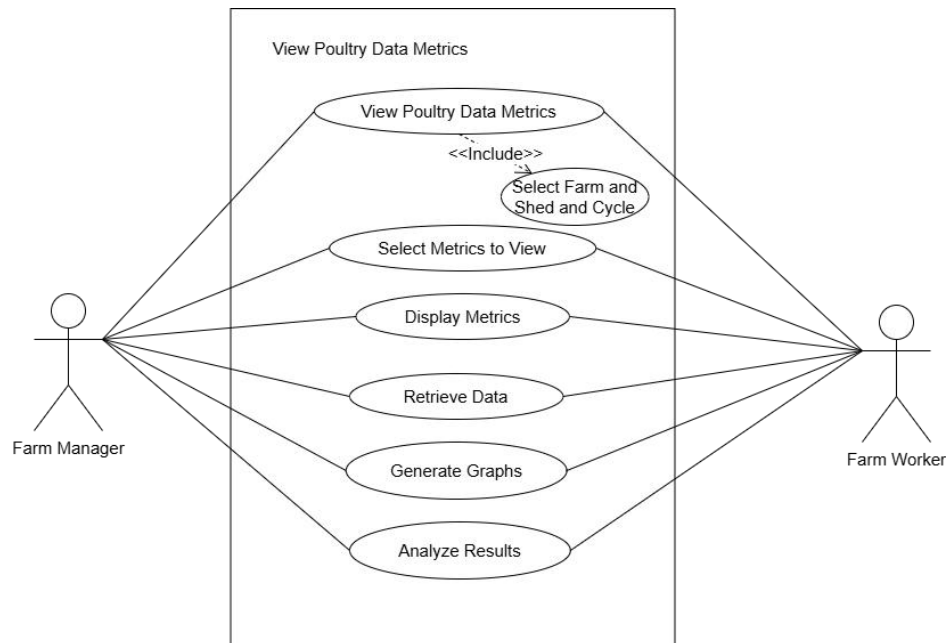


Figure 3.12 Use Case Scenario 6: View Poultry Data Metrics

Use Case ID:	UC6		
Use Case Name:	View Poultry Data Metrics		
Created By:	Tan Zi Siah	Date Last Updated:	30/12/2024
Actor:	Farm Worker, Farm Manager		
Description:	This use case allows users to view current metrics related to Feed Conversion Ratio (FCR), weight, mortality rates, and feed usage, comparing them against standard values. The results are visualized through graphs for easier analysis.		
Preconditions:	1. User must be logged in.		
Post-conditions:	1. Users can see the current metrics visually represented in graphs. 2. Users can analyse whether the metrics meet the standard values.		
Normal Course of Events:	User navigates to the Metrics Dashboard: User accesses the "Poultry Data Metrics" section from the main menu. User selects farm, shed and which cycle: User chooses from a list of farm and sheds available for viewing metrics. User selects metrics to view: User chooses the specific metrics to review (e.g., FCR,		

	weight, mortality rates, feed usage) for the selected shed. 4. System retrieves data: The system retrieves the latest data related to the selected shed from the records. 5. System displays metrics: • The system calculates the current values for FCR, weight, mortality rates, and feed usage specific to the selected shed. • The system compares these values against standard benchmarks. 6. System generates graphs: • The results are displayed in graphical form (e.g., bar graphs, line charts) for easy interpretation. • Users can quickly see which metrics meet or exceed standard values and which do not. 7. User analyzes results: • User reviews the graphs and metrics to determine the performance of the selected shed. • Users can make informed decisions based on the visualized data.
Alternative Courses:	AC 1: Filter Metrics: 1. User applies filters to view specific time periods (e.g., weekly, monthly). 2. The system updates the graphs accordingly.
Exceptions:	EX 1: Data Retrieval Error: • If there is an issue retrieving the latest data for the selected shed, an error message is displayed. EX 2: No Data Available: • If no data exists for the selected metrics in that shed, a message informs the user.
Business Rule:	1. Metrics must be updated in real-time or at regular intervals. 2. Standard values must be defined clearly within the system. 3. Graphs must be clear and easily interpretable, allowing users to make quick assessments. 4. Users must have the option to filter and download graphs for reporting purposes.

3.4.1.8 Use Case Scenario 7: Track Inventory

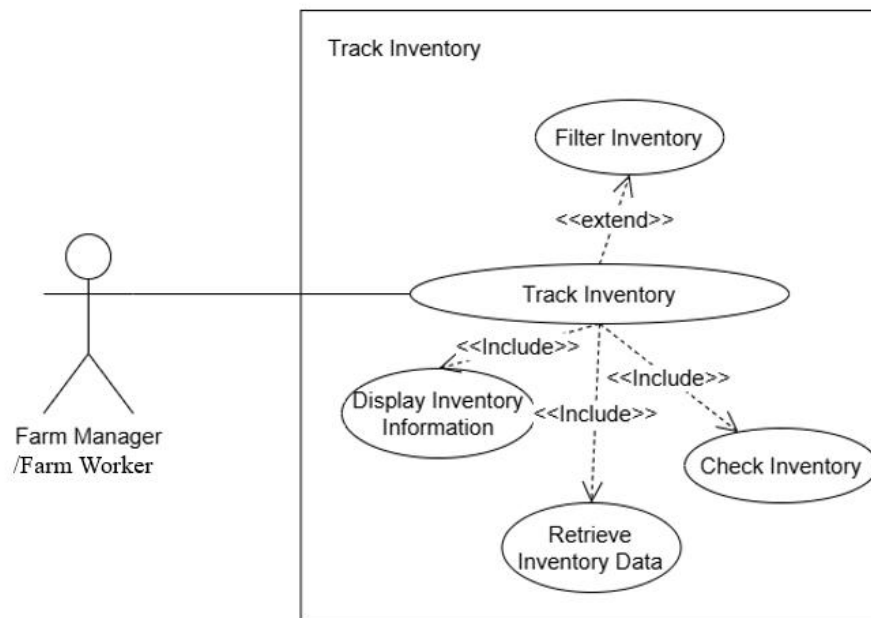


Figure 3.13 Use Case Scenario 7: Track Inventory

Use Case ID:	UC7		
Use Case Name:	Track Inventory		
Created By:	Tan Zi Siah	Date Last Updated:	30/12/2024
Actor:	Farm Manager, Farm Worker		
Description:	This use case allows user to check the inventory of vaccines used and those that are left over, enabling efficient management of poultry health resources.		
Preconditions:	User must be logged in.		
Post-conditions:	1. User can view the current inventory of vaccines. 2. User can see which vaccines have been used and how many are left.		
Normal Course of Events:	1. User navigates to the Inventory Dashboard: User accesses the "Inventory" section from the main menu. 2. User selects "Check Inventory": User clicks on the "Check Inventory" button. 3. System retrieves inventory data: The system retrieves the latest inventory data for vaccines from the database. 4. System displays inventory information: The system displays a list of vaccines, showing quantities used and remaining stock. 5. User reviews inventory: User reviews the inventory data to make decisions about		

	future vaccine orders or usage.
Alternative Courses:	AC 1: Filter Inventory: - User can apply filters to view specific types of vaccines (e.g., by category or date). - The system updates the displayed inventory accordingly.
Exceptions:	EX 1: Data Retrieval Error: - If there is an issue retrieving the inventory data, an error message is displayed. EX 2: No Inventory Data Available: - If no inventory data exists, a message informs the Farm Manager.
Business Rule:	- Inventory data must be updated in real-time or at regular intervals. - Vaccines must be clearly categorized and labeled within the system. - The system must provide user-friendly displays of inventory levels.

3.4.1.9 Use Case Scenario 8: Generate Report

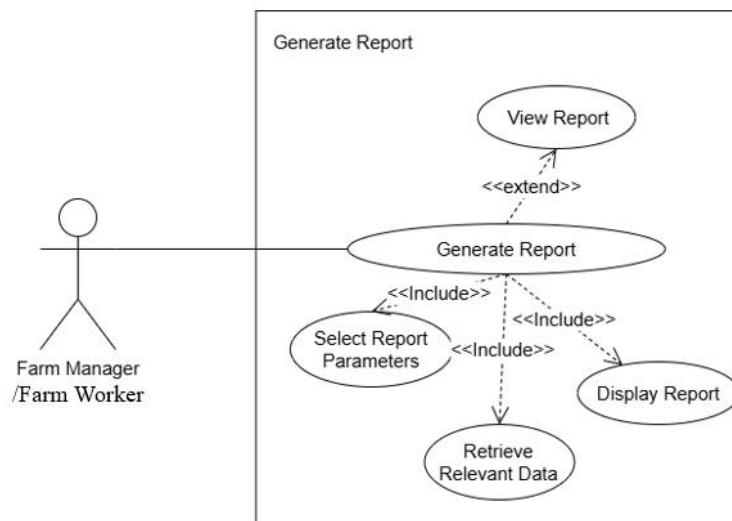


Figure 3.14 Use Case Scenario 8: Generate Report

Use Case ID:	UC8		
Use Case Name:	Generate Report		
Created By:	Tan Zi Siah	Date Last Updated:	30/12/2024
Actor:	Farm Manager, Farm Worker		
Description:	This use case allows user to generate comprehensive reports based on various metrics, including inventory data, poultry performance, and vaccination records. This functionality helps the manager assess farm operations and make informed decisions.		
Preconditions:	User must be logged in.		
Post-conditions:	User can view the generated report.		

Normal Course of Events:	1. User navigates to the Reports Dashboard: User accesses the "Reports" section from the main menu. 2. User selects report parameters: User selects parameters for the report (e.g., date range, type of data). 3. System retrieves relevant data: The system retrieves the data based on the selected parameters. 4. System generates the report: The system compiles the data into a structured report format (e.g., PDF, CSV). 5. System displays the report: The generated report is displayed on the screen for review.
Alternative Courses:	-
Exceptions:	EX 1: Data Retrieval Error: • If there is an issue retrieving the relevant data, an error message is displayed. EX 2: No Data Available: • If no data exists for the selected parameters, a message informs the Farm Manager.
Business Rule:	1. Reports must be generated in real-time or at defined intervals. 2. The system must allow customizable parameters for report generation. 3. Reports must be generated in user-friendly formats.

3.4.2. Activity Diagram

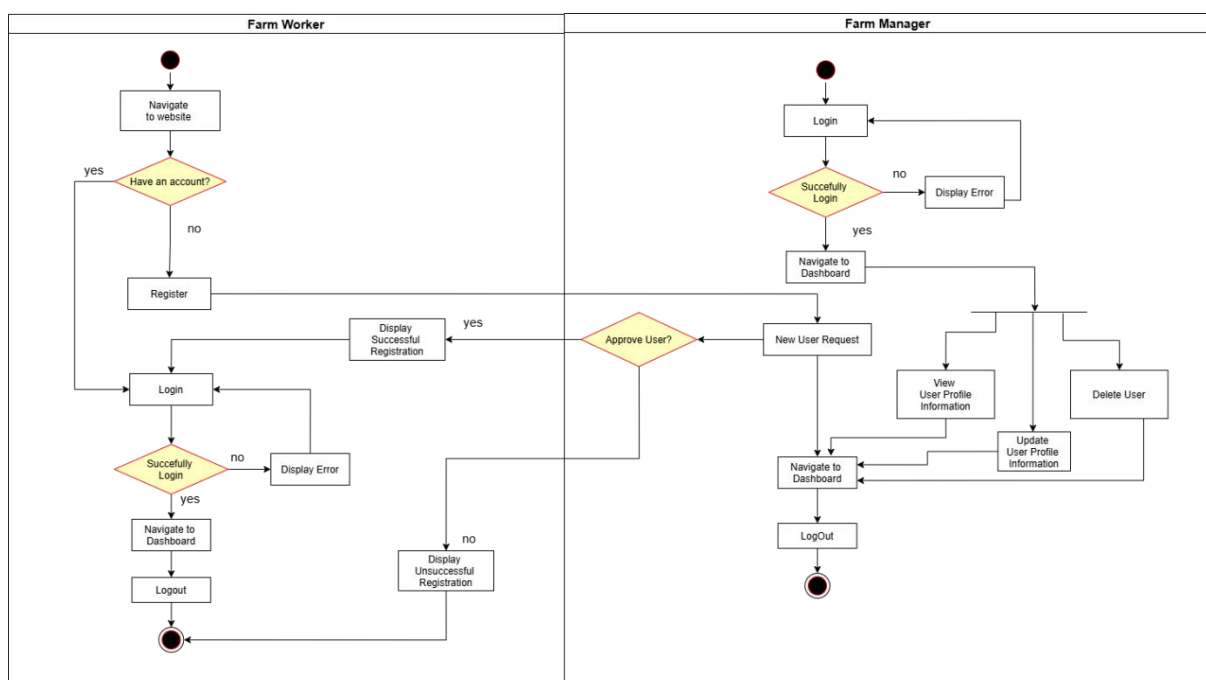


Figure 3.15 Activity Diagram for Use Case 1: User Authentication

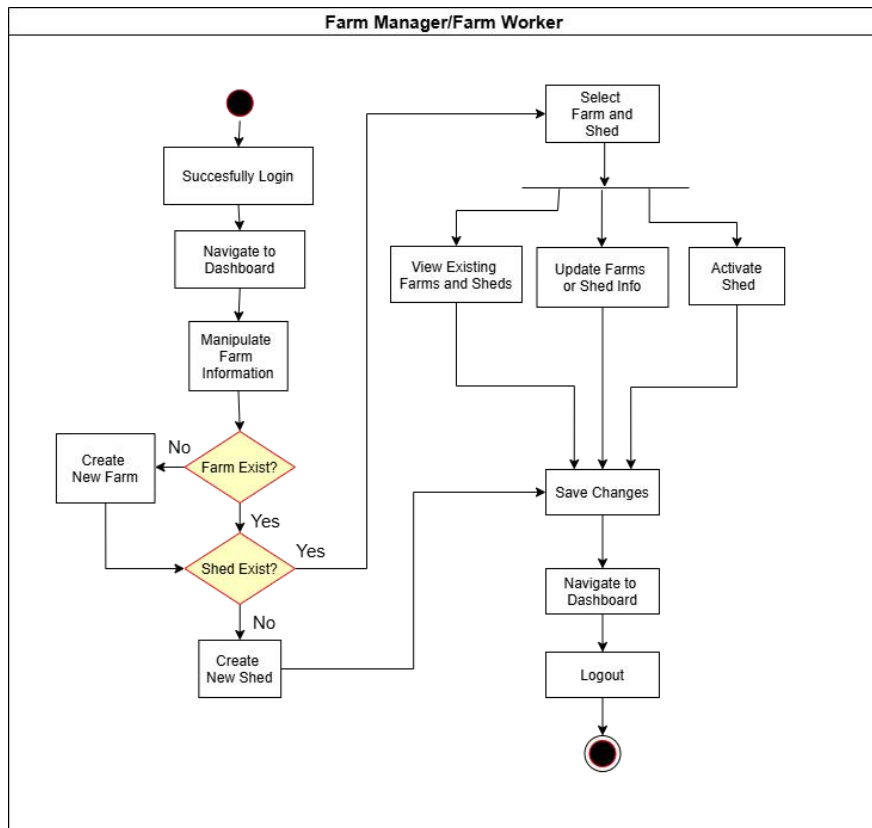


Figure 3.16 Activity Diagram for Use Case 2: Manipulate Farm Information

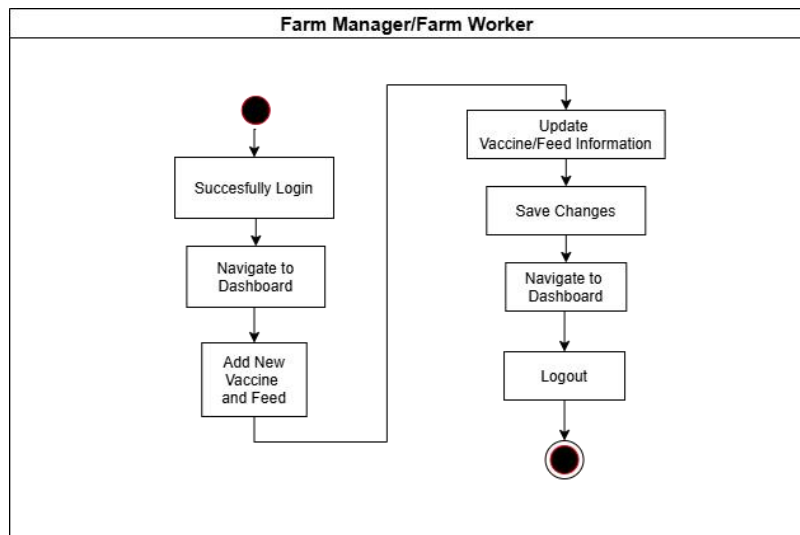


Figure 3.17 Activity Diagram for Use Case 3: Add New Vaccine and Feed

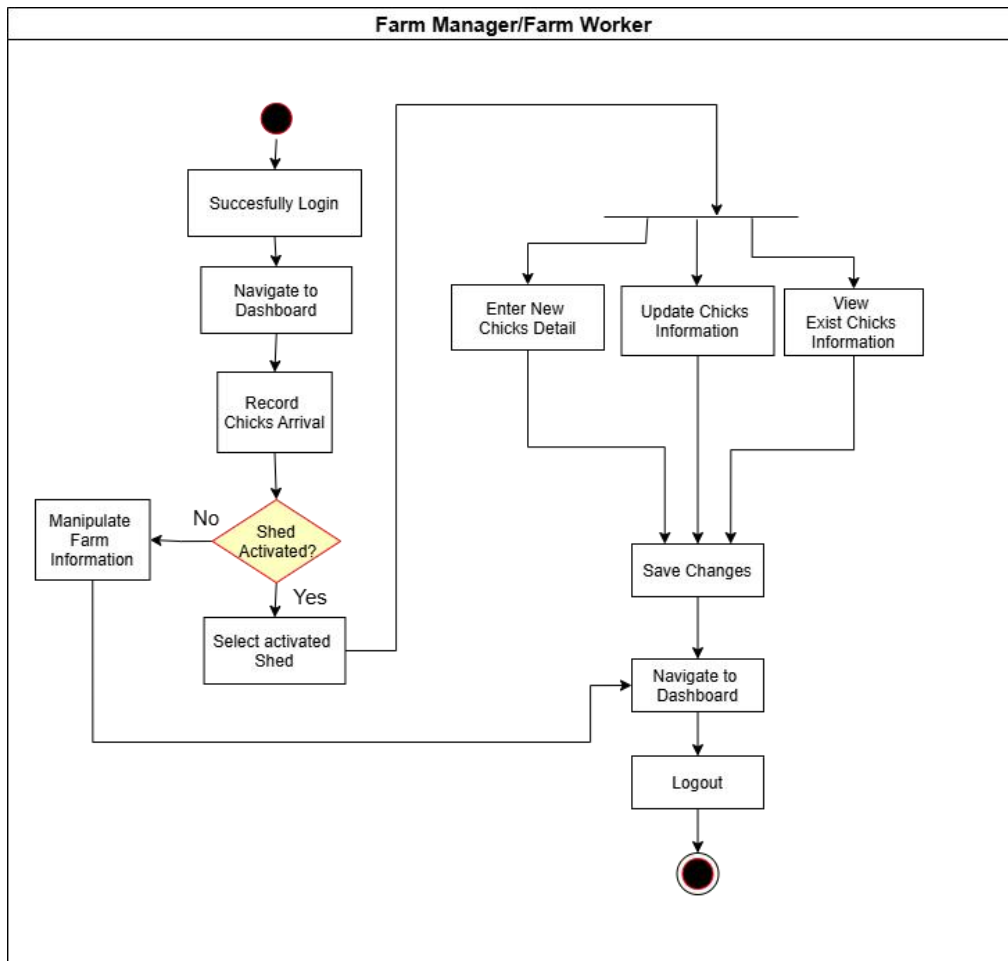


Figure 3.18 Activity Diagram for Use Case 4: Record Chick Arrival

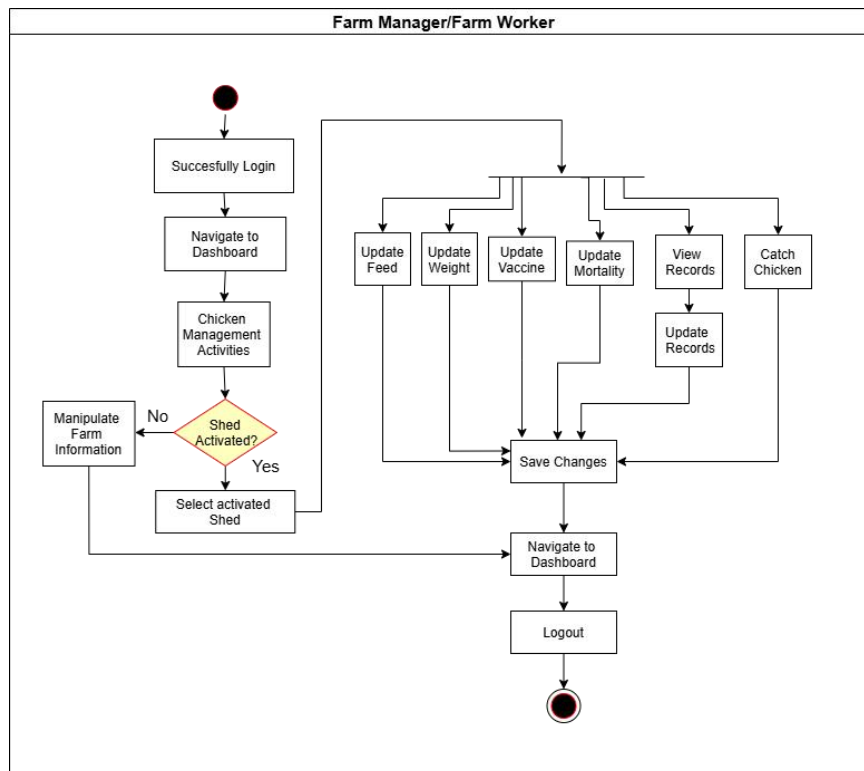


Figure 3.19 Activity Diagram for Use Case 5: Chicken Management Activities

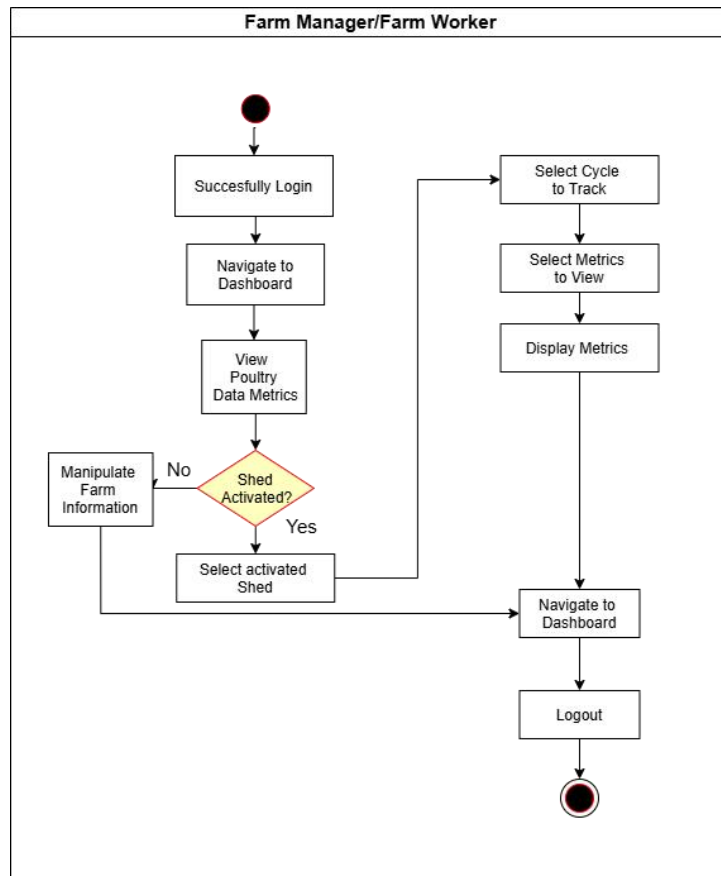


Figure 3.20 Activity Diagram for Use Case 6: View Poultry Data Metrics

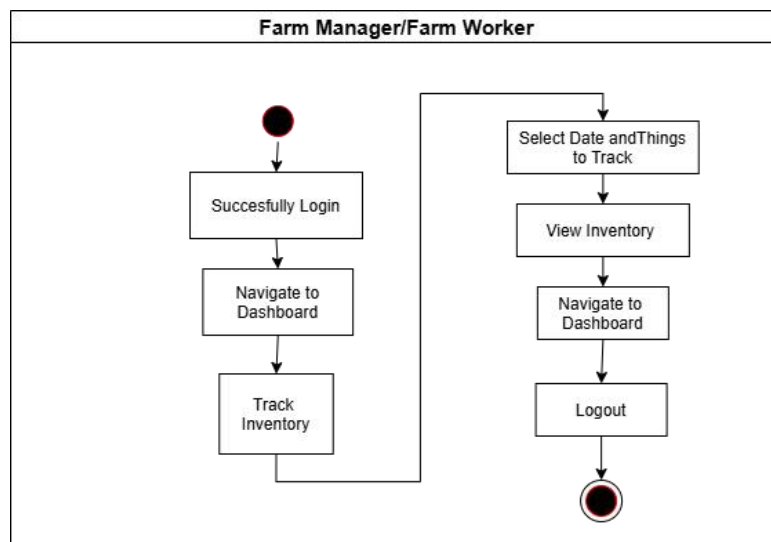


Figure 3.21 Activity Diagram for Use Case 7: Track Inventory

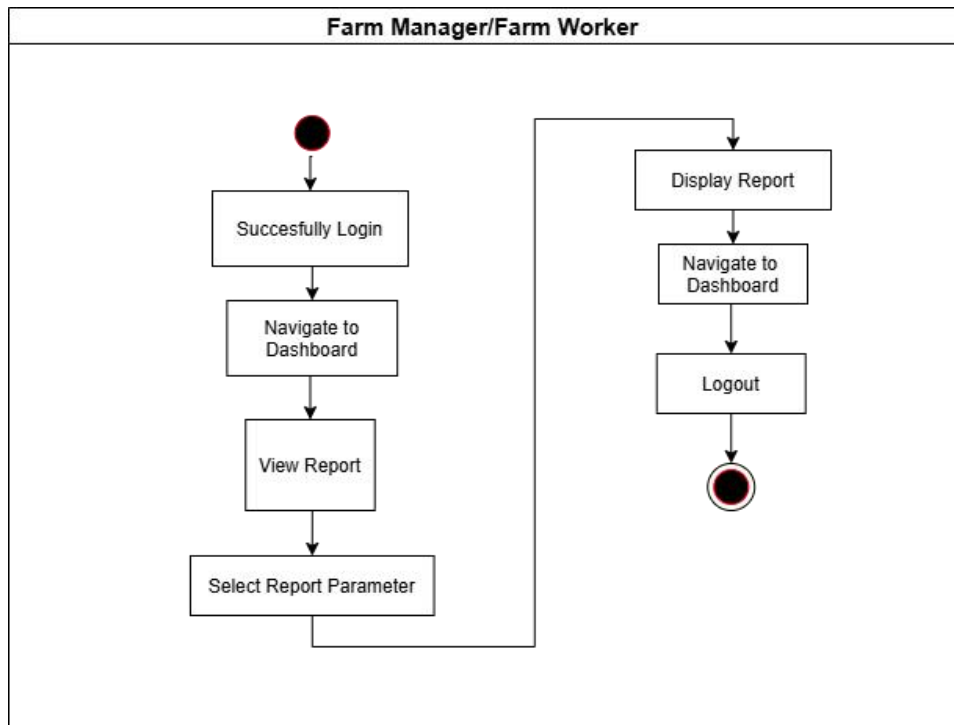


Figure 3.22 Activity Diagram for Use Case 8: Generate Report

3.4.3. Sequence Diagram

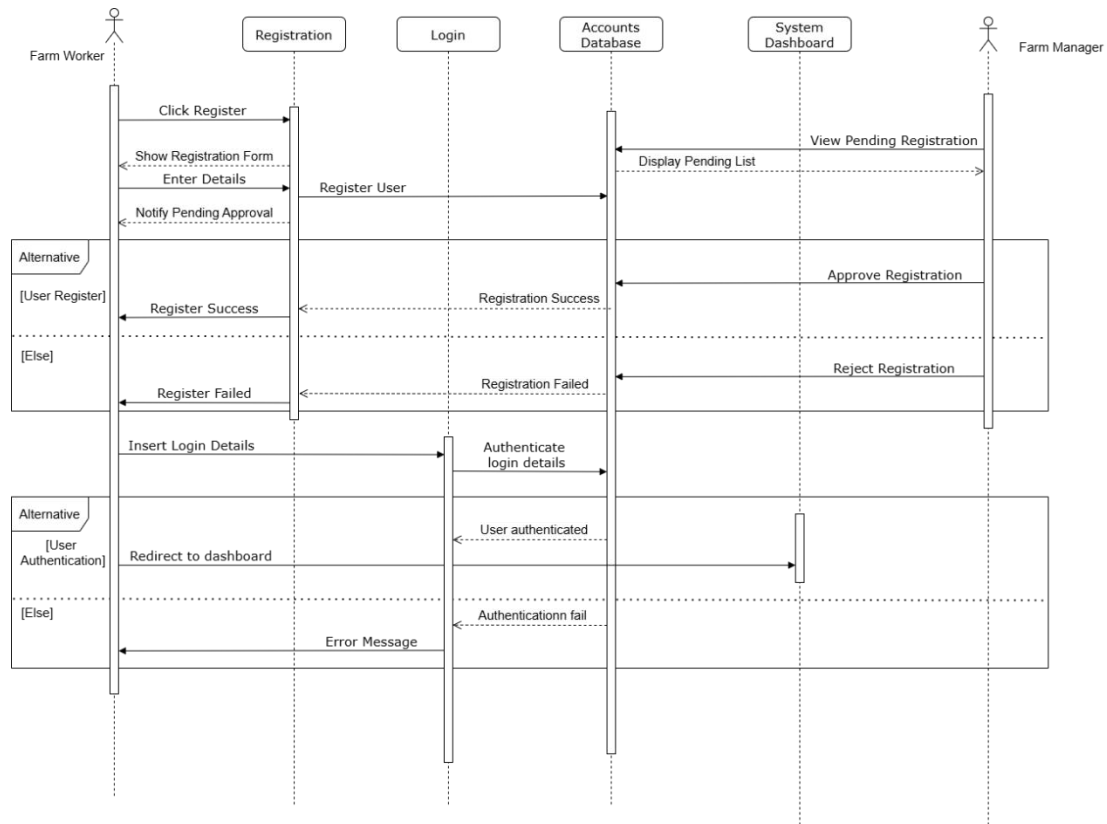


Figure 3.23 Sequence Diagram for Use Case 1: User Authentication

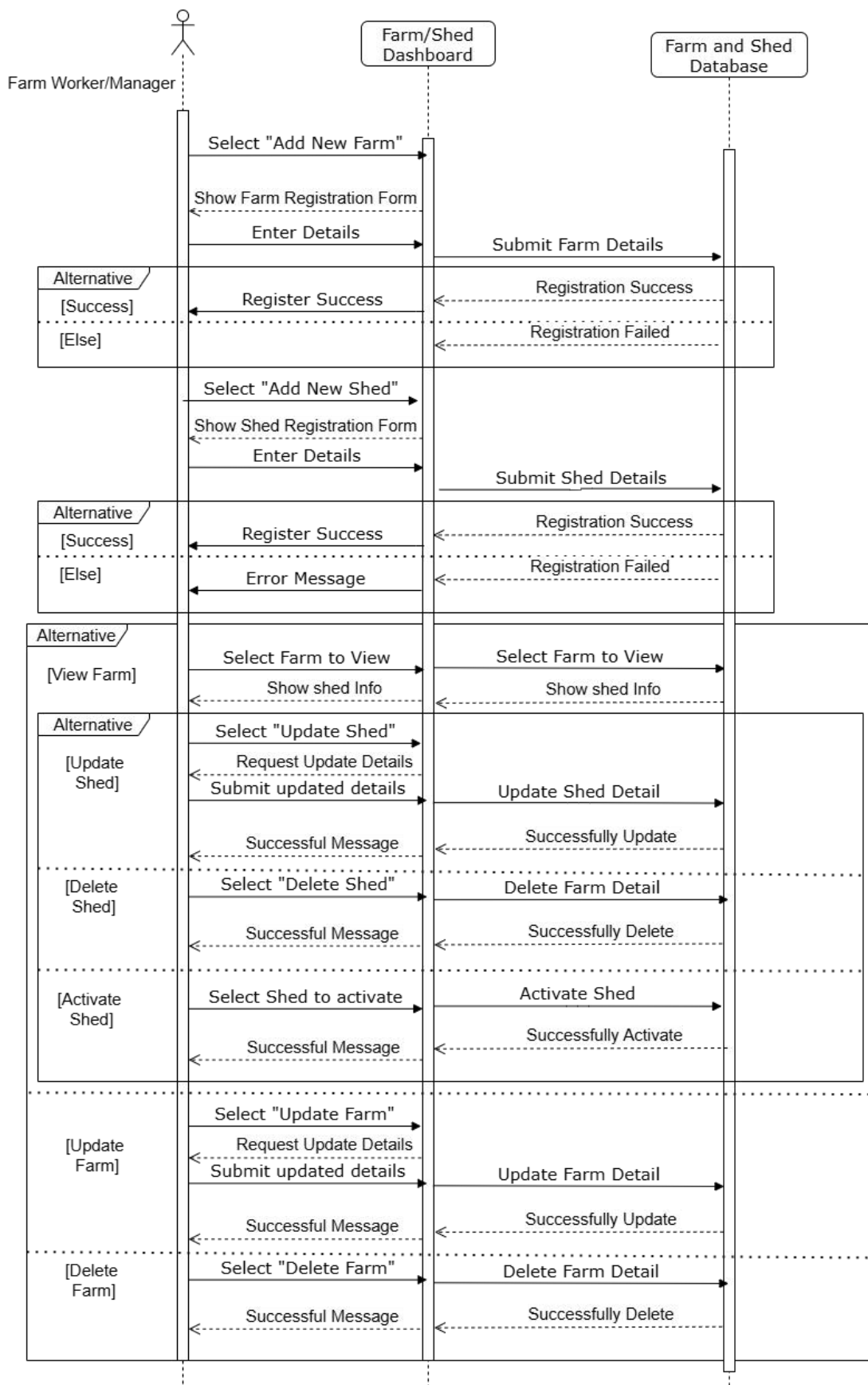


Figure 3.24 Sequence Diagram for Use Case 2: Manipulate Farm Information

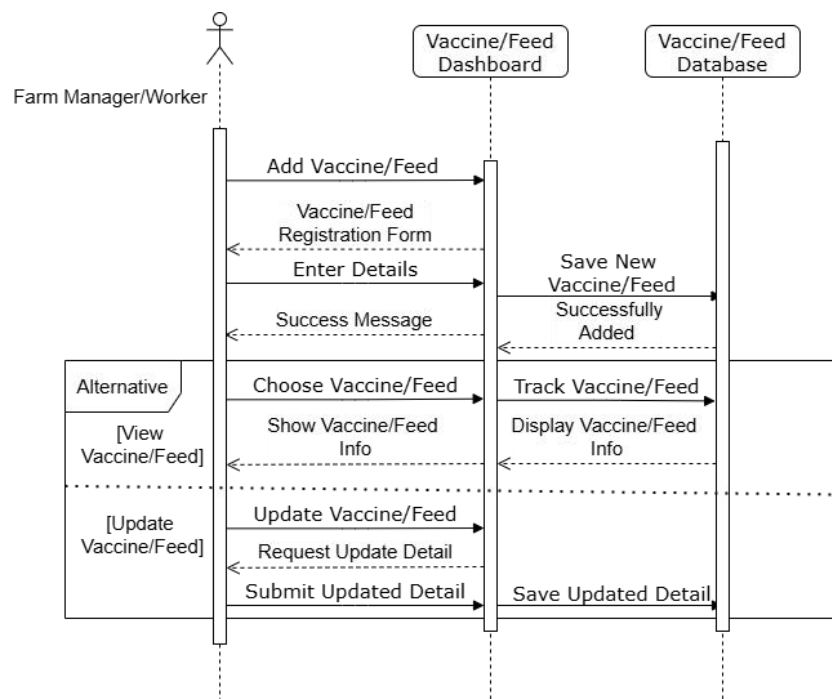


Figure 3.25 Sequence Diagram for Use Case 3: Add New Vaccine and Feed

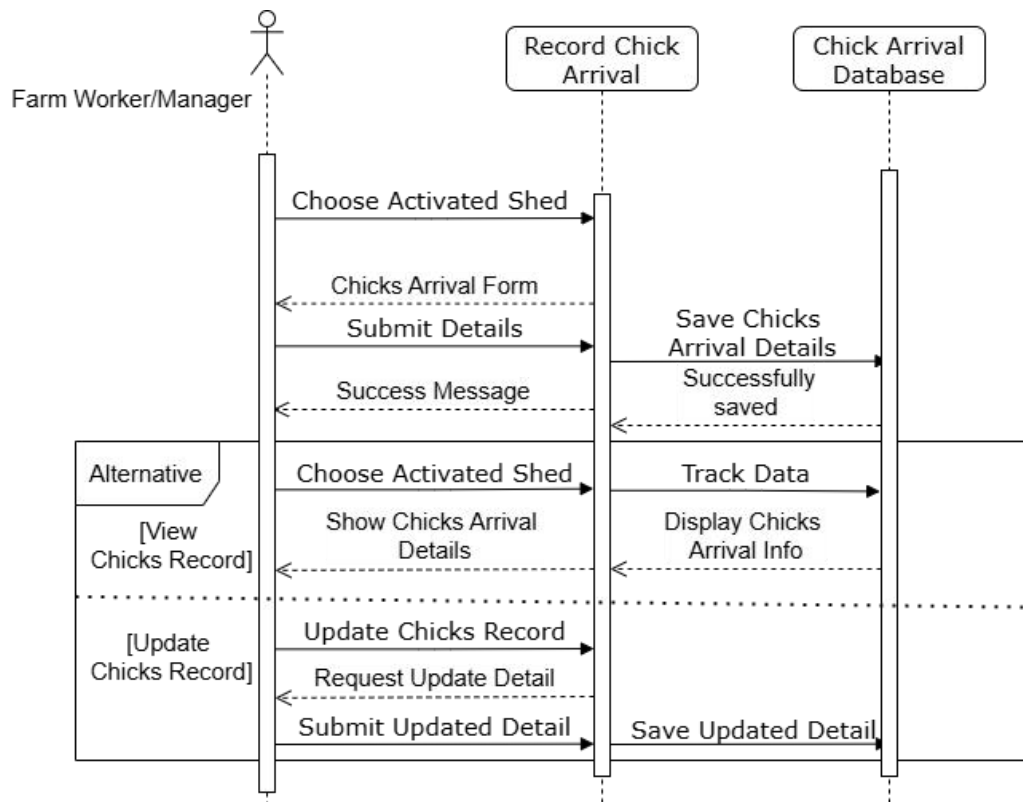


Figure 3.26 Sequence Diagram for Use Case 4: Record Chick Arrival

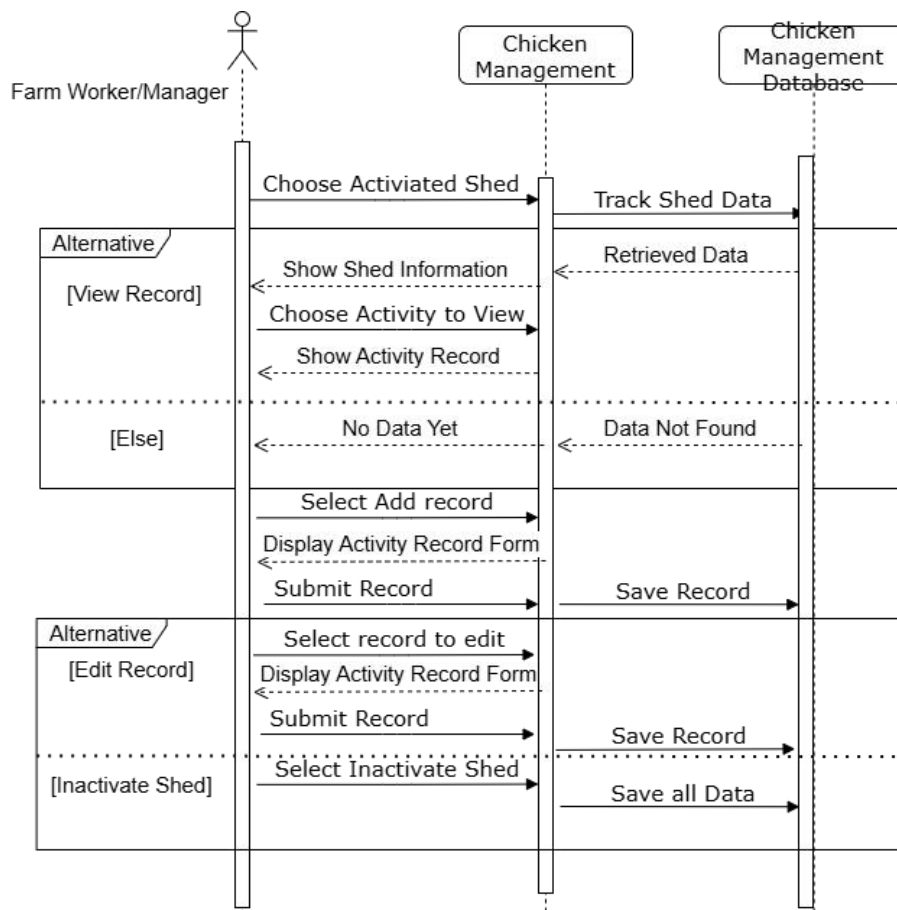


Figure 3.27 Sequence Diagram for Use Case 5: Chicken Management Activities

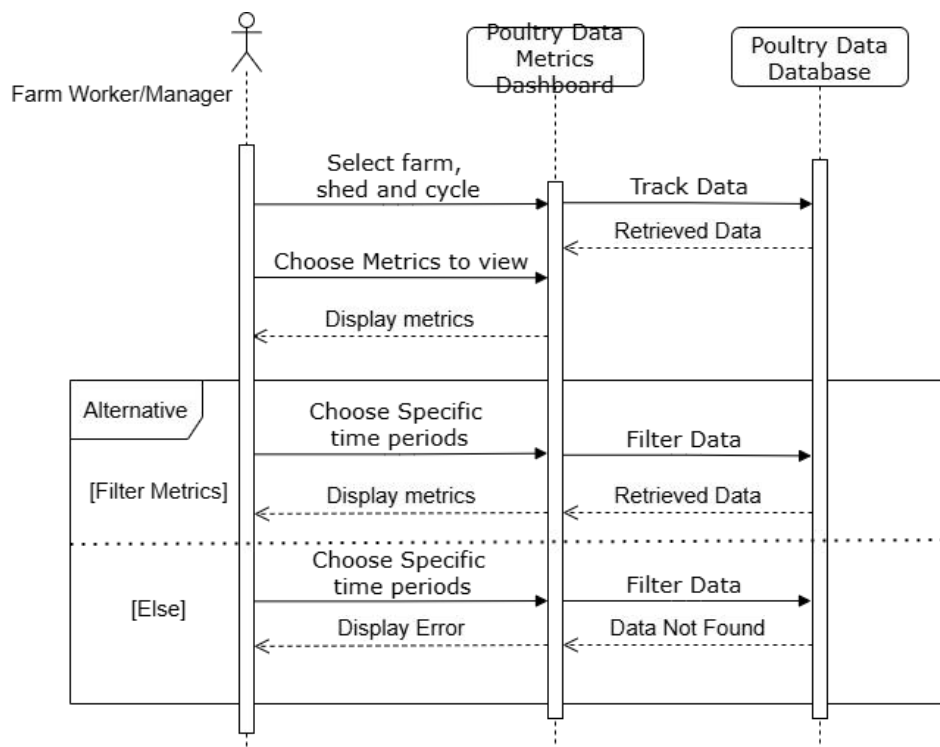


Figure 3.28 Sequence Diagram for Use Case 6: View Poultry Data Metrics

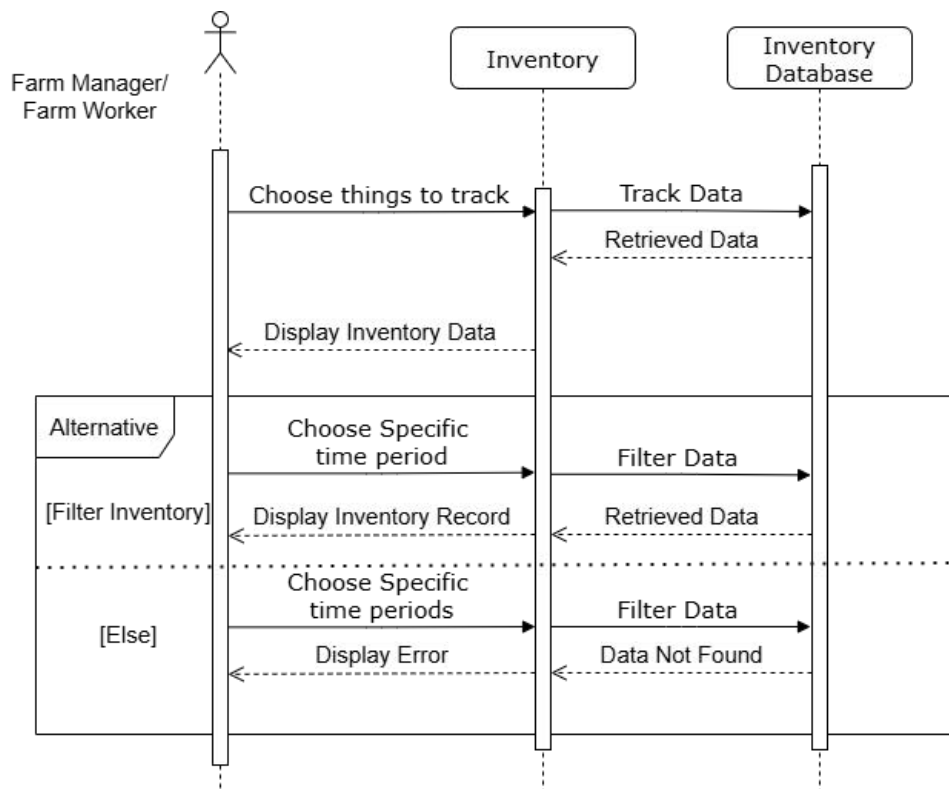


Figure 3.29 Sequence Diagram for Use Case 7: Track Inventory

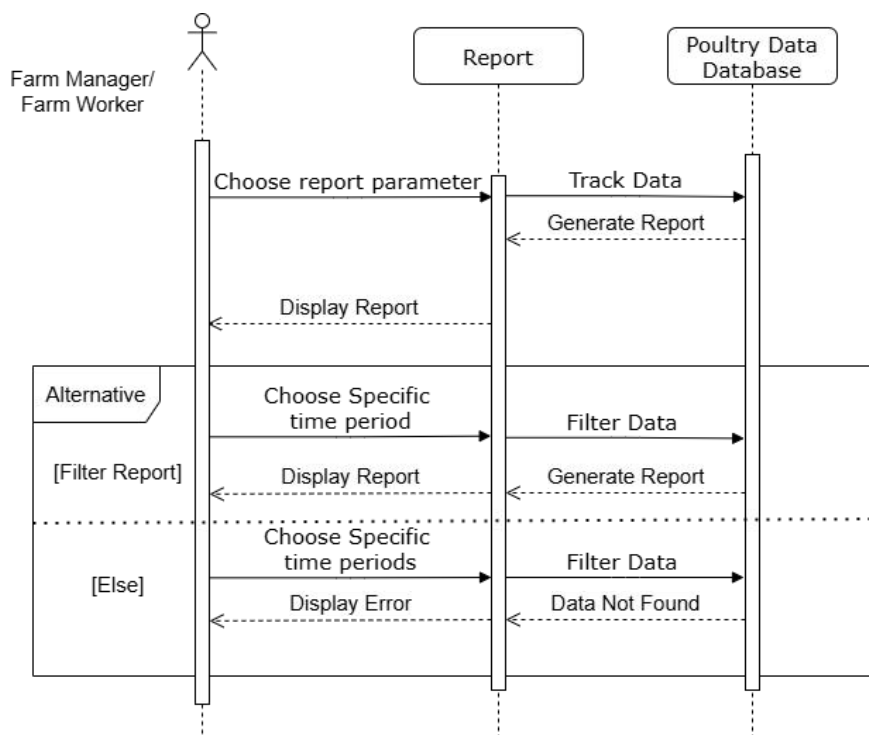


Figure 3.30 Sequence Diagram for Use Case 8: Generate Report

3.4.4. Data Dictionary

Table 5 Data Dictionary for Farm Worker

Variable Name	Data Type	Field Length	Constrain	Description
passport_number	String	20	Primary Key, Unique	Passport number used as the login ID for users
password	String	256	Must meet security criteria	User's password for system login
role	String	20	Must be one of predefined roles (e.g., Farm Worker, Manager)	Role of the user in the system
user_name	String	100	Not Null	Full name of the user
date_of_birth	Integer	-	Must be a positive integer	Age of the User
nationality	String	50	Not Null	Nationality of the user
gender	String	10	Must be one of predefined values (e.g., Male, Female, Other)	Gender of the user
phone_no	String	15	Optional	Phone number of the user (if applicable)
register_date	Date	-	Not Null	Date when the user registered
approval_status	String	20	Must be one of predefined values (e.g., Pending, Approved, Rejected)	Account approval status set by the manager

Table 6 Data Dictionary for Farm

Variable Name	Data Type	Field Length	Constrain	Description
farm_id	String	50	Primary Key, Unique	Unique identifier for each farm
farm_name	String	100	Must be unique, Not Null	Name of the farm

created_by	String	20	Foreign Key	Passport number used as the login ID for users
address	String	100	Not Null	Address location of the farm
farm_register_date	Date	-	Not Null	Date when the farm was registered

Table 7 Data Dictionary for Shed

Variable Name	Data Type	Field Length	Constrain	Description
farm_id	String	50	Foreign Key	Unique identifier for farm
shed_id	String	50	Primary Key, Unique	Unique identifier for each shed
shed_capacity	Integer	-	Must be a positive integer, Not Null	Maximum capacity of the shed
feeding_system_type	String	30	Must be one of predefined values (e.g., Manual, Automatic)	Type of feeding system used in the shed
shed_type	String	30	Must be one of predefined values	Status of the shed (e.g., Open, Closed)
shed_status	String	20	Must be one of predefined statuses	Status of the shed (e.g., Active, Inactive)
shed_register_date	Date	-	Not Null	Date when the shed was registered
created_by	String	20	Foreign Key	Passport number used as the login ID for users

Table 8 Data Dictionary for Flock

Variable Name	Data Type	Field Length	Constrain	Description
shed_id	String	50	Foreign Key	Unique identifier for each shed
flock_id	String	50	Primary Key, Unique	Unique identifier for each flock
chick_id	String	50	Foreign Key	ID for each chicks type
arrival_date	Date	-	Not Null	Date when the flock arrived
quantity	Integer	-	Must be a positive integer, Not Null	Number of chicks in the flock
status	String	20	Must be one of predefined statuses	Status of the flock (e.g., Active, Inactive)
complete_date	Date	-	Not Null	Date when the shed was complete
recorded_by	String	20	Foreign Key	Passport number used as the login ID for users

Table 9 Data Dictionary for Chick Type

Variable Name	Data Type	Field Length	Constrain	Description
chick_id	String	50	Primary Key	Unique identifier for each chick type
type_name	String	50	Not Null	Name of the chick type
supplier	String	50	Not Null	Supplier of the chick type

price	Float	-	Must be positive, Not Null	Price per unit of the chick type
-------	-------	---	-------------------------------	-------------------------------------

Table 10 Data Dictionary for Feed

Variable Name	Data Type	Field Length	Constrain	Description
feed_id	String	50	Primary Key, Unique	Unique identifier for each type of feed
feed_name	String	100	Not Null	Name of the feed
feed_type	String	50	Must be one of predefined types (e.g., Starter, Grower, Finisher)	Type/category of the feed
unit_weight	Integer	-	Must be positive, Not Null	Weight per unit (e.g., kg per bag)
price	Float	-	Must be positive, Not Null	Price per unit of feed

Table 11 Data Dictionary for Feed Record

Variable Name	Data Type	Field Length	Constrain	Description
id	String	50	Primary Key, Unique	Unique identifier for each type of feed record
flock_id	String	50	Foreign Key	Reference to the flock receiving the feed
feed_id	String	50	Foreign Key	Reference to the feed type used
quantity_used	Float	-	Not Null	Amount of feed used (in kg)

usage_date	Date	-	Not Null	Date when feed was given
recorded_by	String	50	Not Null	Passport number used as the login ID for users

Table 12 Data Dictionary for Vaccine

Variable Name	Data Type	Field Length	Constrain	Description
vaccine_id	String	50	Primary Key, Unique	Unique identifier for each type of vaccine
vaccine_name	String	50	Not Null	Name of the vaccine
disease_targeted	String	50	Not Null	Disease the vaccine protects against
vaccine_type	String	50	Not Null	Type of vaccine
dosage_amount	String	50	Not Null	Recommended dosage (e.g., 0.5ml/bird)
application_method	String	50	Not Null	How vaccine is administered (e.g., injection)

Table 13 Data Dictionary for Vaccine Record

Variable Name	Data Type	Field Length	Constrain	Description
id	String	50	Primary Key, Unique	Unique identifier for each type of feed record
flock_id	String	50	Foreign Key	Reference to the flock vaccinated
vaccine_id	String	50	Foreign Key	Reference to the vaccine used
quantity_applied	Integer	-	Not Null	Amount of vaccine used
applied_date	Date	-	Not Null	Date when vaccine was administered

recorded_by	String	50	Not Null	Users who recorded the vaccination
-------------	--------	----	----------	------------------------------------

Table 14 Data Dictionary for Weight Record

Variable Name	Data Type	Field Length	Constrain	Description
id	String	50	Primary Key, Unique	Unique identifier for each weight record
flock_id	String	50	Foreign Key	Reference to the flock being measured
number_of_chicks_weighted	Integer	-	Not Null	Number of chicks sampled for weight
total_weight_measured	Float	-	Not Null	Combined weight of sampled chicks (in kg)
average_weight	Float	-	Not Null	Calculated average weight per chick (in kg)
measured_date	Date	-	Not Null	Date when measurements were taken
recorded_by	String	50	Not Null	Users who recorded the measurements

Table 15 Data Dictionary for FCR Record

Variable Name	Data Type	Field Length	Constrain	Description
id	String	50	Primary Key, Unique	Unique identifier for each fcr record
flock_id	String	50	Foreign Key	Reference to the flock being measured

recorded_date	Date	-	Not Null	Date when FCR was calculated
fcv_value	Float	-	Not Null	Feed Conversion Ratio value

Table 16 Data Dictionary for Mortality Record

Variable Name	Data Type	Field Length	Constrain	Description
id	String	50	Primary Key, Unique	Unique identifier for each weight record
flock_id	String	50	Foreign Key	Reference to the flock being measured
reason_id	String	50	Not Null	Reference to mortality reason
mortality_amount	Integer	-	Not Null	Number of birds lost
date_recorded	Date	-	Not Null	Date when mortality occurred
recorded_by	String	50	Not Null	Users who recorded the measurements

Table 17 Data Dictionary for Feed Inventory

Variable Name	Data Type	Field Length	Constrain	Description
inventory_id	String	50	Primary Key, Unique	Unique identifier for each feed inventory record
farm	String	50	Foreign Key	Reference to farm
vaccine	String	50	Foreign Key	Reference to

				vaccine type
Shed	String	50	Foreign Key	Reference to storage shed
quantity_arrived	Integer	-	Not Null	Initial quantity received (in kg)
quantity_left	Integer	-	Not Null	Remaining quantity (in kg)
arrival_date	Date	-	Not Null	Date when feed was received
recorded_by	String	50	Not Null	Users who recorded the measurements

Table 18 Data Dictionary for Vaccine Inventory

Variable Name	Data Type	Field Length	Constrain	Description
inventory_id	String	50	Primary Key, Unique	Unique identifier for each feed inventory record
farm	String	50	Foreign Key	Reference to farm
feed	String	50	Foreign Key	Reference to associated feed
Shed	String	50	Foreign Key	Reference to storage shed
quantity_arrived	Integer	-	Not Null	Initial vials/doses received
quantity_left	Integer	-	Not Null	Remaining vials/doses
arrival_date	Date	-	Not Null	Date when vaccine was received
recorded_by	String	50	Not Null	Users who recorded the measurements

Table 19 Data Dictionary for Alert Threshold

Variable Name	Data Type	Field Length	Constrain	Description
alert_id	String	50	Primary Key, Unique	Unique identifier for each alert threshold
chick_id	String	50	Foreign Key	Reference to chick breed/type
doc_age	Integer	-	Not Null	Days-old-chicken age threshold
expected_feed	Float	-	Not Null	Expected feed consumption (kg/bird
expected_weight	Float	-	Not Null	Expected average weight (kg)
fcr_threshold_percent	Float	-	Not Null	Acceptable FCR variance percentage

Table 20 Data Dictionary for Mortality Reason

Variable Name	Data Type	Field Length	Constrain	Description
reason_id	String	50	Primary Key, Unique	Unique identifier for each mortality reason
reason_name	String	50	Not Null	Name of the reason

3.4.5. Entity Relationship Diagram (ERD)

The ERD visually represents the structure of the data and relationships within PoultryPlus management system.

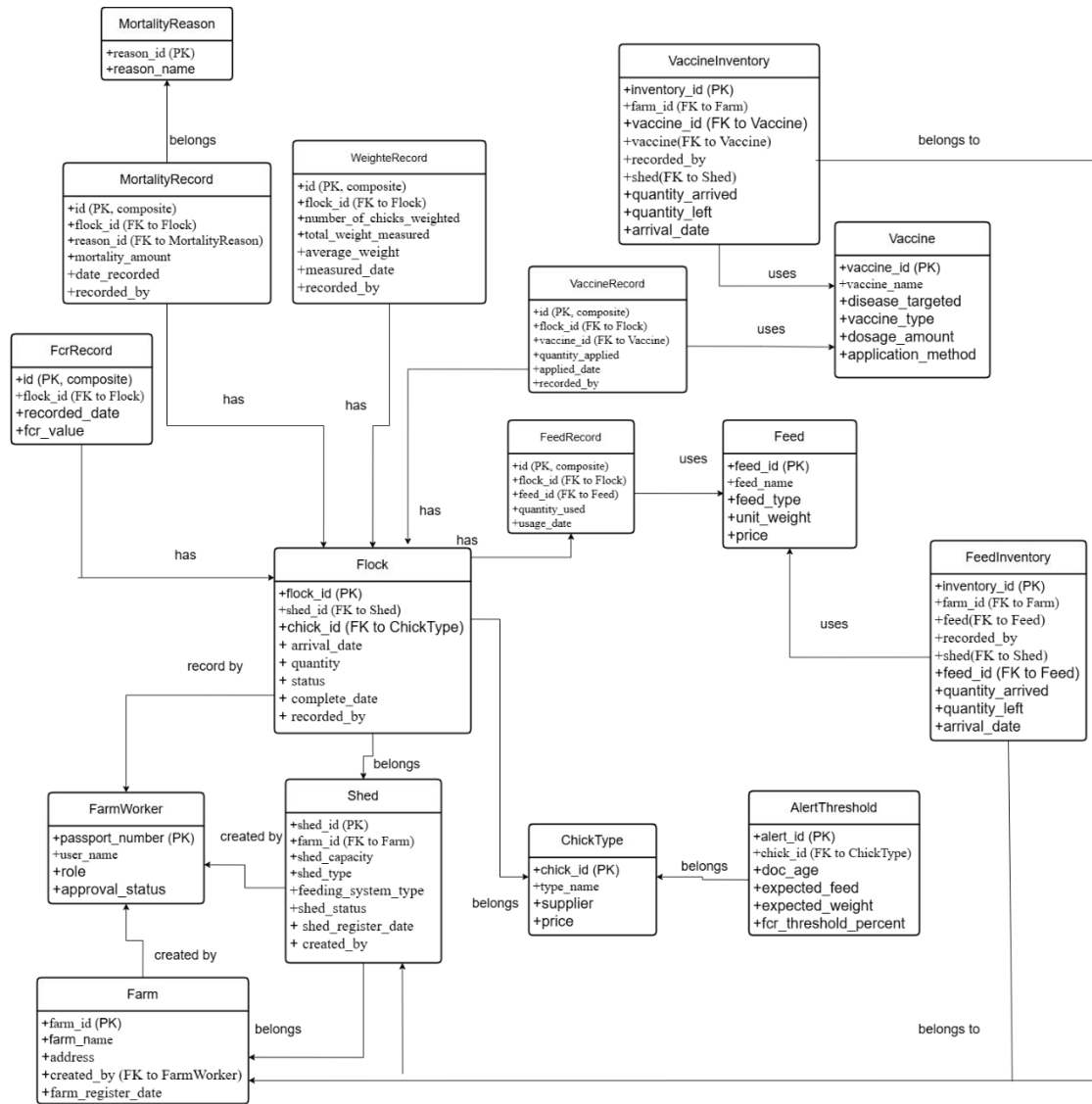


Figure 3.31 ER Diagram for PoultryPlus System

3.4.6. Graphical User Interface Design

The Graphical User Interface (GUI) design is a critical aspect of the PoultryPlus system, as it directly influences user experience and interaction. A well-designed GUI facilitates intuitive navigation, making it easier for users to access the various functionalities of the application. This section outlines the principles and methodologies employed in designing the interface, emphasizing usability, accessibility, and aesthetics. By prioritizing a user-centered approach, we aim to create an engaging and efficient environment that enhances the productivity of poultry management for all users, from farmers to administrators. Through thoughtful layout,

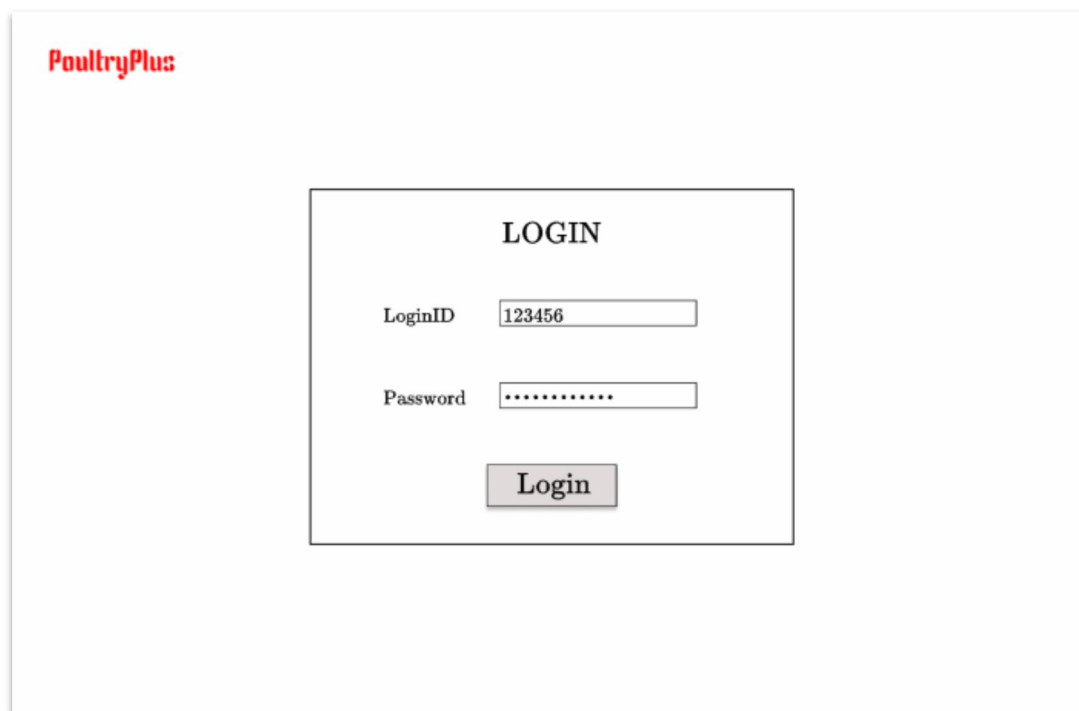
color schemes, and interactive elements, the GUI will serve as a seamless bridge between users and the underlying system functionalities.



The image shows a web form titled "PoultryPlus" in red text at the top left. The form itself is titled "Register" in bold black text. It contains several input fields: "Name" with the value "Ikbar", "Age" with "27", "Nationality" with "Indonesia", "Gender" with radio buttons for "Male" (selected) and "Female", "PhoneNo." with "01199999999", "PassportNo." with "FB0097K", "Role" with radio buttons for "Worker" (selected) and "Manager", and "Password" with "fhjksnd". A "Register" button is located at the bottom of the form.

Register	
Name	Ikbar
Age	27
Nationality	Indonesia
Gender	☒ Male ☐ Female
PhoneNo.	01199999999
PassportNo.	FB0097K
Role	☒ Worker ☐ Manager
Password	fhjksnd
Register	

Figure 3.32 GUI Design 1: Register Page



The image shows a web form titled "PoultryPlus" in red text at the top left. The form itself is titled "LOGIN" in bold black text. It contains two input fields: "LoginID" with the value "123456" and "Password" with a masked value ".....". A "Login" button is located at the bottom of the form.

LOGIN	
LoginID	123456
Password	
Login	

Figure 3.33 GUI Design 2: Login Page

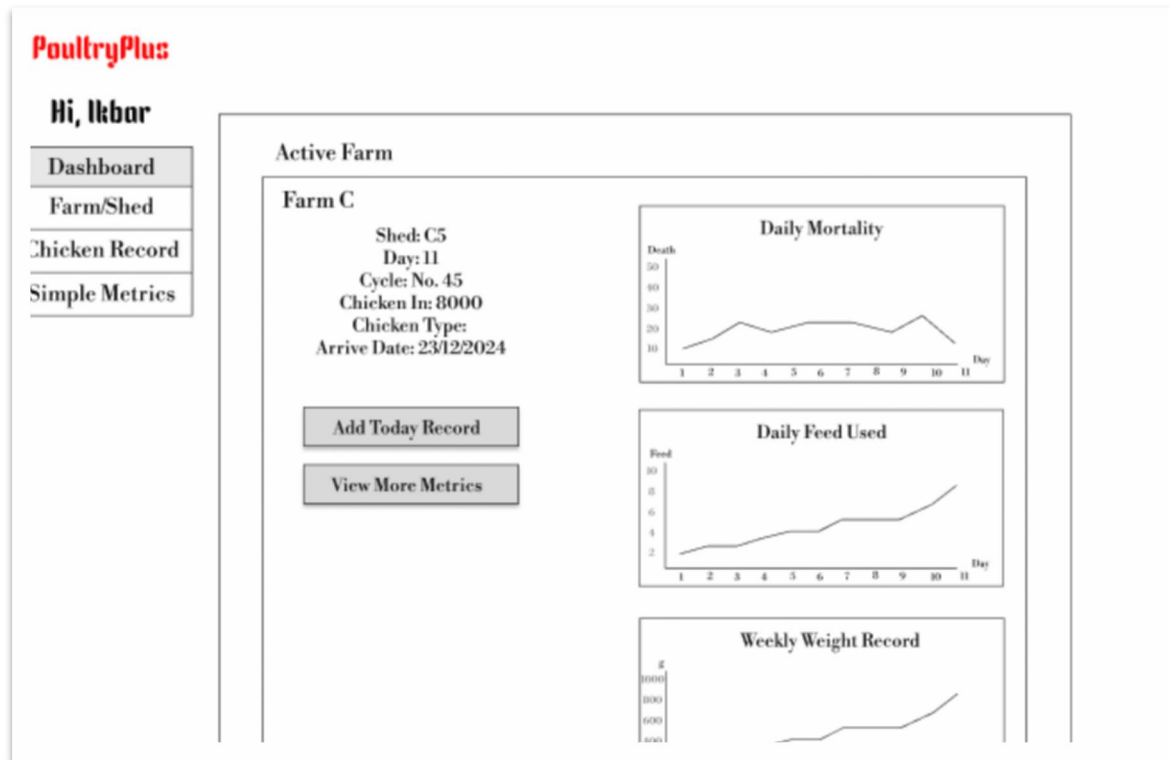


Figure 3.34 GUI Design 3: Dashboard

3.5 Summary

Chapter 3 provides a comprehensive overview of the design and development framework for the PoultryPlus system. It begins by outlining the system's objectives and the overall architecture, emphasizing the integration of modern technologies to create an efficient poultry management solution.

The chapter delves into the software requirements, detailing the necessary components such as the operating system, database server, development languages, and tools, ensuring a robust foundation for the application. It also highlights the selection of a suitable web framework, API protocols, and version control systems to facilitate seamless collaboration and development.

Furthermore, the chapter addresses the importance of user experience through a dedicated section on Graphical User Interface (GUI) design. It emphasizes the principles of

usability and accessibility, aiming to create an intuitive interface that enhances user interaction and engagement.

Overall, Chapter 3 presents a well-structured approach to the design and implementation of the PoultryPlus system, laying the groundwork for subsequent development phases and ensuring that user needs are met through thoughtful planning and execution.

Chapter 4 : SYSTEM IMPLEMENTATION

4.1 Introduction

In this chapter, the focus is placed on the implementation phase of the proposed PoultryPlus system. It showcases how each module was developed with the selected technology stack—Django REST Framework (DRF) for the backend and React 18 with Vite for the frontend—strictly following the analysis and design delivered in Chapter 3. Key aspects covered are:

- System architecture and technical specifications
- Detailed module implementations with code explanations
- Database design and API integrations
- Role-based access control (Manager vs Worker)
- Complete user-interface walkthrough with screenshots

All components were developed to meet both the functional and non-functional requirements, resulting in a robust, web-based poultry farm management solution.

4.2 System Implementation

4.2.1. Installation

Backend (Django REST Framework)

The backend is built using Django and Django REST Framework (DRF) for API development. Key libraries used include:

Table 21 Django Backend Installation

Library	Version	Purpose
Django	4.2	Core backend framework
Django REST Framework	3.14	API development

django-cors-headers	4.2	Cross-origin requests
---------------------	-----	-----------------------

Frontend (React.js)

Table 22 React Frontend Installation

Library	Version	Purpose
React	18.2	Core frontend framework
React Router DOM	6.14	Navigation and routing
Axios	1.4	HTTP requests to Django backend

The system utilizes Django's ORM (Object-Relational Mapping) to manage database models, with data access and updates handled through RESTful APIs. This architecture follows the pattern described by Garner (2021) in their React-Django integration guide:

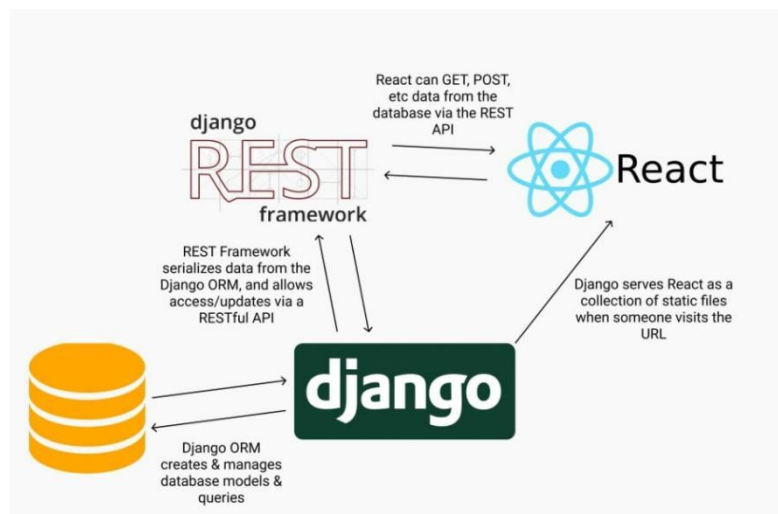


Figure 4.1 Django ORM and REST API data flow (Adapted from Garner, 2021)

Key Components Shown:

1. Django Models – Define database structure (e.g., Farm, Shed)
2. REST Framework – Exposes models as APIs (e.g., GET /api/farms/)
3. React Frontend – Consumes APIs via Axios (as implemented in Sections 4.3.1–4.3.2)

4.2.2. Project File Structure

This section details the logical structure of backend and frontend folders, showing organization by modules and components.

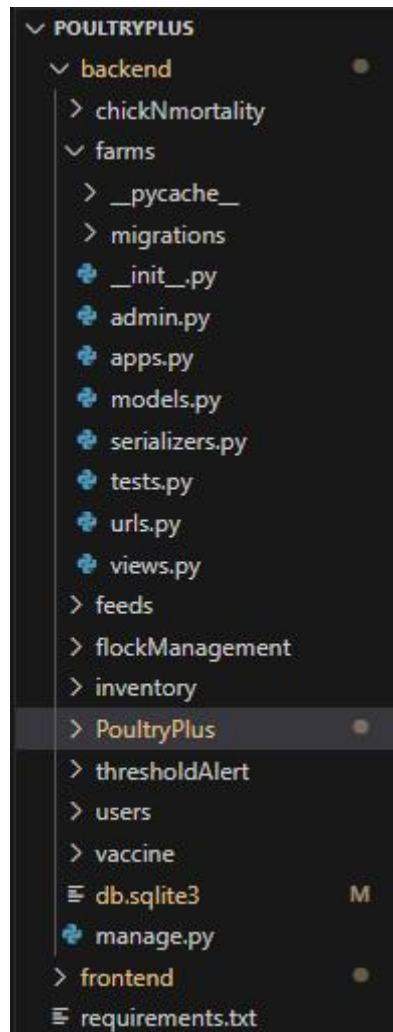


Figure 4.2 Backend File Structure as seen in Visual Studio Code

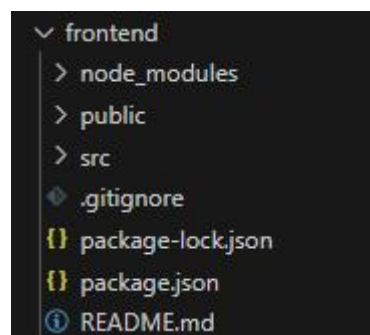


Figure 4.3 Frontend File Structure as seen in Visual Studio Code

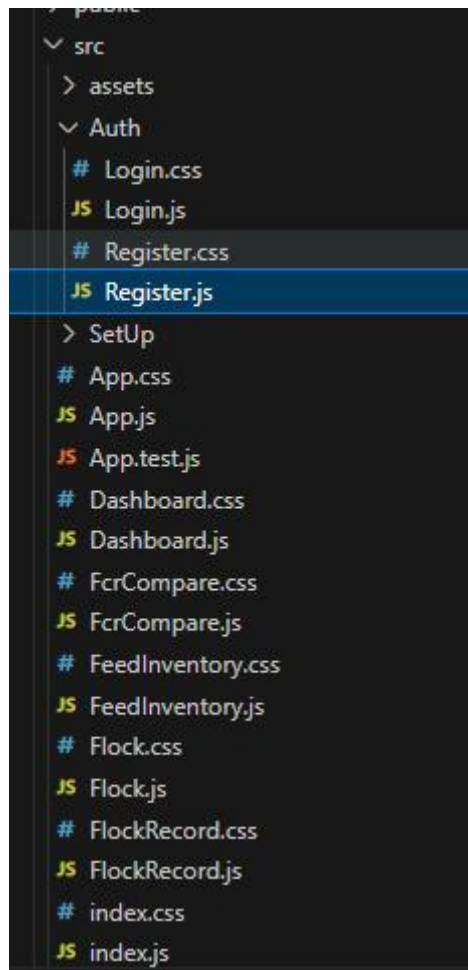


Figure 4.4 Frontend Design File Structure as seen in Visual Studio Code

4.2.3. Coding — Front-to-Back Data Flow

To illustrate how the React frontend exchanges data with the Django backend, this section walks through the user-registration path. All full listings are placed in Appendix A and referenced here by figure numbers.

To demonstrate frontend-backend integration, we trace the user registration workflow:

- 1.

When a worker fills the registration form and presses Register, the component (Figure 4.2)

gathers the input into a userData object and issues. (Full code refers to Appendix A Figure A-1.)

```
const response = await fetch('http://localhost:8000/api/register/', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify(userData),
});
```

Figure 4.5 React <Register> component code

Because the request is JSON and cross-origin, CORS is permitted by django-cors-headers in the backend settings. Axios sends a POST request to /api/register/

(Step 2) Backend Routing (Django)

The urls.py (Figure 4.3) maps /api/register

```
from django.urls import path
from .views import register_user, login_user
from .views import user_profile

urlpatterns = [
    path('register/', register_user, name='register_user'),
    path('login/', login_user, name='login_user'),
    path('users/<str:passport_number>', user_profile, name='get_user_profile'),
]
```

Figure 4.6 Users App urls.py

so the HTTP POST is routed to the corresponding Django view.

(Step 3) Backend Processing (Django View)

register_user in views.py (Appendix A · Figure A-2)

1. Deserialises request.body (json.loads).
2. Checks for duplicate passport numbers.
3. Hashes the plaintext password with make_password.

4. Calculates age from date_of_birth.
5. Creates a new FarmWorker instance (model definition in Appendix A · Figure A-3).
6. Returns a JSON success message (201 Created) or an error status.

(Step 4) Frontend Response Handling (React)

Back in React, the promise resolves:

```
if (response.ok) {
  alert("Registered successfully!");
  navigate('/login'); // Redirect to login page
} else {
  setError(result.error || 'Registration failed');
}
```

Figure 4.7 Response Handling in Register.js

Any network or validation error is surfaced instantly in the UI, while the Django view's status code controls the branch that runs.

This diagram illustrates the end-to-end data flow during user registration:

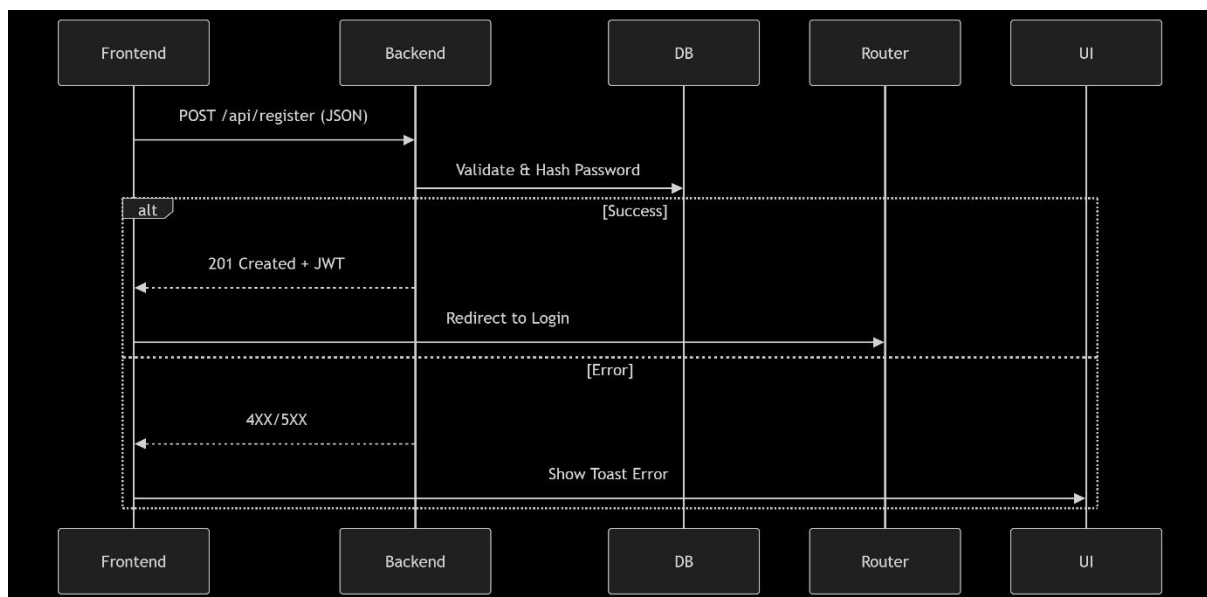


Figure 4.8 Sequence Diagram for The Registration Flow

4.2.4. Implementation of PoultryPlus

The PoultryPlus system was implemented as a modular web-based application using React.js for the frontend and Django REST Framework for the backend. The design follows a role-based access control structure, offering different functionalities for Manager and Worker users. The modules developed support farm operations such as inventory tracking, mortality monitoring, and real-time data entry.

Below is an overview of each implemented module, along with screenshots and relevant code snippets (where appropriate). More extensive code listings are provided in the Appendix.

4.2.4.1 Authentication Module

Login Module

The login module is the entry point for all users of the PoultryPlus system. It is developed using React for the frontend and Django REST Framework for the backend authentication. The login page ensures that only authorized users—Manager, or Worker—can access the system based on their assigned roles.

Users are required to input their Passport Number and Password to log in. Upon form submission, the input data is sent to the backend API for validation. If the credentials are correct and the account has been approved by the manager, the user is redirected to the respective dashboard based on their role. If the account status is still pending or has been rejected, a relevant alert message is shown. Form validation is included to detect and display error messages for incorrect or empty input fields. Figure 4.9 shows the React frontend code implementation of the login module, while Figure 4.10 illustrates the user interface of the login page.

```

    if (data.status === 'approved') {
      localStorage.setItem('passport_number', userData.passport_number);
      navigate('/dashboard');
    } else if (data.status === 'pending') {
      alert('Your account is pending approval. Please wait for your manager to approve it.');
```

Figure 4.9 Login Frontend Code Snippet (React)

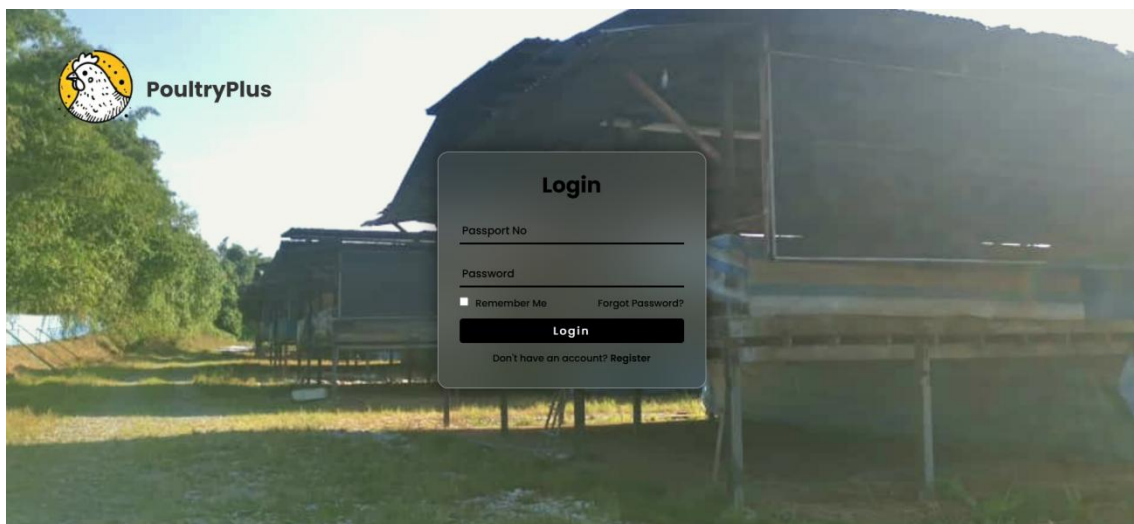


Figure 4.10 Login Page

Register Module

The Register Module allows new workers to create an account in the PoultryPlus system through a structured registration form built with React and integrated with a Django REST API. The form collects essential information such as passport number, full name, date of birth, nationality, gender, phone number (auto-prefixed with +60), and password. The phone number field includes formatting logic to ensure the number begins with Malaysia’s country code. Password and confirm password fields are cross-checked for consistency to prevent submission errors. The gender field is implemented using radio buttons, while nationality is handled with a dropdown menu offering common options and an “Others” fallback.

On form submission, the input is validated on the client side, and then the data is sent to the backend via a POST request to the `/api/register/` endpoint. If the server responds successfully, a success message is shown, and the user is redirected to the login page. If any error occurs, such as a duplicate passport number or API issue, a relevant error message is displayed below the form. The button is also disabled while the submission is being processed to prevent duplicate entries. This module is exclusively for worker registration; manager accounts are not created via this form but are manually pre-set in the backend by the system maintainer. Figure 4.11 shows the Register Page input form layout designed for new worker account creation.

The image shows a mobile application interface for a registration form. The background is a photograph of a rural landscape with a thatched-roof building and trees. In the top left corner, there is a logo for 'PoultryPlus' featuring a stylized chicken head. Overlaid on the center of the image is a dark, semi-transparent 'Register' form. The form contains the following fields and elements:

- Register** (Section Header)
- Passport No (Text input field)
- Username (Text input field)
- Date of Birth (Text input field with a calendar icon)
- Nationality (Dropdown menu with 'Select Nationality' text)
- Phone Number (+60) (Text input field)
- Gender (Radio buttons for Male and Female)
- Password (Text input field)
- Confirm Password (Text input field)
- Register** (Black button with white text)
- Already have an account? Login (Text link)

Figure 4.11 Register Page Input Form

4.2.4.2 Dashboard Module

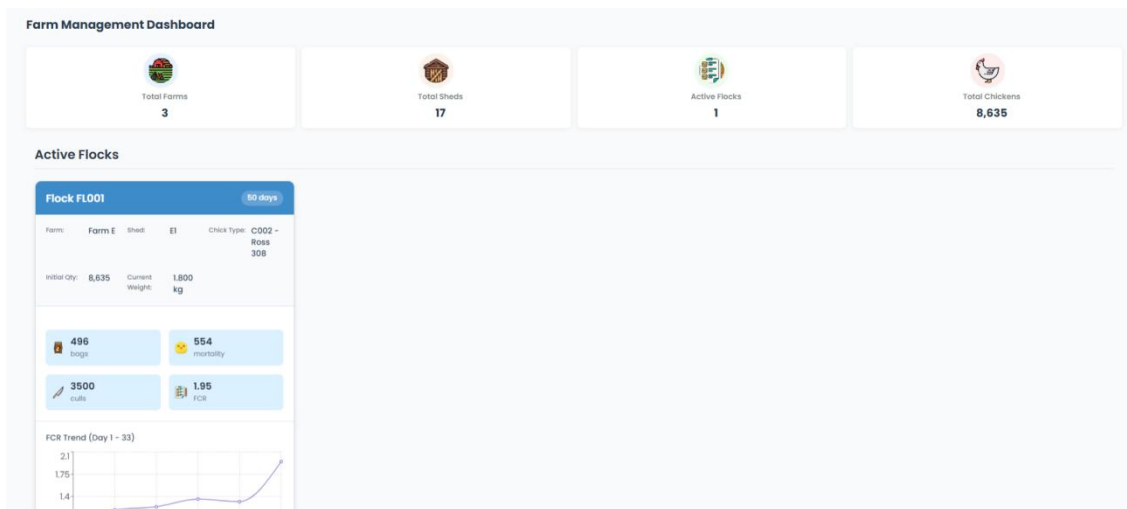


Figure 4.12 PoultryPlus Dashboard

The Dashboard Module, as illustrated in Figure 4.12, serves as the central hub for users, providing real-time insights into poultry operations through summarized metrics and visual analytics. Built with React and integrated with Django REST Framework, it displays four key summary cards—Total Farms, Total Sheds, Active Flocks, and Total Chickens—with icons for quick recognition. Below the summaries, an Active Flocks section lists each flock with detailed metrics (farm/shed location, chick type, initial quantity) and performance indicators (feed consumption, mortality/cull counts, current weight, and FCR). Interactive mini charts visualize FCR trends, daily mortality, and culling rates using Recharts, with tooltips for precise day-to-day analysis. Clicking any flock card navigates to its dedicated management page. The module fetches data asynchronously from multiple API endpoints, handles errors gracefully with loading spinners and retry options, and adapts layouts for desktop and mobile views. This design enables users to monitor farm health, spot inefficiencies, and drill into problem areas briefly.

4.2.4.3 Users Module

The Workers Module provides a streamlined interface for managing worker profiles within the system. Each worker can access their personal profile to view and update essential information such as passport number, username, date of birth, nationality, and phone number. The interface offers two modes: view mode for displaying information in a read-only format, and edit mode, which allows workers to update specific fields. Changes made are sent to the backend API to ensure that the data remains current and synchronized. For users with the manager role, the module extends functionality to include an overview and management of all registered workers. Managers are presented with a list of workers, from which they can select an individual to view their profile, make updates, or perform administrative tasks such as password resets. This empowers managers to maintain accurate records and assist workers in profile maintenance or account recovery when necessary. Figure 4.13 illustrates a worker's profile interface, highlighting the switch between view and edit modes. Figure 4.14 displays the manager's view with a list of workers and integrated controls for resetting passwords.

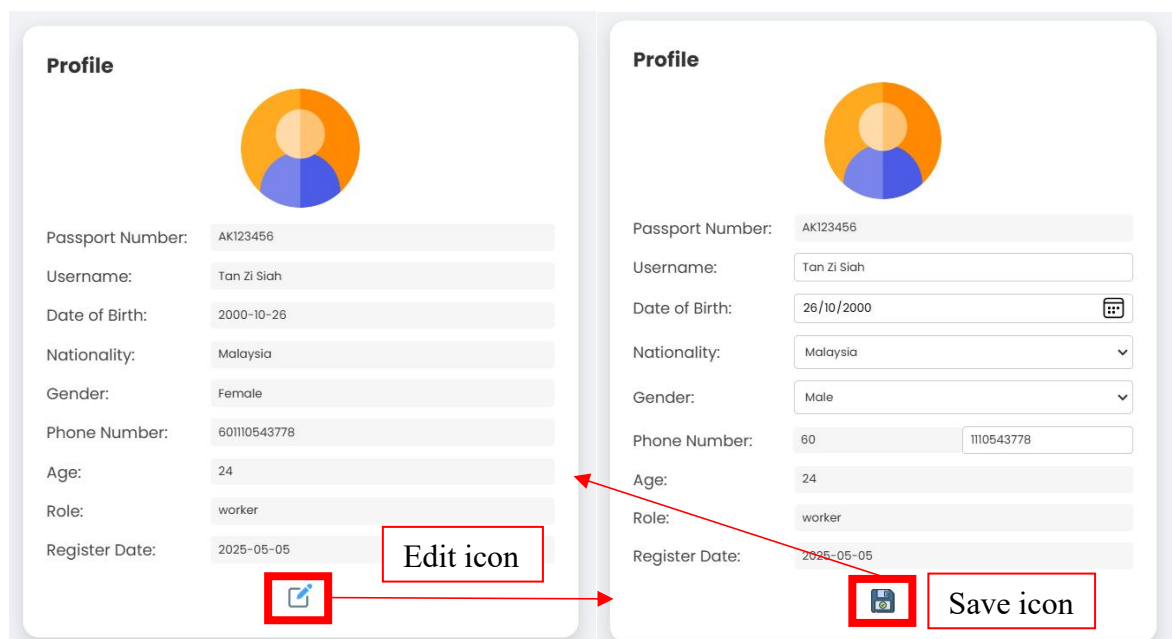


Figure 4.13 User profile interface showing edit/view modes

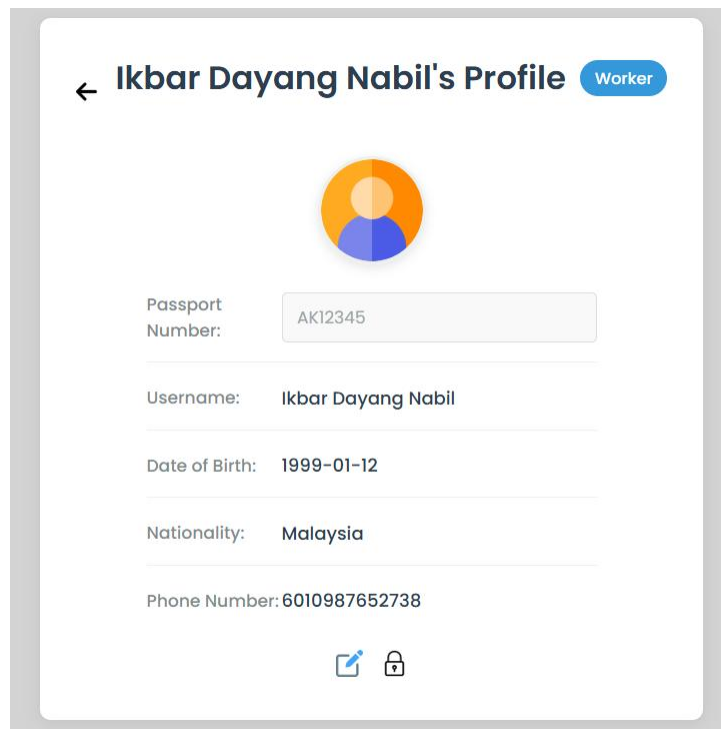


Figure 4.14 Manager view with worker list and password reset controls

4.2.4.4 Setup Module

The Setup Module in the system encompasses several key sections: Farm List, Feed Type, Shed List, Vaccine Type, and Chick Mortality. These sections serve a common purpose in managing essential data for the poultry farm, with similar functionality for adding, editing, and deleting records. Each section is designed to allow users to efficiently manage and update the information relevant to their operations.

Farm Register Module

This section lets users view, add, edit, and delete farms, enabling them to manage farm details such as names, addresses, and related statistics.

All Farm

Farm Information

Farm Summary

Total Farms

3

Total Sheds

12

Farms with Active Shed

2

Farm List

Add Farm
Sort by Newest

Farm ID	Farm Name	Address	Created By	Registered Date	No. of Sheds	Action
F003	Farm G	Sungai Rait Simpang 7	Tan Zi Siah	2025-05-02	5 View Shed	
F001	Farm C	Sungai Rait Simpang 1B	Ikbar	2025-05-01	6 View Shed	
F002	Farm E	Sungai Rait Simpang 1A	Ikbar	2025-04-23	1 View Shed	

Figure 4.15 Farm Listing Page

Figure 4.14 displays the Farm Listing Page, where users are presented with a Farm Summary section providing an overview of the total number of farms, sheds, and farms with active sheds. Below the summary, the Farm List is shown in a table format, including details like Farm ID, Farm Name, Address, Created By, Registered Date, and number of sheds. Users can click the *View Shed* button to see more details.

To add a new farm, users click the Add Farm button, which opens a modal—shown in Figure 4.15—where they can input the farm name and address. The Created By and Register Date fields are auto filled. Editing a farm works similarly by clicking the edit icon, which pre-fills the modal with existing data. Users can then update and save the changes.

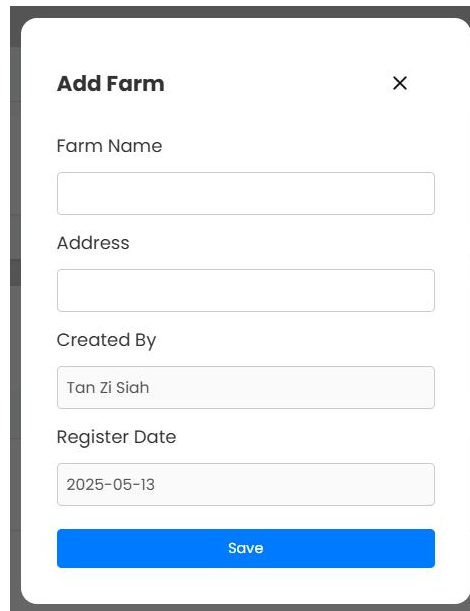
The image shows a modal window titled "Add Farm" with a close button (X) in the top right corner. Inside the modal, there are four input fields: "Farm Name" (empty), "Address" (empty), "Created By" (pre-filled with "Tan Zi Siah"), and "Register Date" (pre-filled with "2025-05-13"). At the bottom of the modal is a blue button labeled "Save".

Figure 4.16 Farm Modal (Add/Edit)

In case a user wants to delete a farm, Figure 4.16 illustrates the Delete Confirmation Alert, which appears after clicking the delete icon. The action proceeds only upon user confirmation.

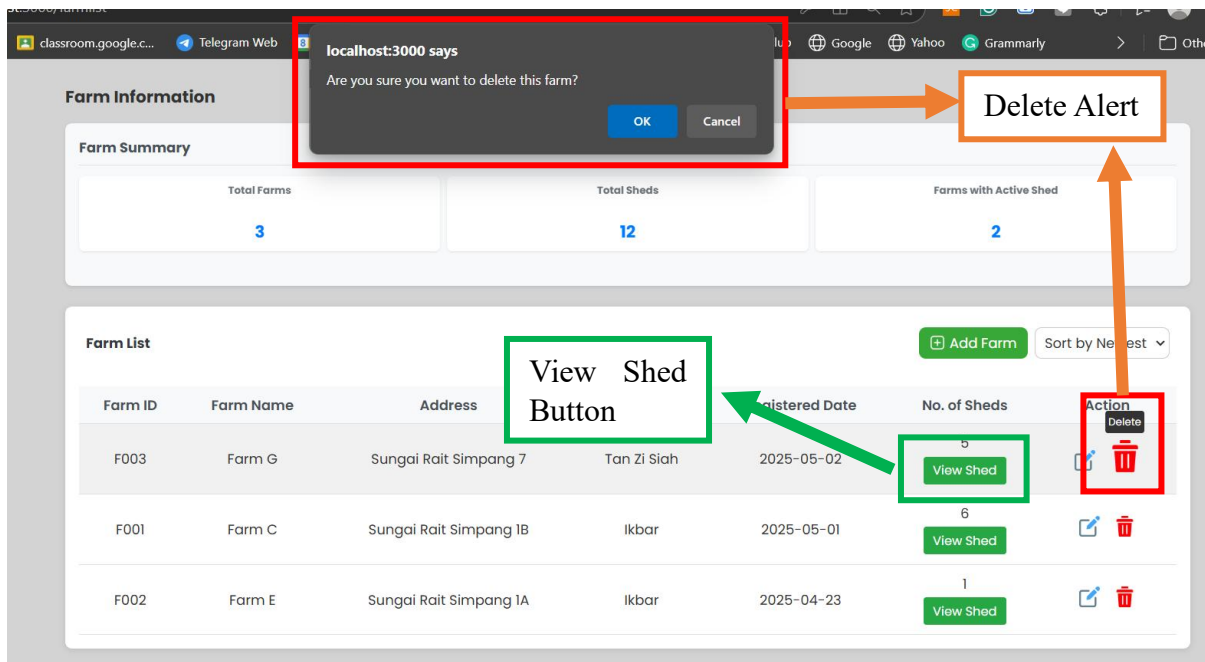


Figure 4.17 Delete Confirmation Alert

The user can also sort the farms by Newest or Oldest using a dropdown at the top-right of the list. Additionally, by clicking the View Shed button for any farm, the user is redirected to the Shed List page for that farm to manage its sheds. This streamlined flow makes it easy for users to add, edit, delete, and sort farms while keeping track of farm details and shed information.

Shed Register Module

The Shed List section allows users to manage shed information related to each farm. Users can view, add, edit, and delete sheds associated with specific farms. The Shed List page, shown in Figure 4.17, presents Shed ID, Farm Name, Capacity, and Status (active or inactive). Users can click on a farm's View Shed button to navigate to the specific shed management page. Adding a shed involves filling in details like Shed Name, Farm ID, and Capacity via a modal. Editing or deleting shed records follows the same procedure, with modals pre-filled with existing data and a confirmation alert for deletions.

All Farm > FD03 > All Shed

Shed Information

Shed Summary

Total Sheds	Total Capacity	Active Sheds
5	40000	1

Shed List

Add Shed
Sort by Newest
Show Active Only

Shed ID	Shed Name	Max Capacity	Shed Type	Feeding System	Status	Registered Date	Created By	Action
S011	G1	10000	Semi-Close House	Manual	Active	2025-05-02	Ikbar	Edit Delete
S012	G2	10000	Semi-Close House	Manual	Inactive Activate	2025-05-02	Tan Zi Siah	Edit Delete
S013	G3	10000	Semi-Close House	Manual	Inactive Activate	2025-05-02	Tan Zi Siah	Edit Delete
S018	G4	8000	Open House	Manual	Inactive Activate	2025-05-02	Tan Zi Siah	Edit Delete
S019	G5	2000	Open House	Manual	Inactive Activate	2025-05-02	Tan Zi Siah	Edit Delete

Figure 4.18 Shed List Page

Feed Type Module

The Feed Type section helps users manage feed types used across farms. Users can view all feed types, add new ones, and modify or delete existing records. As shown in Figure 4.18, the Feed Type Page includes columns like Feed Type ID, Name, and Description. Adding or editing uses modals, and deletions are confirmed with alerts to prevent accidental removal.

Feed Information

Feed Summary

Total Feeds	Unique Suppliers	Avg. Price
2	1	RM 145.75

Feed List

Filter by Feed Type:
Sort by Price:

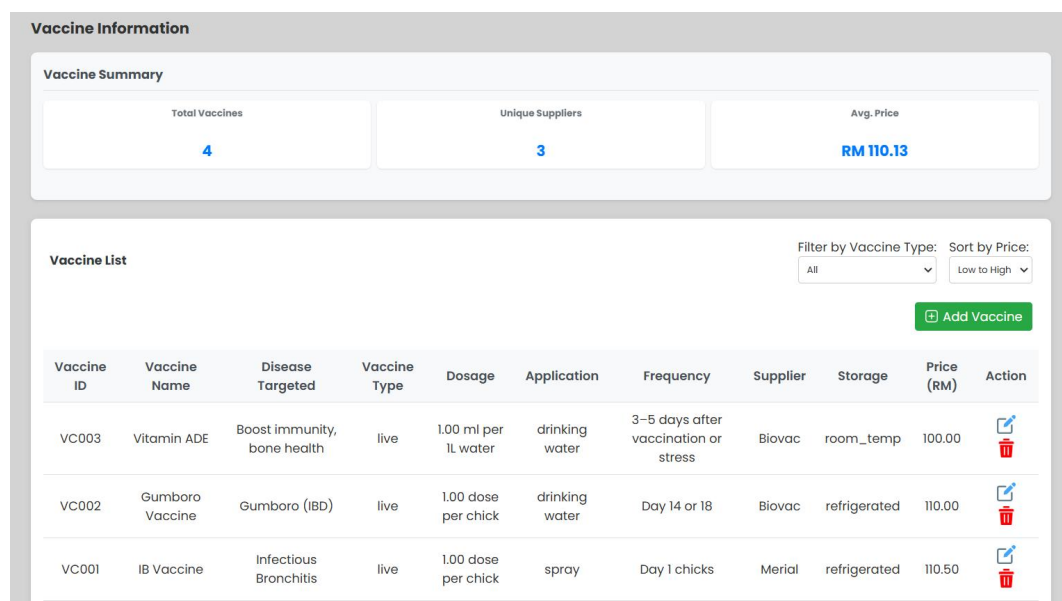
Add Feed

Feed ID	Feed Name	Supplier Name	Feed Type	Unit Weight	Price (RM)	Action
FD002	FFM 5220 Broiler Finisher Pellet II	FFM BERHAD	finisher	50 kg	144.20	Edit Delete
FD001	FFM 5210 Starter Crumble I	FFM BERHAD	starter	50 kg	147.30	Edit Delete

Figure 4.19 Feed Type Page

Vaccine Type Module

This section allows users to manage vaccines administered in the poultry farms. Similar to other modules, it supports viewing, adding, editing, and deleting vaccine records. Figure 4.19 illustrates the Vaccine Type Page, which lists Vaccine ID, Vaccine Name, Disease Targeted, and other relevant information. Modals are used for both adding and editing, where users input Vaccine Name, Disease Targeted, Dosage, and Storage Condition. A confirmation alert ensures safe deletion.



The screenshot shows the 'Vaccine Information' page. At the top, there's a 'Vaccine Summary' section with three cards: 'Total Vaccines' (4), 'Unique Suppliers' (3), and 'Avg. Price' (RM 110.13). Below this is a 'Vaccine List' section with a table of vaccine records. The table has columns for Vaccine ID, Vaccine Name, Disease Targeted, Vaccine Type, Dosage, Application, Frequency, Supplier, Storage, Price (RM), and Action. There are three rows of data. Above the table, there are filters for 'Filter by Vaccine Type' (set to 'All') and 'Sort by Price' (set to 'Low to High'). A green 'Add Vaccine' button is located above the table.

Vaccine ID	Vaccine Name	Disease Targeted	Vaccine Type	Dosage	Application	Frequency	Supplier	Storage	Price (RM)	Action
VC003	Vitamin ADE	Boost immunity, bone health	live	1.00 ml per 1L water	drinking water	3-5 days after vaccination or stress	Biovac	room_temp	100.00	 
VC002	Gumboro Vaccine	Gumboro (IBD)	live	1.00 dose per chick	drinking water	Day 14 or 18	Biovac	refrigerated	110.00	 
VC001	IB Vaccine	Infectious Bronchitis	live	1.00 dose per chick	spray	Day 1 chicks	Merial	refrigerated	110.50	 

Figure 4.20 Vaccine Type Page

Chick Type and Mortality Module

The Chick Mortality section helps users track and manage chick mortality data across farms. This module allows users to add new mortality records, edit existing ones, and delete unnecessary entries. The Chick Mortality table shown in Figure 4.20 provides insights into Farm ID, Date, Mortality Rate, and Reason for Mortality. Users can input data into a modal when adding or editing mortality records. Deleting a mortality record requires user confirmation, ensuring accurate data management and preventing accidental deletions.

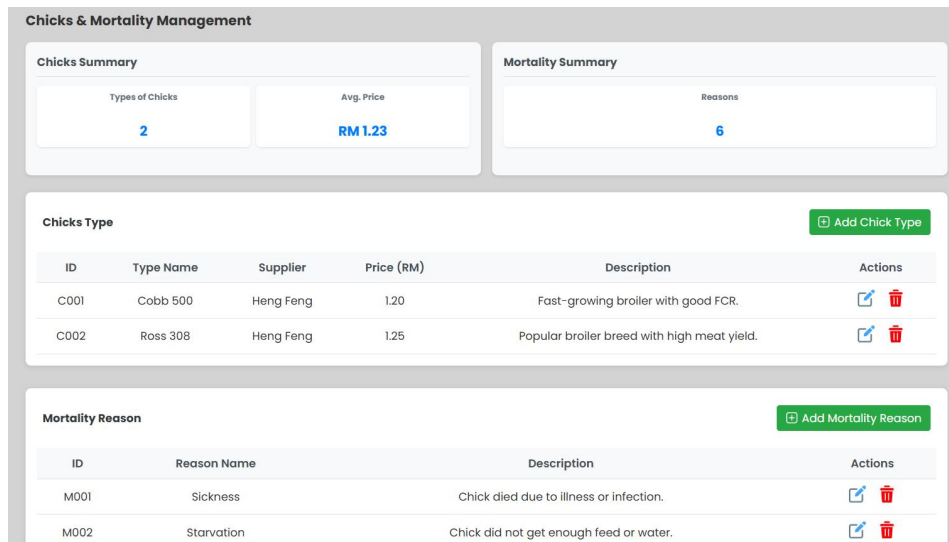


Figure 4.21 Chicks and Mortality Page

4.2.4.5 Alert Module

The Alert Threshold Module is used to manage and set threshold values for different chick types to monitor their weekly performance. It helps users track key indicators such as weight, feed intake, and Feed Conversion Ratio (FCR) across various weeks, ensuring that the chickens are growing according to expected standards. By setting these thresholds, users can easily identify underperformance or irregularities in the flock's development, enabling timely corrective actions to maintain optimal farm productivity.

Figure 4.21 shows the standard preset data for chick types C001 and C002, which are automatically saved into the system. These presets act as baseline references for chick development. Users can select either type and set it as the default threshold for monitoring purposes.

The Alert Thresholds Management interface, illustrated in Figure 4.22, enables users to view, filter, and manage the threshold data week by week. It includes tools for comparing actual performance against target values, allowing for quick detection of anomalies in chick growth trends.

This module is essential for performance tracking and provides users with an efficient way to ensure that flocks are progressing as expected over time.

```
const standardData = {
  c001: {
    weight: {
      1: 42, 2: 63, 3: 74, 4: 90, 5: 109, 6: 134, 7: 163, 8: 193, 9: 228,
      10: 269, 11: 313, 12: 362, 13: 414, 14: 469, 15: 528, 16: 589, 17: 654,
      18: 722, 19: 792, 20: 865, 21: 941, 22: 1018, 23: 1098, 24: 1180,
      25: 1264, 26: 1349, 27: 1436, 28: 1525, 29: 1615, 30: 1706, 31: 1798,
      32: 1892, 33: 1986, 34: 2081, 35: 2177, 36: 2273, 37: 2369, 38: 2466,
      39: 2563, 40: 2661, 41: 2758, 42: 2855, 43: 2952, 44: 3049, 45: 3145,
      46: 3240, 47: 3335, 48: 3430, 49: 3524, 50: 3617
    },
    feed: {
      1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0, 5: 0.0, 6: 0.0, 7: 0.0, 8: 0.2, 9: 3.6, 10: 4.5,
      11: 5.5, 12: 6.6, 13: 7.9, 14: 9.3, 15: 10.8, 16: 12.4,
      17: 14.1, 18: 15.9, 19: 18.0, 20: 20.1, 21: 22.4, 22: 24.8, 23: 27.2,
      24: 29.8, 25: 32.4, 26: 35.2, 27: 38.1, 28: 41.1, 29: 44.2, 30: 47.4,
      31: 50.7, 32: 54.0, 33: 57.4, 34: 60.9, 35: 64.4, 36: 68.0, 37: 71.6,
      38: 75.3, 39: 79.2, 40: 85.0, 41: 87.0, 42: 91.0, 43: 95.2, 44: 99.5,
      45: 103.8, 46: 105.3, 47: 112.8, 48: 117.5, 49: 122.2, 50: 127.0
    },
    fcr: {
      1: 0.00, 2: 0.00, 3: 0.00, 4: 0.00, 5: 0.00, 6: 0.00, 7: 0.00, 8: 0.76, 9: 0.80,
      10: 0.84, 11: 0.88, 12: 0.92, 13: 0.95, 14: 1.00, 15: 1.03, 16: 1.05,
      17: 1.08, 18: 1.10, 19: 1.13, 20: 1.16, 21: 1.19, 22: 1.23, 23: 1.24,
      24: 1.26, 25: 1.28, 26: 1.30, 27: 1.33, 28: 1.35, 29: 1.37, 30: 1.39,
      31: 1.41, 32: 1.43, 33: 1.45, 34: 1.46, 35: 1.48, 36: 1.50, 37: 1.51,
      38: 1.53, 39: 1.54, 40: 1.56, 41: 1.58, 42: 1.59, 43: 1.61, 44: 1.63,
      45: 1.65, 46: 1.67, 47: 1.69, 48: 1.71, 49: 1.73, 50: 1.76
    }
  }
}
```

Figure 4.22 Standard Data saved in the system

The screenshot displays the 'Alert Threshold Management' interface. At the top, there's a section for 'Chicken Type: C001 - Cobb 500'. Below this is a table with columns: 'Day', 'Standard Weight (kg)', and 'FCR'. The table is divided into three sections: 'Week 1 (Days 1-7)', 'Week 2 (Days 8-14)', and 'Week 3 (Days 15-21)'. The 'Day' column lists days 8 through 14. The 'Standard Weight (kg)' column shows values like 0.15, 0.18, 0.23, etc. The 'FCR' column shows values like 0.79, 0.84, 0.88, etc. Annotations include: a red box around the 'Filter by Week' dropdown menu; a green box around the 'Reset to Defaults' button; a green box around the 'Edit' buttons in the 'Actions' column; and an orange box around the 'Edit and save data' text with an arrow pointing to the 'Edit' buttons.

Figure 4.23 Alert Thresholds Management

4.2.4.6 Daily Record Module

Flock Module

The Flock module displays all active sheds in the system, allowing users to view and manage flocks. It fetches shed and farm data, filters for active sheds, and retrieves the creator's username based on passport numbers. Users can filter sheds by farm using a dropdown, and each shed card shows key details like shed name, farm, capacity, type, feeding system, register date, and creator info. A “View Flocks” button allows navigation to flock records for each shed, while an “Activate New Shed” button redirects to the shed activation page. This module acts as the main entry point for accessing and managing flocks. Figure 4.23 shows the interface for Active Shed management.

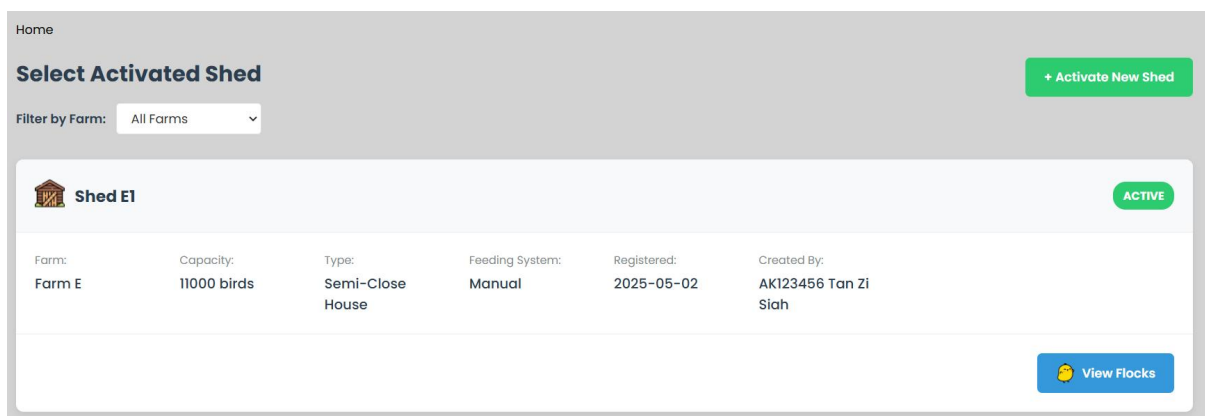


Figure 4.24 Active Shed

Flock Daily Record Module

The Flock Daily Record module allows users to manage and monitor daily data for an active chicken flock in a selected shed. If no active flock exists, users are prompted to add chick arrival details through a modal form. Once a flock is active, the page displays key information such as flock ID, shed name, arrival date, and current age. Users can view and manage four types of records—Feed, Vaccine, Mortality, and Weight—organized in tabs.

Each tab includes a table of existing records and an "Add Record" button that opens a modal form. These modals provide structured input fields specific to the record type (e.g., date, quantity, supplier, cause of death) and allow users to add or edit records efficiently. This module ensures accurate, real-time documentation of flock health and performance throughout its cycle. Figure 4.24 presents the Daily Flock Record interface, while Figure 4.25 illustrates the modal form used for inputting data.

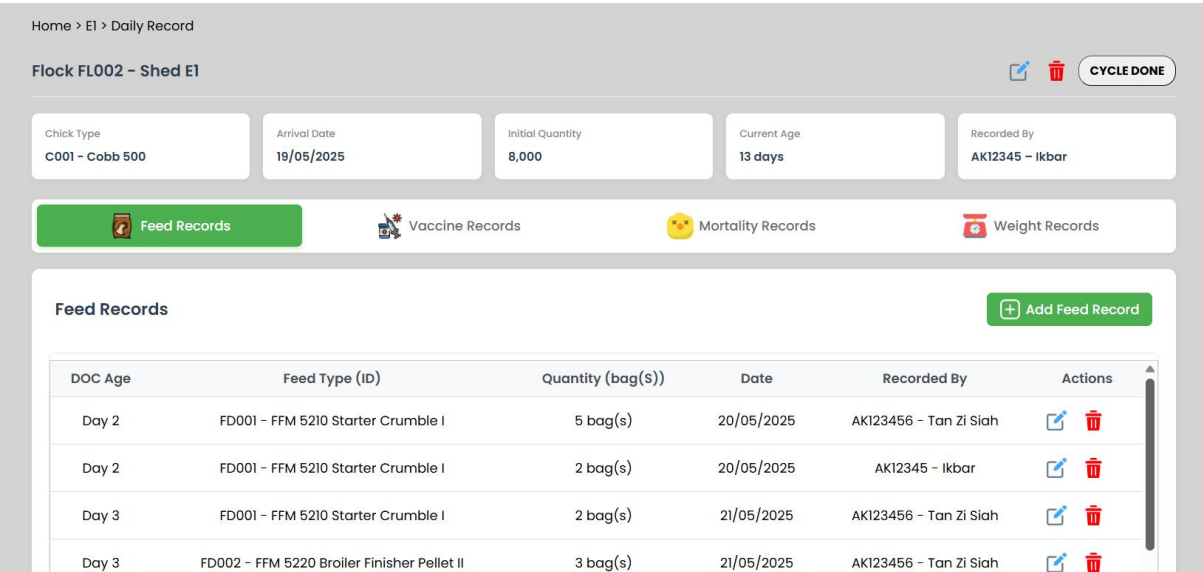


Figure 4.25 Daily Flock Record

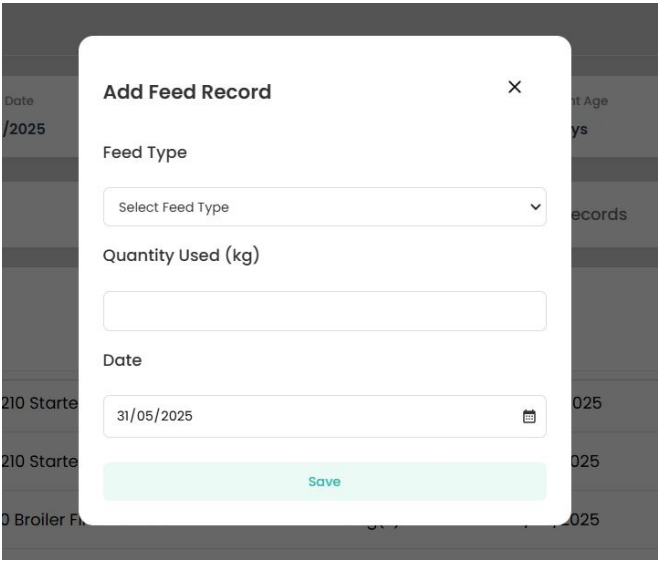


Figure 4.26 Modal Form to Input Data

Performance Module

The Performance Module provides a real-time dashboard for evaluating an active flock's progress against its breed's predefined alert thresholds. After retrieving data related to feed consumption, weight records, mortality rates, and threshold benchmarks, the module displays three key summary cards: Latest Average Weight, Cumulative Feed Used, and Current Survival Rate. Below the summary, two tabs present detailed analytics:

- The FCR Comparison tab aggregates data at weekly intervals (e.g., Day 1, 7, 14... 49) and compares actual Feed Conversion Ratios (FCR) to expected values. Color-coding and icons highlight whether the flock is meeting performance targets.
- The Mortality Tracking tab lists day-by-day death counts, automatically calculates the cumulative survival percentage, and flags any day where survival drops below 90% as a warning.

This compact interface enables users to quickly identify feed efficiency issues or health concerns and take action before flock performance deteriorates. Figure 4.26 illustrates the FCR Performance Dashboard interface.

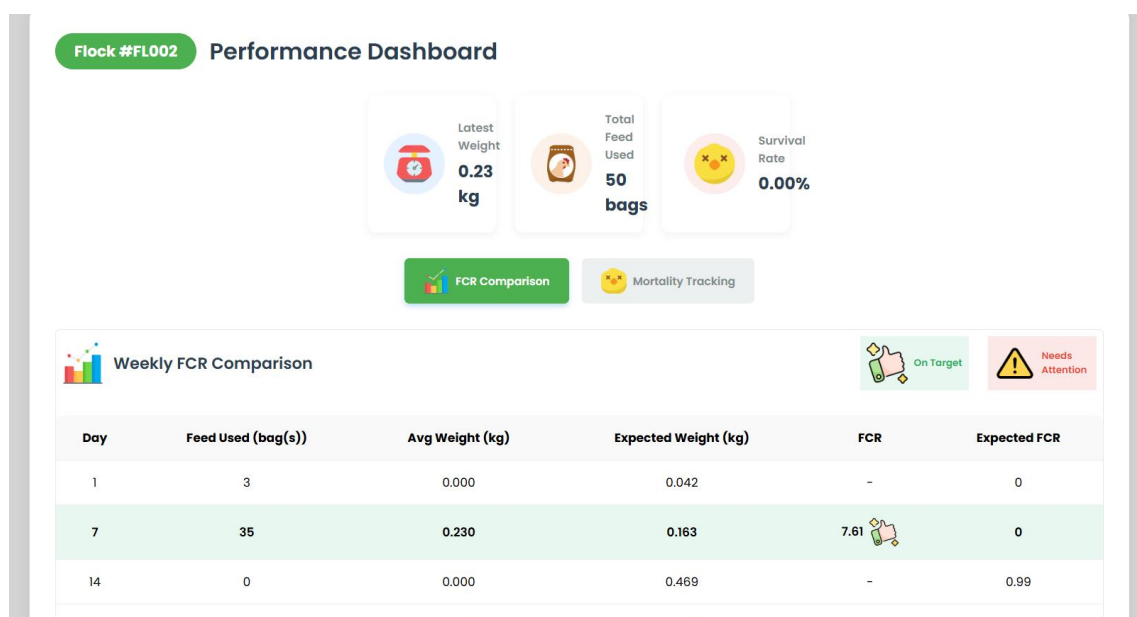


Figure 4.27 FCR Performance Dashboard

4.2.4.7 Feed Inventory Module

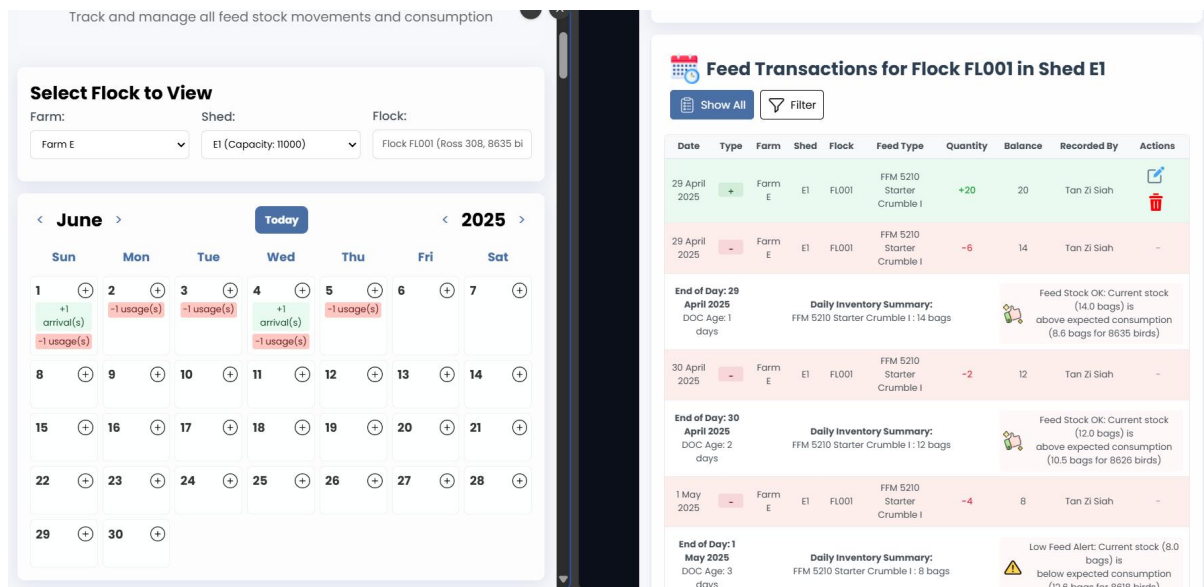


Figure 4.28 Feed inventory interface with calendar and transaction log



Figure 4.29 Feed Balance Summary

The Feed Inventory Module allows users to track feed movements and usage across farms. Users can select a farm, view active sheds and flocks, and manage feed transactions using an interactive calendar. Feed arrivals and usage are color-coded for clarity. Modal forms enable users to add, edit, or delete records, with automatic balance calculations based on usage and mortality data.

The module includes daily summaries, 3-day stock alerts, and a balance table showing total feed arrived, used, and remaining. Filtering by date, month, or year is supported for easier reporting. Visual indicators and responsive design help ensure efficient inventory control. Figure 4.27 shows the feed inventory interface with calendar and summary views.

4.2.4.8 Vaccine Inventory Module

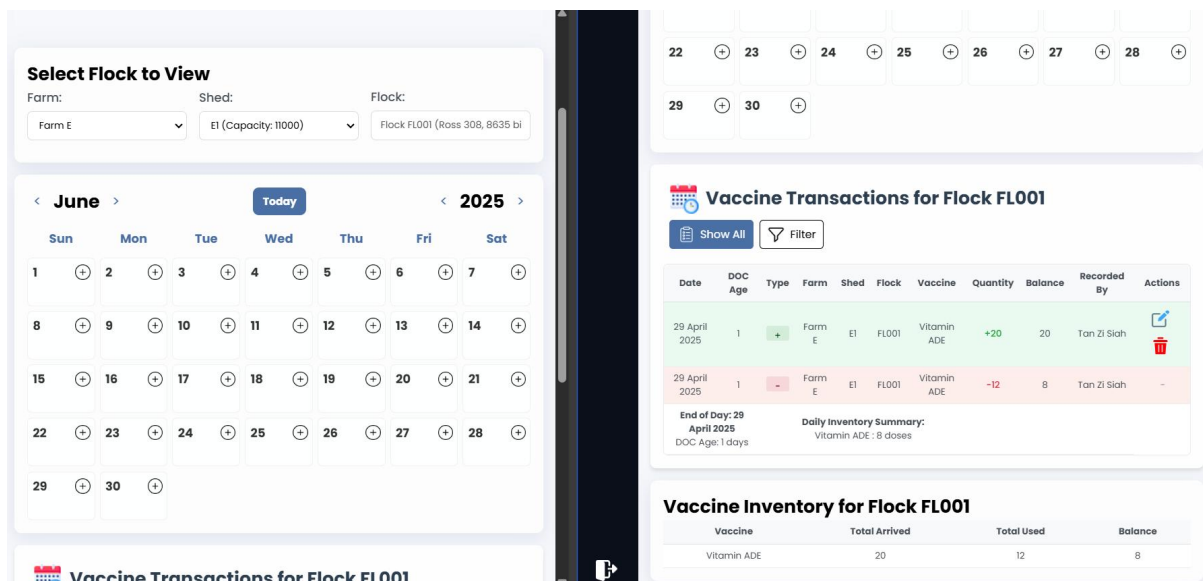


Figure 4.30 Vaccine inventory interface with calendar and transaction log



Figure 4.31 Vaccine Balance Summary

The Vaccine Inventory Module enables users to monitor vaccine stock arrivals and usage across flocks. Users can select a farm, shed, and flock to view transactions. An interactive calendar highlights daily vaccine arrivals (green) and usage (red), while modal forms support adding or editing records.

The system auto-calculates remaining doses and provides daily summaries showing DOC age and current balances. A balance table tracks total arrived, used, and remaining per vaccine type, with color-coded alerts for low or negative stock. Figures 4.29 and 4.30 show the vaccine inventory interface and balance summary.

4.2.4.9 Report Module

The Report Module offers a centralized platform for analyzing flock performance, integrating data from feed, vaccine, mortality, and weight records. Users can filter by farm and shed, and view multi-tab reports including daily activities, transaction logs, and FCR comparisons against breed thresholds. Each tab presents structured tables and meaningful insights, helping users track and understand flock conditions day by day. Reports can be exported as PDFs with charts and tables for easier documentation and decision-making. Integrated visualizations help track performance trends across the flock cycle, allowing users to quickly identify areas needing attention. Figures 4.31 and 4.32 display the report interface and export preview.

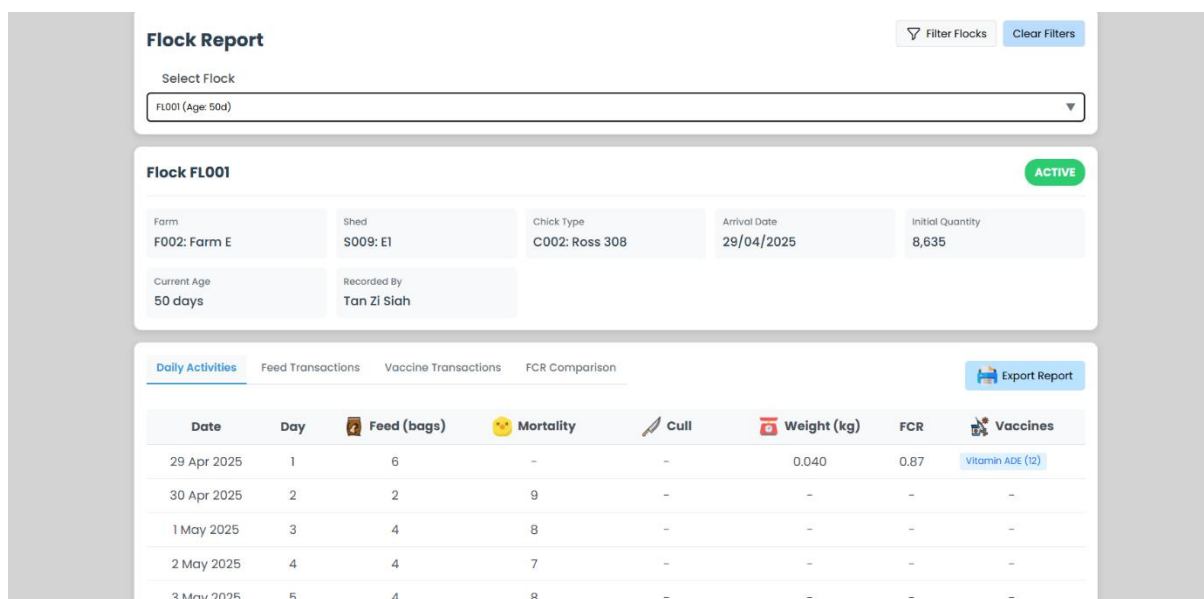


Figure 4.32 Report Interface with Filter Controls and Tab Navigation

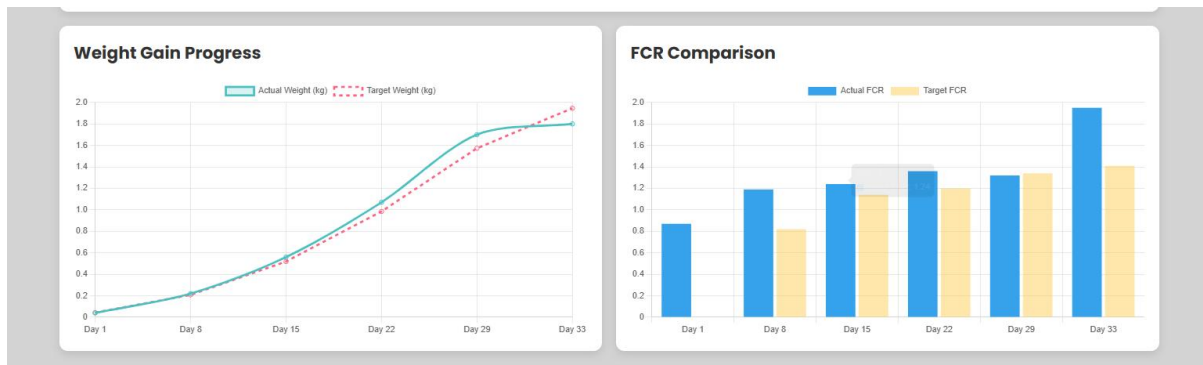


Figure 4.33 Flock Performance Charts

4.3 Database Implementation

The backend of this system is developed using Django, which manages the database using its built-in ORM (Object-Relational Mapping). The database is used to store data such as farms, flocks, feeds, vaccines, and users, and this data can be viewed, added, or updated through both the admin panel and the React frontend.

4.3.1. Django Models and Migrations

Django models are Python classes that define the structure of the database tables. Each model corresponds to one table in the database. Django automatically creates and updates tables through a migration system.

- During development, the following commands were used to generate and apply database structure:

```
python manage.py makemigrations
python manage.py migrate
```

- Foreign keys were used to create relationships between tables. For example:

```
class Flock(models.Model):
    flock_id = models.CharField(max_length=10, primary_key=True, editable=False)
    shed_id = models.ForeignKey(Shed, to_field='shed_id', on_delete=models.CASCADE)
    chick_id = models.ForeignKey(ChickType, to_field='chick_id', on_delete=models.CASCADE)
    arrival_date = models.DateField()
    quantity = models.PositiveIntegerField()
    status = models.CharField(max_length=10, choices=[('active', 'Active'), ('done', 'Done')], default='active')
    complete_date = models.DateField(null=True, blank=True, help_text="Date when flock is completed")
    recorded_by = models.ForeignKey('users.FarmWorker', to_field='passport_number', on_delete=models.CASCADE)
```

Figure 4.34 Example of a Django Model Definition

4.3.2. Django Admin Panel

Django provides a built-in Admin panel that allows developers and staff to manage database records using a simple web interface.

- It supports creating, updating, and deleting records for all registered models.
- The admin site is accessible at <http://localhost:8000/admin/>.
- Models must be registered in admin.py to appear in the panel:

```
from django.contrib import admin
from .models import Flock, VaccineRecord, FeedRecord, MortalityRecord, WeightRecord, FcrRecord

@admin.register(Flock)
class FlockAdmin(admin.ModelAdmin):
    list_display = ('flock_id', 'shed_id', 'chick_id', 'arrival_date', 'complete_date', 'quantity', 'status', 'doc_age', 'current_fcr')
    list_filter = ('status', 'shed_id', 'arrival_date')
    search_fields = ('flock_id', 'shed_id_shed_name', 'chick_id_chick_name')
    readonly_fields = ('flock_id', 'doc_age', 'current_fcr')
```

Figure 4.35 Example of a Django Admin Registration

Figure 4.36 below displays how the Flocks are shown in the Django Admin panel.

Select flock to change

Q Search

Action: Go 0 of 1 selected

☐	FLOCK ID	SHED ID	CHICK ID	ARRIVAL DATE	COMPLETE DATE	QUANTITY	STATUS	DOC AGE	CURRENT FCR
☐	FL001	E1 (S009) - Farm E	Ross 308 (Heng Feng)	April 29, 2025	-	8635	Active	51	5.91

1 flock

Figure 4.36 Django Admin Interface Showing Flock Table

4.3.3. Database Setup for Production

During development, the system used SQLite due to its simplicity. For deployment in a production environment, the database was switched to PostgreSQL for improved scalability and performance. Figure 4.37 shows the code used to connect to a PostgreSQL database in production mode.

```
# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Figure 4.37 Database Configuration Architecture

4.4 Summary

This chapter presents the implementation of PoultryPlus, a comprehensive poultry farm management system developed using the Django REST Framework for the backend and React.js for the frontend. The system was built based on the design specifications outlined in Chapter 3, integrating key features such as role-based access control (for managers and workers), real-time data monitoring, and automated alerts.

All core modules were implemented, including authentication, flock management, inventory tracking (feeds and vaccines), performance monitoring, and report generation. Django's ORM was used to manage database interactions, with SQLite used during development and PostgreSQL configured for production.

The React frontend communicates with the backend via RESTful APIs, using JWT authentication to ensure secure data access. The chapter demonstrates successful integration

of backend and frontend components, supported by screenshots and code examples. Overall, the implementation meets the defined requirements and is ready for deployment.

Chapter 5 : SYSTEM TESTING

5.1 Introduction

This chapter outlines the testing phase conducted on the PoultryPlus system to ensure its reliability, correctness, and readiness for deployment. Black box testing was used as the main approach, focusing on verifying that each module behaves according to the specified functional requirements, without knowledge of the internal code structure. Testing was performed module by module after each implementation phase, with results used to fix bugs and improve functionality. Test cases were designed based on the expected behavior defined in the system requirements, and the system was tested from the end-user's perspective. The goal was to confirm that the system fulfills its intended purpose, supports user roles (manager and worker), and maintains data integrity across all modules. Feedback from users (farm managers and workers) was also collected to validate usability, which helped refine the interface and ensure the system meets real-world needs.

5.2 Functionality Testing

Functionality testing was conducted on all core modules of the PoultryPlus system to verify that each feature works according to its intended purpose. The modules tested included authentication, flock management, feed inventory, vaccine management, and user role control. For each module, test cases were designed based on the system's functional requirements, focusing on input validation, output accuracy, data consistency, and proper access control for both managers and workers. The authentication module was tested to ensure secure login using JSON Web Tokens (JWT) and correct role-based redirection. The flock module was checked for accurate creation, editing, and deletion of flock records. Feed and vaccine modules were tested for proper inventory tracking, validation of quantities, and system responses to threshold alerts. The user management module confirmed that only managers

could approve or view worker profiles. All test results were compared against expected outcomes, and any issues discovered were documented, resolved, and retested to ensure system reliability.

Table 23 Test Case for Login Module

Test Case ID: TC1			Test Title: Verification on login function		Test Designed by: Tan Zi Siah	
Test Designed date: 18 th June 2025			Module Name: Login Module of the PoultryPlus System		Test Priority (Low/Medium/High): High	
Description: Test the login function						
Pre-conditions: Valid user accounts exist in database						
Dependencies: Authentication API is operational						
Test ID	Test Case	Test Data	Expected Result	Actual Result	Status (pass/fail)	Notes (Types of User)
1	Valid Manager login	Passport: AK123456 Password: 12345	Redirect to manager dashboard	As expected	Pass	Manager role
2	Valid Worker login	Passport: AK12345 Password: 12345	Redirect to manager dashboard	As expected	Pass	Worker role
3	Invalid credentials	Passport: MGR001 Password: Wrong123	"Invalid credentials" error	Error displayed	Pass	Manager/Worker
4	Empty submission	Passport: (blank) Password: (blank)	Field validation errors	"Required field" alerts	Pass	Manager/Worker
5	Pending account	Passport: AK1234 Password: 12345	Account pending approval	Message displayed	Pass	New Worker
Post-conditions: Successful login redirects user to the correct dashboard based on role.						

Table 24 Test Case for Farm Module

Test Case ID: TC2		Test Title: Farm CRUD Operations		Test Designed by: Tan Zi Siah		
Test Designed date: 18 th June 2025		Module Name: Farm Management Module		Test Priority (Low/Medium/High): High		
Description: Test farm creation, editing, and deletion						
Pre-conditions: User is logged in successfully						
Dependencies: Farm API endpoints are active						
Test ID	Test Case	Test Data	Expected Result	Actual Result	Status (pass/fail)	Notes (Types of User)
1	Add valid farm	Name: Farm G Address: Sungai Rait Simpang 7	Success notification	Farm appears in list	Pass	Manager/Worker
2	Add duplicate farm	Name: Farm G	"Farm already exists" error	Error displayed	Pass	Manager/Worker
3	Edit farm details	New Address: 456 Agri Park	Changes saved in DB	UI updates	Pass	Manager/Worker
4	Delete farm	Confirm deletion	Farm removed from list	DB updated	Pass	Manager/Worker
Post-conditions: Successful add new farm to the system.						

Table 25 Test Case for Shed Module

Test Case ID: TC3		Test Title: Shed Operations		Test Designed by: Tan Zi Siah		
Test Designed date: 18 th June 2025		Module Name: Shed Management Module		Test Priority (Low/Medium/High): High		
Description: Test shed creation and capacity validation						
Pre-conditions: At least one farm exists						
Dependencies: -						
Test ID	Test Case	Test Data	Expected Result	Actual Result	Status (pass/fail)	Notes (Types of User)
1	Add valid shed	Shed Name: G1 Max Capacity: 10000 Shed Type: Semi-Close House Feeding System Type: Manual	Shed successfully added and listed in the table	As expected	Pass	Manager/Worker
2	Exceed or under capacity limit	Shed Name: G1 Max Capacity: 200000 Shed Type: Semi-Close House Feeding System Type: Manual	Error message shown: "Capacity must be between 2000 and 100000"	As expected	Pass	Manager/Worker

3	Activate shed	Shed Name: G1 Status: Active	Shed marked as active and visually indicated in table	As expected	Pass	Manager/Worker
4	Edit shed details	Change capacity of G1 from 2000 to 8000	Capacity updated successfully and reflected in the table	As expected	Pass	Manager/Worker
5	Delete shed	Confirm deletion	Confirmation prompt shown, shed deleted after confirmation	As expected	Pass	Manager/Worker
Post-conditions: Shed operations performed correctly; data updates reflect in frontend and database.						

Table 26 Test Case for Feed Module

Test Case ID: TC4		Test Title: Feed Management		Test Designed by: Tan Zi Siah		
Test Designed date: 18 th June 2025		Module Name: Feed Module		Test Priority (Low/Medium/High): High		
Description: Test the functionality of adding, editing, and deleting feed types.						
Pre-conditions: -						
Dependencies: -						
Test ID	Test Case	Test Data	Expected Result	Actual Result	Status (pass/fail)	Notes (Types of User)
1	Add New Feed	Feed Name: FFM 5220 Broiler Finisher Pellet II Supplier Name: FFM BERHAD Feed Type: finisher Unit Weight:50kg Price: 144.20	Feed added successfully and displayed in list	As expected	Pass	Manager/Worker
2	Missing mandatory fields	Feed Name: (blank) Unit Weight: 50kg	Validation error shown ("Feed Name is required")	As expected	Pass	Manager/Worker
3	Edit Existing Feed	Change Price from	Feed updated and	As expected	Pass	Manager/Worker

	Data	RM144.20 to RM145.00	new price shown			
4	Delete Existing Feed	Select feed and click delete	Feed deleted after confirmation prompt	As expected	Pass	Manager/Worker
Post-conditions: Feed information is consistently stored, validated, and displayed in the inventory module.						

Table 27 Test Case for Vaccine Module

Test Case ID: TC5		Test Title: Vaccine Record Management		Test Designed by: Tan Zi Siah		
Test Designed date: 18 th June 2025		Module Name: Vaccine Module		Test Priority (Low/Medium/High): High		
Description: Test CRUD functionality for vaccine records.						
Pre-conditions: -						
Dependencies: -						
Test ID	Test Case	Test Data	Expected Result	Actual Result	Status (pass/fail)	Notes (Types of User)
1	Add New Vaccine	Vaccine Name: NDV-LS Disease Targeted: Newcastle Type: Live Dosage: 0.2 ml	Vaccine saved and listed	As expected	Pass	Manager/Worker

		Unit: ml Method: Oral Frequency: Once Supplier: ABC Vet Storage: Cool place Price: RM1.20				
2	Missing mandatory fields	Vaccine Name: (blank), Disease: Newcastle	Validation error shown	As expected	Pass	Manager/Worker
3	Edit Existing Vaccine Data	Change Application Method from Oral to Injection	Vaccine updated successfully	As expected	Pass	Manager/Worker
4	Delete Existing Vaccine	Select vaccine and click delete	Vaccine removed after confirmation	As expected	Pass	Manager/Worker
Post-conditions: Vaccine data is saved securely, and available for vaccination scheduling in daily activity logs.						

Table 28 Test Case for Chicks Module

Test Case ID: TC6	Test Title: Chick Type Management	Test Designed by: Tan Zi Siah
Test Designed date: 18 th June 2025	Module Name: Chick Type Module	Test Priority (Low/Medium/High): High
Description: Test the management of different types of chicks used in flock		

operations.						
Pre-conditions: -						
Dependencies: -						
Test ID	Test Case	Test Data	Expected Result	Actual Result	Status (pass/fail)	Notes (Types of User)
1	Add New Chick type	Type: Cobb 500 Supplier: Charoen Price: RM3.20 Description: High growth breed	Chick type added and displayed in list	As expected	Pass	Manager/Worker
2	Missing mandatory fields	Type Name: (blank), Price: RM3.20	Validation error: "Type Name required"	As expected	Pass	Manager/Worker
3	Edit Existing Chicks Data	Update price from RM3.20 to RM3.50	Data updated successfully	As expected	Pass	Manager/Worker
4	Delete Existing Chicks	Select chick and click delete	Chick type deleted after confirmation	As expected	Pass	Manager/Worker
Post-conditions: Chick type is accurately maintained and used during flock registration and monitoring.						

Table 29 Test Case for Mortality Module

Test Case ID: TC7			Test Title: Mortality Reason Management		Test Designed by: Tan Zi Siah		
Test Designed date: 18 th June 2025			Module Name: Mortality Module		Test Priority (Low/Medium/High): High		
Description: Test managing mortality reasons for data entry in activity logs.							
Pre-conditions: -							
Dependencies: -							
Test ID	Test Case		Test Data	Expected Result	Actual Result	Status (pass/fail)	Notes (Types of User)
1	Add New Mortality Reason	Reason: Dehydration Description: High temp.	Reason added and selectable in activity logs	As expected	Pass	Manager/Worker	
2	Missing mandatory fields		Reason: (blank), Description: High temp.	Validation error shown	As expected	Pass	Manager/Worker
3	Edit Existing Mortality Data	Update reason from "Dehydration" to "Overheating"	Reason updated successfully	As expected	Pass	Manager/Worker	
4	Delete Existing Mortality Reason	Select reason and click delete	Reason deleted after confirmation	As expected	Pass	Manager/Worker	
Post-conditions: Mortality reasons are maintained properly for daily activity tracking and reporting accuracy.							

Table 30 Test Case for Alert Module

Test Case ID: TC8		Test Title: Alert Threshold Management		Test Designed by: Tan Zi Siah		
Test Designed date: 18 th June 2025		Module Name: Alert Module		Test Priority (Low/Medium/High): High		
Description: Test viewing and editing alert thresholds for standard flock performance monitoring.						
Pre-conditions: Chick type thresholds must be initialized.						
Dependencies: -						
Test ID	Test Case	Test Data	Expected Result	Actual Result	Status (pass/fail)	Notes (Types of User)
1	View the Alert Threshold based on chick type	Chicken Type: C002	Standard threshold table displays correct data	As expected	Pass	Manager/Worker
2	Edit alert threshold	Chicken Type: C002 Day: 2 Standard Weight (kg): change from 0.061 to 0.065 Total Feed per chicken (kg): 0.01	Threshold updated.	Updated correctly	Pass	Manager/Worker
Post-conditions: Alert thresholds are updated in the system and applied during comparison in daily performance monitoring.						

Table 31 Test Case for Flock Module

Test Case ID: TC9		Test Title: Flock Arrival Records		Test Designed by: Tan Zi Siah		
Test Designed date: 18 th June 2025		Module Name: Flock Management Module		Test Priority (Low/Medium/High): High		
Description: Test the functionality of recording flock arrivals						
Pre-conditions: Farm and shed must be registered and activated; at least one chick type must be available in setup.						
Dependencies: Chick Type, Farm, and Active Shed records must exist.						
Test ID	Test Case	Test Data	Expected Result	Actual Result	Status (pass/fail)	Notes (Types of User)
1	Add valid flock record	Chick Type: C002 Arrival Date: 16/6/2025 Quantity: 8000	Flock added successfully	As expected	Pass	Manager/Worker
2	Missing mandatory fields	Chick Type: (blank) Arrival Date: 16/6/2025 Quantity: 8000	Validation error: "Please fill all required fields"	As expected	Pass	Manager/Worker
3	Quantity exceeds shed limit	Quantity: 200000	Error: "Value must be less than	As expected	Pass	Manager/Worker

			or equal to 100000"			
4	Edit flock record	Change quantity from 8000 to 4000 for existing flock	Flock updated successfully	As expected	Pass	Manager/Worker
5	Delete Flock	Select flock record and click delete	Confirmation prompt displayed; record removed upon confirmation	As expected	Pass	Manager/Worker
6	Invalid arrival date	Arrival Date:26/6/2025(future date)	Error: "Arrival date cannot be in the future"	As expected	Pass	Manager/Worker
7	Cycle Done	Set status to "Cycle Done" for completed flock	Flock status updated	As expected	Pass	Manager/Worker
Post-conditions: Flock arrival records are validated, saved in database, and reflected in the flock summary interface.						

Table 32 Test Case for Daily Flock Activity Module

Test Case ID: TC10		Test Title: Daily Flock Activities		Test Designed by: Tan Zi Siah		
Test Designed date: 18 th June 2025		Module Name: Daily Flock Activities Management Module		Test Priority (Low/Medium/High): High		
Description: Test the functionality of recording flock daily activities						
Pre-conditions: At least one active flock record must exist with valid feed types, vaccine types, and mortality reasons configured.						
Dependencies: Requires Flock, Feed Type, Vaccine Type, and Mortality Reason to be present in the system.						
Test ID	Test Case	Test Data	Expected Result	Actual Result	Status (pass/fail)	Notes (Types of User)
1	Add daily feed record	Feed Type: F001 Quantity Used: 7 bags Date: 16/6/2025	Record saves successfully and reflected in feed summary	As expected	Pass	Manager/Worker
2	Add daily vaccine record	Vaccine: VC003 Quantity Applied: 10 Date: 16/6/2025	Record saves successfully and reflected in vaccine chart	As expected	Pass	Manager/Worker
3	Add mortality record	Reason: M001 No. of Deaths: 10 Date: 16/6/2025	Record saves successfully and mortality rate updated	As expected	Pass	Manager/Worker

4	Add weight record	Number of Chicks Weighted: 10 Total Weight Measured (kg): 3.8 Date: 16/6/2025	Record saves successfully and reflected in performance chart	As expected	Pass	Manager/Worker
5	Missing mandatory fields	For feed: Feed Type: (blank) Quantity Used: 7 bags Date: 16/6/2025	Error: "Feed Type is required"	As expected	Pass	Manager/Worker
6	Edit existing activity entry	Change feed quantity used from 7 to 8	Record updates successfully and reflects new value	As expected	Pass	Manager/Worker
7	Delete existing activity entry	Select record and click delete	Record is removed from database with confirmation prompt	As expected	Pass	Manager/Worker
Post-conditions: All daily flock activity records (feed, vaccine, mortality, and weight) are stored in the database, validated, and accurately displayed in performance and monitoring interfaces.						

Table 33 Test Case for User Module

Test Case ID: TC11			Test Title: User Account and Profile Management		Test Designed by: Tan Zi Siah	
Test Designed date: 18 th June 2025			Module Name: User Module		Test Priority (Low/Medium/High): High	
Description: Test the ability to manage user profiles, approval workflows, password updates, and user removal.						
Pre-conditions: User accounts must exist in the system.						
Dependencies: Authentication and user role management components.						
Test ID	Test Case	Test Data	Expected Result	Actual Result	Status (pass/fail)	Notes (Types of User)
1	Edit Profile	Change nationality from Malaysia to Indonesia	Profile updated successfully	As expected	Pass	Manager/Worker
2	Approve or Reject User	Approve new registered worker	User status updated and reflected in user list	As expected	Pass	Manager
3	Change Worker Password	New Password: 54321	Password updated successfully	As expected	Pass	Manager
4	Delete Worker	Select worker account	User removed	As expected	Pass	Manager

		and click delete	from system after confirmation			
Post-conditions: User data is updated or removed appropriately, and access rights reflect changes immediately.						

Table 34 Test Case for Report Module

Test Case ID: TC12		Test Title: Flock Performance Report Generation		Test Designed by: Tan Zi Siah		
Test Designed date: 18 th June 2025		Module Name: Report Module		Test Priority (Low/Medium/High): High		
Description: Test the ability to view and export performance reports per flock.						
Pre-conditions: Flock activity records must exist.						
Dependencies: -						
Test ID	Test Case	Test Data	Expected Result	Actual Result	Status (pass/fail)	Notes (Types of User)
1	View report	Select Flock: FL001	Report generated with correct summary	As expected	Pass	Manager/Worker
2	Export Report	PDF / Print Option	Report is downloadable and printable	As expected	Pass	Manager/Worker

Post-conditions: Reports are generated accurately and can be exported for review and documentation.

Table 35 Test Case for Feed Inventory Module

Test Case ID: TC13		Test Title: Feed Inventory Management		Test Designed by: Tan Zi Siah		
Test Designed date: 18 th June 2025		Module Name: Feed Inventory Module		Test Priority (Low/Medium/High): High		
Description: Test the tracking of feed usage and arrivals in inventory.						
Pre-conditions: Feed types must be initialized in the system.						
Dependencies: Feed data setup and feed usage logs.						
Test ID	Test Case	Test Data	Expected Result	Actual Result	Status (pass/fail)	Notes (Types of User)
1	View Feed Used and Arrive Transactions	Farm: Farm E Sshed: E1 Flock: FL001	All feed usage and arrival records are displayed	As expected	Pass	Manager/Worker
2	Add Feed Arrival	Feed Type: F001 Quantity: 30	New arrival recorded and feed quantity updated	As expected	Pass	Manager/Worker
3	Edit Feed Arrival	Change Quantity from 30 to 32	Arrival record updated	As expected	Pass	Manager/Worker

4	Missing mandatory fields	Feed Type: (blank) Quantity: 30	Error: "Feed Type is required"	As expected	Pass	Manager/Worker
5	Delete existing feed arrival	Select record and delete	Arrival record removed after confirmation	As expected	Pass	Manager/Worker
Post-conditions: Feed inventory data is consistently updated and reflects accurate stock levels.						

Table 36 Test Case for Vaccine Inventory Module

Test Case ID: TC14		Test Title: Vaccine Inventory Management		Test Designed by: Tan Zi Siah		
Test Designed date: 18 th June 2025		Module Name: Vaccine Inventory Module		Test Priority (Low/Medium/High): High		
Description: Test the tracking and updating of vaccine usage and arrival in inventory.						
Pre-conditions: Vaccine types must be initialized.						
Dependencies: Vaccine data setup and usage logs.						
Test ID	Test Case	Test Data	Expected Result	Actual Result	Status (pass/fail)	Notes (Types of User)
1	View Vaccine Used and Arrive Transactions	Farm: Farm E Sshed: E1 Flock: FL001	All vaccine usage and arrival records displayed	As expected	Pass	Manager/Worker
2	Add Vaccine Arrival	Vaccine: F001	Vaccine arrival	As expected	Pass	Manager/Worker

		Quantity: 30	recorded and stock updated			
3	Edit Vaccine Arrival	Change Quantity from 30 to 35	Record updated and total reflects changes	As expected	Pass	Manager/Worker
4	Missing mandatory fields	Vaccine: (blank) Quantity: 30	Validation error: "Vaccine is required"	As expected	Pass	Manager/Worker
5	Delete existing vaccine arrival	Select record and delete	Arrival record removed after confirmation	As expected	Pass	Manager/Worker
Post-conditions: Vaccine inventory reflects accurate arrival and usage data for operational transparency.						

5.3 Usability Testing

The usability testing for PoultryPlus was conducted to assess the satisfaction, learnability, and efficiency of the system's user interface from the perspective of real users. A total of eight (8) respondents from UniGold Sdn Bhd Company, consisting of both farm managers and workers, participated in the evaluation.

Each participant interacted with core modules of the system and provided feedback using a structured usability questionnaire. The aim was to gather insights on system performance, ease of use, and design effectiveness.

The questionnaire was divided into three main parts:

- System Efficiency
- Graphical User Interface (GUI) Evaluation
- Overall Satisfaction

A sample of the questionnaire is attached in Appendix C. The results are summarized as shown in Table 37 and Table 38 below.

Table 37 Test Scenario 1

Test Scenario 1: In this situation, users are testing the efficiency and responsiveness of the main modules in PoultryPlus.						
No	Metric	Rating (Easy → Difficult)				
		1	2	3	4	5
1	Login Module	8	0	0	0	0
2	Dashboard Module	7	1	0	0	0
3	Setup Module (Farm, Shed, Vaccine, Feed, Chicks, Mortality)	6	2	0	0	0

4	Flock Management Module	6	2	0	0	0
5	Feed Inventory Module	6	2	0	0	0
6	Vaccine Inventory Module	5	3	0	0	0
7	Report Module	7	1	0	0	0
8	Daily Activity (Feed/Vaccine/Weight/Mortality) Recording	5	3	0	0	0
9	Alert Module	5	2	1	0	0

Evaluation on Test Scenario 1

Based on the results in Table 37, the majority of the eight respondents found the modules in PoultryPlus easy to use, with over 98% of the responses rated as either very easy (1) or easy (2). Notably, the Login Module received a perfect score, with all users rating it as very easy to use. Similarly, the Dashboard, Setup, Report, and Alert Modules were consistently rated as easy, with no scores above 2.

Some variation was observed in the Flock Management, Vaccine Inventory, and Daily Activity Recording Modules, where a few users gave a rating of 3. These users noted that while the core functionality was effective, certain forms could benefit from clearer instructions or more descriptive field labels to improve user understanding.

Importantly, no module received a rating of 4 or 5, indicating that none of the functionalities were perceived as difficult. This reflects strong usability and thoughtful user interface design across the PoultryPlus system.

Table 38 Test Scenario 2

Test Scenario 2:						
In this situation, users are asked to evaluate the layout, clarity, and usability of the interface.						
No	Metric	Rating (Disagree → Agree)				
		1	2	3	4	5
1	The buttons are easy to find and clearly named	0	0	0	2	6
2	The form fields are labeled clearly and understandable	0	0	0	3	5
3	Elements are logically grouped and organized	0	0	0	3	5
4	Icons and visuals are meaningful and help understanding the purpose	0	0	1	1	6
5	Error or alert messages are clear and helpful	0	0	0	0	8
6	Overall layout is clean and not cluttered	0	0	1	1	6
7	The system is user-friendly and intuitive to use	0	0	0	2	6

Evaluation on Test Scenario 2

Based on the results in Table 38, the overall user feedback on the PoultryPlus interface is highly positive. Most respondents agreed that the system is clear, well-organized, and easy to use. For example, 6 out of 8 users strongly agreed that the buttons are easy to find, the layout is clean, and the system is intuitive. Additionally, all 8 users fully agreed that error or alert messages are clear and helpful, indicating effective communication of system responses. Likewise, key interface components such as field labels, logical grouping of elements, and meaningful icons also received mostly level 4 or 5 ratings, showing general satisfaction. These results indicate that the GUI design successfully supports usability and user-friendliness, contributing to a smooth user experience across modules. Minor improvements could still be made by refining icon clarity and adding more intuitive visual cues, but overall, the interface meets user expectations very well.

5.4 Summary

This chapter presented the testing phase of the PoultryPlus system, focusing on both functionality and usability. Black-box testing was conducted on core modules such as Login, Flock Management, Feed Inventory, Vaccine Management, and Setup to ensure that each component performed according to specifications. All test cases passed successfully, confirming the system's reliability in handling user input and processes.

Usability testing was carried out with eight respondents to evaluate user satisfaction, efficiency, and interface design. Test Scenario 1 showed that the majority of users found the system modules easy to use, with no ratings indicating significant difficulty. Test Scenario 2

highlighted strong agreement on the clarity of buttons, forms, layout, and overall user-friendliness, demonstrating that the interface is well-structured and intuitive. Overall, the testing results affirm that PoultryPlus is functionally stable, efficient, and user-centred, with only minor areas for future enhancement in interface refinement.

Chapter 6 : CONCLUSION AND FUTURE WORK

6.1 Introduction

This chapter presents the conclusion of the PoultryPlus system and outlines possible future enhancements. It summarizes the key achievements and limitations observed during the development and testing phases. Suggestions for further improvement are also discussed based on usability test results and user feedback.

6.2 Project Achievement

PoultryPlus successfully achieved its objectives by providing an efficient and user-friendly platform for managing poultry farm operations. The system enables users to record and monitor critical data such as flock performance, feed usage, vaccine application, daily activity logs, and mortality records. Through proper role-based access control, both managers and workers can perform their respective duties effectively. The system's modular structure makes it scalable and adaptable to various farm environments. Additionally, usability testing showed positive responses for most functional modules, including the Login, Dashboard, and Inventory components. Users found these modules efficient, indicating that the system meets practical operational needs.

6.3 Project Limitation

Although PoultryPlus successfully meets its core functional goals, several limitations were identified that affect user experience and system completeness. These limitations, while not critical, indicate areas for future enhancement. Table 39 summarizes the key limitations observed during development and usability testing.

Table 39 Limitations of PoultryPlus

Limitations	Reasons
No in-system notification for alerts or feedback	The current version lacks real-time notifications or feedback messages to guide users, such as confirmation prompts or warnings for incomplete inputs.
Basic visual design	Icons and layout were considered too basic or non-intuitive by several users. This affected the perceived ease of use and overall user experience.
No integrated communication module	There is no internal messaging feature within the system for communication between workers and managers.

6.4 Future Work

To move beyond basic digital record-keeping and transform PoultryPlus into a smart farm management system, the following technological enhancements are proposed:

1. IoT Sensor Integration

Connect PoultryPlus to IoT devices for automatic environmental monitoring—such as temperature, humidity, and feed levels. Real-time data from sensors can enhance accuracy, reduce manual data entry, and trigger automated alerts when thresholds are breached.

2. Predictive Analytics for Flock Health

Integrate machine learning models to analyze historical flock data and predict potential health issues or performance declines. Early warnings can support proactive health management and reduce mortality.

3. Automated Feed and Water Tracking

Enable data input from automated feeding and watering systems to track consumption trends. This integration would reduce manual workload and provide a more precise understanding of flock behavior and efficiency.

4. Mobile and Offline Capability

Develop a mobile version of PoultryPlus with offline support. This would be especially valuable for farm workers operating in remote areas without stable internet connectivity.

5. Integration with E-Vet Systems

Incorporate external veterinary systems for direct access to prescriptions, health alerts, and vaccination recommendations. This feature would enhance decision support for disease prevention and treatment.

6. Blockchain for Data Integrity

Explore blockchain-based data logging to ensure the traceability and security of critical data such as vaccination, feed intake, and mortality. This is especially useful for audits, certifications, and product traceability in the supply chain.

7. Cycle and Economic Analysis Module

Introduce advanced tracking of flock cycles and integrate cost inputs such as feed price, labor, and medication. The system can generate profitability metrics for each batch to assist managers in making more informed business decisions.

6.5 Conclusion

In conclusion, PoultryPlus has successfully addressed a key need in the poultry farming industry by delivering a digital, modular, and accessible management platform. It empowers

both workers and managers with accurate, real-time insights and simplifies daily operations. Despite some usability limitations, feedback from actual farm users has been overwhelmingly positive.

Looking forward, PoultryPlus has the potential to evolve into a smart farm ecosystem by leveraging emerging technologies such as IoT, predictive analytics, and blockchain. These enhancements will not only automate routine operations but also elevate data-driven decision-making, reduce risks, and increase farm productivity. With continuous innovation, PoultryPlus can become a next-generation tool for sustainable and intelligent poultry farm management.

References

- Dhillon, R., & Moncur, Q. (2023). Small-Scale Farming: A review of challenges and potential opportunities offered by technological advancements. *Sustainability*, 15(21), 15478. <https://doi.org/10.3390/su152115478>
- FarmBrite | Multi-Farm and farm co-op management software. (n.d.). Farmbrite. <https://www.farmbrite.com/farm-cooperative-management#:~:text=Full%20Farmbrite%20Suite,them%20more%20organized%20and%20profitable.>
- Garner, B. (2021, December 10). *React on Django: Getting started*. Medium. <https://bennettgarner.medium.com/react-on-django-getting-started-f30de8d23504>
- Getachew, S. (2024, September 17). Building Scalable APIs with Django REST Framework: Best Practices for 2024. Medium. <https://medium.com/django-unleashed/building-scalable-apis-with-django-rest-framework-best-practices-for-2024-7a2e4856c2c0>
- Martinez, A. a. G., De Alencar Nääs, I., Abe, J. M., & Pereira, D. F. (2021). A mobile application to follow up the management of broiler flocks. *AgriEngineering*, 3(4), 990–1000. <https://doi.org/10.3390/agriengineering3040062>
- Navfarm. (2019, February 10). Navfarm. Navfarm. <https://navfarm.wordpress.com/>
- Nillofar, P., Francis, D. P., Lazarova-Molnar, S., Vulpe, A., Vochin, M., Suciu, G., Balanescu, M., Anestis, V., & Bartzanas, T. (2021). Data-driven decision support in livestock farming for improved animal health, welfare, and greenhouse gas emissions: Overview and challenges. *Computers and Electronics in Agriculture*, 190, 106406. <https://doi.org/10.1016/j.compag.2021.106406>
- Pradeep, & Pradeep. (2020, February 17). Broiler Management System Integration - NavFarm Blog. Navfarm Blog - NavFarm. [https://www.navfarm.com/blog/broiler-management-system-integration/#:~:text=Broiler%20management%20system%20\(BMS%20System,%2C%20Feed%20Formulation%2C%20and%20Accounting.](https://www.navfarm.com/blog/broiler-management-system-integration/#:~:text=Broiler%20management%20system%20(BMS%20System,%2C%20Feed%20Formulation%2C%20and%20Accounting.)
- Sajna, K. (2022, May 17). SDLC methodologies — which one to choose for your project? CodiLime. <https://codilime.com/blog/sdlc-methodologies/>
- Salma. (2023, April 4). Introducing BroilerX: Manage and monitor your broiler farm at the touch of a button. Universitas Gadjah Mada. <https://ugm.ac.id/en/news/21548-introducing-broilerx-manage-and-monitor-your-broiler-farm-at-the-touch-of-a-button/#:~:text=It%20brings%20about%20smart%20farming%20and%20technological,of%20the%20UGM%20Faculty%20of%20Animal%20Science>
- Schwaber, K. and Beedle, M. (2002) *Agile Software Development with Scrum*. Prentice-Hall, Upper Saddle River.
- Wu, D., Cui, D., Zhou, M., & Ying, Y. (2022). Information perception in modern poultry farming: A review. *Computers and Electronics in Agriculture*, 199, 107131. <https://doi.org/10.1016/j.compag.2022.107131>

Appendix A

```
const handleSubmit = async (e) => {
  e.preventDefault();

  // Check if passwords match
  if (password !== confirmPassword) {
    setError("Passwords do not match");
    return;
  }
  setError('');
  setLoading(true);

  const userData = {
    passport_number: passportNumber,
    user_name: userName,
    date_of_birth: dateOfBirth,
    nationality: nationality,
    gender: gender,
    password: password,
    phone_no: phoneNumber,
  };

  try {
    const response = await fetch('http://localhost:8000/api/register/', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(userData),
    });

    const result = await response.json();

    if (response.ok) {
      alert("Registered successfully!");
      navigate('/login'); // Redirect to login page
    } else {
      setError(result.error || 'Registration failed');
    }
  } catch (error) {
    console.error('Error:', error);
    setError('An error occurred. Please try again.');
```

```
  } finally {
    setLoading(false);
  }
};
```

Figure A-1 Register.js handleSubmit component full code

```

from django.views.decorators.csrf import csrf_exempt
from django.views.decorators.http import require_http_methods # ✅ Import added
from django.http import JsonResponse
import json
from .models import FarmWorker
from .serializers import FarmWorkerSerializer # ✅ Import added
from django.contrib.auth.hashers import make_password, check_password
from datetime import datetime, date

@csrf_exempt
def register_user(request):
    if request.method == 'POST':
        data = json.loads(request.body)

        try:
            passport_number = data['passport_number']
            user_name = data['user_name']
            date_of_birth = datetime.strptime(data['date_of_birth'], "%Y-%m-%d").date()
            nationality = data['nationality']
            phone_no = data['phone_no']
            gender = data['gender']
            hashed_password = make_password(data['password'])

            age = date.today().year - date_of_birth.year - ((date.today().month, date.today().day) <
                (date_of_birth.month, date_of_birth.day))

            if FarmWorker.objects.filter(passport_number=passport_number).exists():
                return JsonResponse({'error': 'User already exists'}, status=400)

            FarmWorker.objects.create(
                passport_number=passport_number,
                user_name=user_name,
                date_of_birth=date_of_birth,
                nationality=nationality,
                phone_no=phone_no,
                gender=gender,
                password=hashed_password,
                age=age,
                register_date=date.today()
            )
            return JsonResponse({'message': 'User registered successfully'})
        except Exception as e:
            return JsonResponse({'error': str(e)}, status=500)
    else:
        return JsonResponse({'error': 'This endpoint only supports POST requests.'}, status=405)

```

Figure A-2 Users view.py

```

from django.db import models
from datetime import date

# Create your models here.

class FarmWorker(models.Model):
    passport_number = models.CharField(max_length=50, primary_key=True, unique=True)
    password = models.CharField(max_length=256) # Ensure to handle password securely
    user_name = models.CharField(max_length=100, null=False)
    date_of_birth = models.DateField(null=False)
    nationality = models.CharField(max_length=50, null=False)

    # Define gender choices outside of the model
    GENDER_CHOICES = [
        ('M', 'Male'),
        ('F', 'Female'),
    ]

    gender = models.CharField(max_length=1, choices=GENDER_CHOICES, default='M') # e.g., Male, Female

    phone_no = models.CharField(max_length=15)
    register_date = models.DateField(auto_now_add=True)

    # Add role choices
    ROLE_CHOICES = [
        ('worker', 'Worker'),
        ('manager', 'Manager'),
    ]
    role = models.CharField(max_length=20, choices=ROLE_CHOICES, default='worker') # New field

    APPROVAL_STATUS = [
        ('pending', 'Pending'),
        ('approved', 'Approved'),
        ('rejected', 'Rejected'),
    ]
    approval_status = models.CharField(max_length=10, choices=APPROVAL_STATUS, default='pending')

    age = models.IntegerField(null=True, blank=True)

    def is_manager(self):
        return self.role == 'manager'

    def calculate_age(self):
        today = date.today()
        return today.year - self.date_of_birth.year - (
            (today.month, today.day) < (self.date_of_birth.month, self.date_of_birth.day)
        )

    def __str__(self):
        return f"{self.user_name} ({self.get_role_display()}, {self.get_approval_status_display()})"

```

Figure A 3 Users model

Appendix B

Interview Questions for Requirement Collection

These questions were used during interviews with the farm manager (Mr. William) and 6 farm workers from Farm C, Farm E, and Farm G. The purpose was to identify current challenges

1. Current Practices

- How do you currently record farm data (e.g., mortality, feed)?
- What problems do you face with this method?
- Is it easy to access or share this data with others?

2. Daily Operations

- How do you keep track of chicken deaths?
- How do you record feed usage?
- Do you use a fixed format or template for daily records?
- How is chicken harvesting data recorded (e.g., weight)?

3. Vaccination & Health

- How do you manage the vaccination schedule?
- How do you make sure all chickens are vaccinated on time?
- Do you record missed or delayed vaccinations?

4. Shed & Cycle Tracking

- Do you track cycle numbers for each shed?
- Why is it important to separate cycles?
- Is it difficult to manage multiple sheds or cycles?

5. Reports & KPIs

- What performance metrics do you usually monitor?
- Would it help if the system calculated these automatically?
- How do you usually generate reports now?

6. User Preferences

- What issues do you face when using digital tools?
- Do you prefer using the system on a phone or computer?
- Is there a preferred language or wording style for the system?

7. Communication & Alerts

- How do workers and managers communicate issues currently?
- Would internal messages or notifications help?
- Would real-time alerts (e.g., missed vaccine) be useful?

8. Feature Needs

- What features do you wish a system like this could offer?
- Would a dashboard for quick summary be helpful?
- Should the system work offline or with weak internet?

9. Access & Roles

- What types of users should the system support (e.g., worker, manager)?
- Should some features only be available to managers?
- Is login/password protection important?

Appendix C

Usability Testing of PoultryPlus System

Please answer the following questions by putting a tick (✓) in the appropriate box.

Part 1: System Efficiency Evaluation

Test Scenario 1:

In this situation, users are testing the efficiency and responsiveness of the main modules in PoultryPlus.

No	Metric	Rating (Easy → Difficult)				
		1	2	3	4	5
1	Login Module					
2	Dashboard Module					
3	Setup Module (Farm, Shed, Vaccine, Feed, Chicks, Mortality)					
4	Flock Management Module					
5	Feed Inventory Module					
6	Vaccine Inventory Module					
7	Report Module					
8	Daily Activity (Feed/Vaccine/Weight/Mortality) Recording					
9	Alert Module					

2. Are the results generated by the system accurate and aligned with your expectations?

Yes	
No	

3. Please rate the response time and loading speed of the system.

Very fast	
Average	
Slow	
Very slow	

4. Is it easy to navigate between different modules in the system?

Yes	
No	
Sometimes	

5. How often did you encounter errors or bugs while using the system?

Never	
Rarely	
Sometimes	
Often	

Part 2: Graphical User Interface (GUI) Evaluation

Test Scenario 2:

In this situation, users are asked to evaluate the layout, clarity, and usability of the interface.

No	Metric	Rating (Disagree → Agree)				
		1	2	3	4	5
1	The buttons are easy to find and clearly named					
2	The form fields are labeled clearly and understandable					
3	Elements are logically grouped and organized					
4	Icons and visuals are meaningful and help					

	understanding the purpose					
5	Error or alert messages are clear and helpful					
6	Overall layout is clean and not cluttered					
7	The system is user-friendly and intuitive to use					

2. Please rate the stability and consistency of the system's design.

Very stable	
Average	
Unstable	
Very unstable	

Part 3: Overall Satisfaction

1. How satisfied are you with the system overall?

Very satisfied	
Satisfied	
Neutral	
Dissatisfied	
Very dissatisfied	

2. Would you recommend PoultryPlus to others in poultry farming?

Yes	
No	
Maybe	

3. Any suggestions for improvement or features you'd like to see?

(Please write your answer below)
