SWIFTLET BIRD'S NEST HARVESTING MANAGEMENT TRACKING SYSTEM

ERIC LEON GOM

This project is submitted in partial fulfillment of the requirements for the degree of Bachelor of Software Engineering with Honours

Faculty of Computer Science and Information Technology

UNIVERSITI MALAYSIA SARAWAK

2025

UNIVERSITI MALAYSIA SARAWAK

THESIS STATUS ENDORSEMENT FORM

TITLE SWIFTLET BIRD'S NEST HARVESTING TRACKING MANAGEMENT SYSTEM

ACADEMIC SESSION: 2024/2025

(CAPITAL LETTERS)

hereby agree that this Thesis* shall be kept at the Centre for Academic Information Services, Universiti Malaysia Sarawak, subject to the following terms and conditions:

1. The Thesis is solely owned by Universiti Malaysia Sarawak
2. The Centre for Academic Information Services is given full rights to produce copies for educational purposes only
3. The Centre for Academic Information Services is given full rights to do digitization in order to develop local content database
4. The Centre for Academic Information Services is given full rights to produce copies of this Thesis as part of its exchange item program between Higher Learning Institutions [or for the purpose of interlibrary loan between HLI]
5. ** Please tick (√)

☐	CONFIDENTIAL	(Contains classified information bounded by the OFFICIAL SECRETS ACT 1972)
☐	RESTRICTED	(Contains restricted information as dictated by the body or organization where the research was conducted)
☐	UNRESTRICTED	

(AUTHOR'S SIGNATURE)

Validated by

(SUPERVISOR'S SIGNATURE)

Permanent Address

KAMPUNG RAMAYAH
PENAMPANG
SABAH

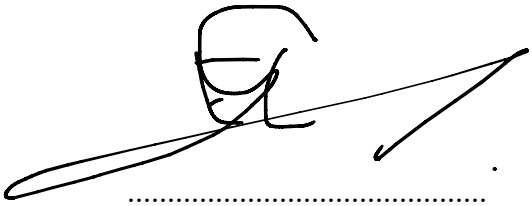
Date: 16th JANUARY 2025

Date: 16/01/2025

Note * Thesis refers to PhD, Master, and Bachelor Degree
** For Confidential or Restricted materials, please attach relevant documents from relevant organizations / authorities

Declaration

I hereby declare that this project is my original work. I have not copied from any other student's work or from any other sources except where due reference acknowledgement is not made explicitly in the text, nor has any part had been written for me by another person.



.....

(ERIC LEON GOM) 16th January 2025

Faculty of Computer Science and Information Technology

Universiti Malaysia Sarawak

Acknowledgement

I would like to express my deepest gratitude to everyone who has contributed to the successful completion of my Final Year Project, which is the Swiftlet Bird's Nest Harvesting Management Tracking System. First and foremost, I would like to extend my sincerest appreciation to my academic supervisor, Madam Hamizan binti Sharbini, for her invaluable guidance, encouragement, and constructive feedback throughout the entire journey. The expertise and patience were instrumental in shaping the direction and quality of the project. I would also like to thank the Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak, for providing the necessary resources and facilities that enabled the smooth execution of this project. Furthermore, I am grateful to my stakeholders, especially the swiftlet farmers, for their patience and willingness to share insights and experiences that greatly enriched the system's design and functionality. I would like to offer my special thanks to my family and friends for their unwavering support, motivation, and understanding during this period of challenges. Their belief in my capabilities gave me the strength to persevere. Finally, I would like to acknowledge the authors of the literature and research that inspired and guided the development of this project. To all who have contributed directly or indirectly, I am deeply grateful for your support.

Abstract

The Swiftlet Bird's Nest Harvesting Management Tracking System was developed to address operational challenges faced by swiftlet farmers, such as inefficient manual data tracking, lack of structured task planning, and the absence of digital reporting tools. This web-based system aims to streamline and centralize the farm management process through a user-friendly interface. Built using the Agile Scrum methodology, development was conducted in iterative sprints, with continuous refinement based on real stakeholder feedback. Key features include logging nest data, managing expenses, scheduling cleaning or harvesting tasks, and generating downloadable reports. Stakeholder requirements were gathered via interviews and walkthroughs, and the system was developed using tools like Visual Studio Code, XAMPP, and MySQL. Acceptance testing was conducted with a swiftlet birdhouse owner who participated in the early design phases; the stakeholder accepted all major system functions and described the system as easy to use, well-structured, and tailored to the needs of swiftlet farming. Based on their feedback, the system achieved a 100% acceptance rate for core features. The system offers real-time data entry, secure database management, and customizable report generation, contributing to more efficient, traceable, and scalable swiftlet farming operations.

Table of Content

List of Figures

List of Tables

CHAPTER 1: INTRODUCTION

1.1 Background Studies

As swiftlet farming continues to grow in popularity and demand, managing the complex process of harvesting and maintaining high-quality edible bird nests has become a significant challenge for farmers. This challenge extends to managing swiftlet birdhouses, which require careful oversight to ensure optimal conditions for swiftlet nesting and productivity. To address these operational needs, this project focuses on the development of an online platform called the Swiftlet Bird's Nest Harvesting Management Tracking System, designed specifically to simplify farm management and enhance the organization of data in swiftlet farming.

The proposed system provides tools to streamline essential processes such as managing multiple swiftlet birdhouses, harvest scheduling, inventory tracking, and quality assessment. Farmers will be able to manually log key information into a centralized web-based platform, including data related to individual birdhouses, such as nest production, quality, and environmental conditions. The system incorporates CRUD (Create, Read, Update, Delete) functionality, enabling users to create new records for harvests, inventory, and quality data, read and review logged data for specific birdhouses, update existing records to ensure data accuracy, and delete outdated or incorrect entries to maintain a clean and organized database.

Additionally, the system includes features for generating detailed productivity reports and visual tools for trend analysis. Using the data logged through CRUD processes, the system automates the creation of actionable reports, providing farmers with valuable insights into the performance of each birdhouse and overall farm productivity over time. These tools enable

informed decision-making, helping farmers optimize operations and improve yields across their birdhouses.

By implementing this system, the project aims to reduce the administrative workload on swiftlet farmers, enabling them to focus on improving production while ensuring accurate and transparent data management. This inventory and tracking solution are tailored to support the growth of the swiftlet farming industry by promoting long-term sustainability and scalability through efficient digital management techniques for both individual birdhouses and entire farms.

1.2 Problem Statement

As swiftlet farming grows in Sabah, Malaysia, farmers face several challenges in managing their bird nest harvesting operations. Currently, many farmers rely on manual methods to track harvest schedules, record nest production, and maintain inventory. This lack of automation often leads to missed harvests, inaccurate records, and difficulties in assessing farm performance over time. Additionally, without a centralized tracking system, decision-making is hindered, affecting profitability and overall productivity. No other web-based solutions currently address the specific needs of swiftlet farmers, which include efficient harvest scheduling, production tracking, and inventory management. This manual approach has also led to repetitive, time-consuming tasks in logging harvests and monitoring nest quality and weight, risking mismanagement and loss of data accuracy. Without a streamlined system, farmers face potential operational delays and reduced profitability in an increasingly competitive industry.

1.3 Objectives

1. To design the management system to manage swiftlet houses, harvest schedules, inventory, and farm records, ensuring efficient and organized farm operations.
2. To develop the tracking system to log and monitor bird nest harvests, including data such as nest position, date, quantity, quality, and weight, providing real-time insights into farm productivity.
3. To generate analytical reports that identify trends and improve decision-making in nest production.

1.4 Scope

1. Only registered swiftlet farmers can access the system.
2. The prototype will be web-based, accessible on desktop and mobile browsers.
3. Real-time tracking and reports will focus on nest production and inventory.

1.5 Methodology

For the development of the Swiftlet Bird's Nest Harvesting Management Tracking System, an Agile methodology, specifically SCRUM, has been chosen to ensure flexibility and efficiency in meeting the project's objectives. This approach enables incremental progress through small, manageable tasks, allowing the project team to develop the system with continuous feedback and improvements (Sutherland & Schwaber, 2017). Agile methodology divides the project into smaller, more achievable goals or user stories, which can be completed in cycles called sprints. These iterations allow the project to continuously evolve, with frequent adjustments based on user feedback and performance evaluation (VersionOne, 2020).

SCRUM, as a specific framework within Agile, organizes tasks into short, time-boxed sprints, typically lasting 1-4 weeks, and each sprint focuses on delivering a specific set of priorities from the product backlog (Sutherland & Schwaber, 2017). The SCRUM framework emphasizes key roles such as the SCRUM Master, Product Owner, and development team, who work together to ensure smooth communication and clarity of objectives. Each sprint begins with a planning meeting, where the team selects which tasks will be prioritized based on their value and complexity. The team then proceeds to design, develop, and test these features, ensuring that a potentially shippable product increment is delivered by the end of the sprint (Sutherland & Schwaber, 2017).

The process follows a cycle of planning, design, development, testing, and deployment, with regular reviews and retrospectives to ensure that the system meets the evolving needs of swiftlet farmers. At the end of each sprint, a sprint review meeting allows stakeholders to evaluate the progress and provide feedback, which is then incorporated into future work. Additionally, a sprint retrospective is held to reflect on the team's performance and identify process improvements for future sprints. These continuous cycles of evaluation and improvement ensure that the project can quickly adapt to changes, whether in user needs, technological advancements, or unforeseen challenges (VersionOne, 2020). By adopting SCRUM, the team can prioritize high-value features and resolve issues efficiently, ensuring that the final product remains functional and user-friendly (Beck et al., 2001).

This methodology will allow for quick adaptation to user feedback, enabling the development team to address any changes or new requirements. SCRUM’s iterative approach fosters continuous improvement and collaboration, ensuring the project is delivered within the desired timeline and meets the needs of swiftlet farmers effectively.

1.6 Project Outcomes

Swiftlet farmers will have access to a centralized harvest scheduling system through Swiftlet Bird's Nest Harvesting Management Tracking System, which will enable them to log and track harvest dates manually to ensure timely collections. In addition to providing data analytics and reporting tools that generate insights into production trends to improve farming operations, the system will facilitate efficient inventory management by allowing users to manually input and monitor the position, quantity, weight, and quality of collected nests. By preserving thorough harvest records through its CRUD (Create, Read, Update, Delete) functionality, the platform simplifies record-keeping processes and ensures data accuracy. Farmers will be able to access and manage their operations from any device through its mobile-responsive web-based design. With scalability for future improvements, the application's goal is to support farm production and profitability through data-driven decision-making and streamlined digital workflows.

1.7 Study Schedule

The Gantt chart, Figure 1, lays out the timeline and key checkpoints for the FYP1 (Final Year Project 1) progress. It starts with the "Topic Selection & Approval" phase from September 30 to October 06, 2024, followed by the "Submission of Approved Brief Proposal" on October 20, 2024. The next milestone is the "Submission of Full Proposal" on October 28, 2024, marked by a yellow bar. The subsequent tasks include the "Submission of Chapter 1" on November 14, 2024, and the "Submission of Chapter 2" on December 13, 2024. The final tasks are the "Submission of Chapter 3" on January 5, 2025, and the "Submission of FYP 1 Final Report & Paper for Assessment" on January 12, 2025, which conclude the project's timeline. The chart uses color-coded bars to visually represent each task, providing a clear and concise overview of the project milestones.

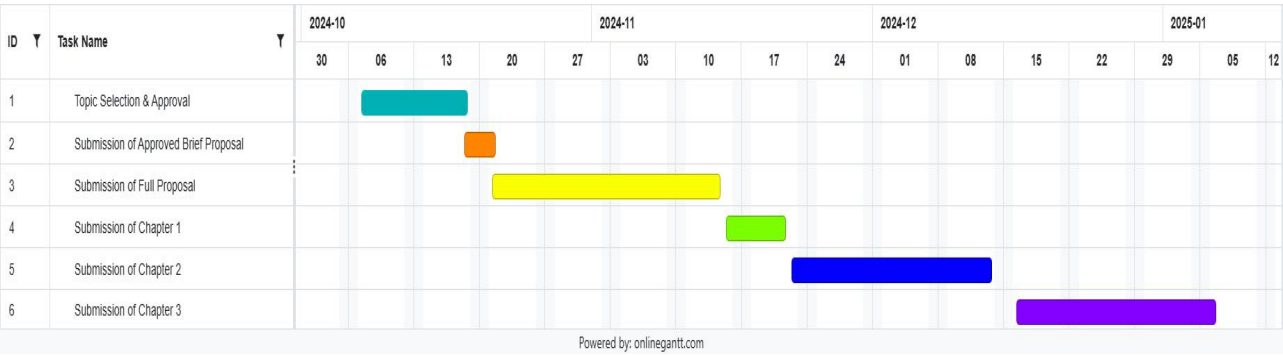


Figure 1: Gantt Chart

1.8 Expected Outcome

It is expected that this project will provide swiftlet farmers with a reliable tool to manage their bird nest harvesting operations. The web-based system will offer features such as logging harvest dates, tracking nest quantity, quality, and weight, generating detailed productivity reports, and automating harvest scheduling. All data will be stored in a fully operational database, ensuring accuracy and accessibility for farmers. The system will improve farm management efficiency, enabling farmers to make informed decisions that enhance productivity and profitability.

1.9 Report Outline

Chapter 1 provides a comprehensive introduction to the project, focusing on the rationale for developing Swiftlet Bird's Nest Harvesting Management Tracking System. It includes the background of the study, identifies the challenges faced by swiftlet farmers, and articulates the problem statement. This chapter also defines the project's objectives, scope, and significance, highlighting its potential to improve operational efficiency and support the growth of the swiftlet farming industry.

Chapter 2: Literature Review

2.1 Introduction

Chapter 2 focuses on the review of the three existing systems such as AgriWebb, Cropio, and Herdwatch to provide the derivation of value insights for the Swiftlet Bird's Nest Harvesting Management Tracking Systems' development. The main objective of this review prioritizes the analysis of features, functionalities, benefits, and drawbacks of these systems. A comprehensive comparison will be conducted to underline their strengths and weaknesses.

Additionally, this chapter also identifies the tools that are required for designing and implementing the Swiftlet Bird's Nest Harvesting Management Tracking System. These tools are divided into two main groups which are the backend development tools and supporting software. The goal of this analysis is to identify the essential resources required for the successful creation and implementation of the suggested system.

2.2 Review on Similar Existing System

2.2.1 AgriWebb

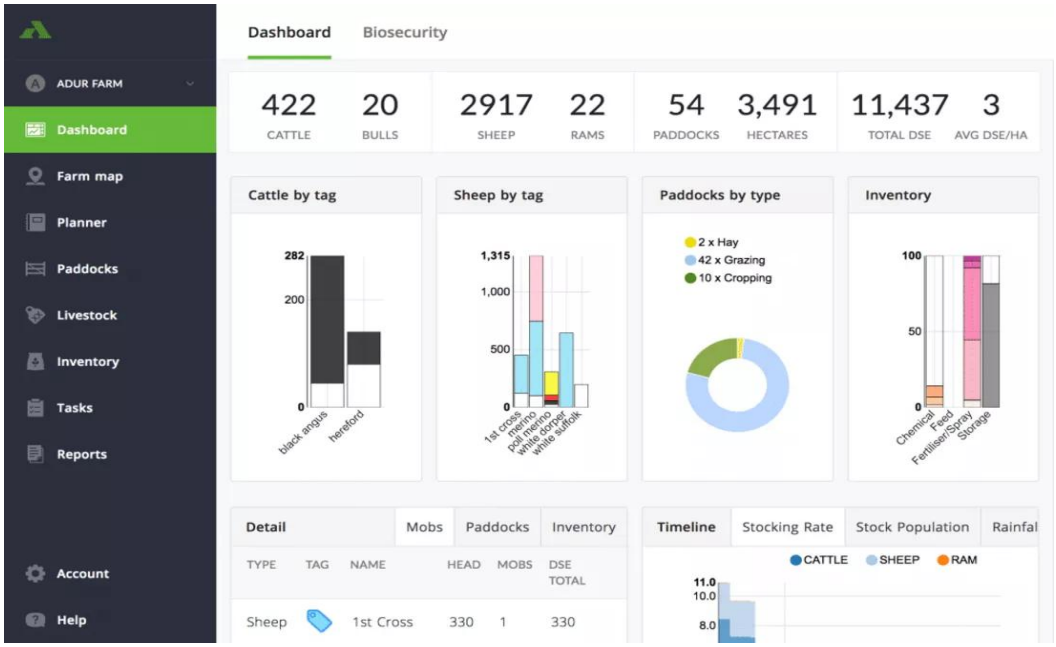


Figure 2: AgriWebb Dashboard

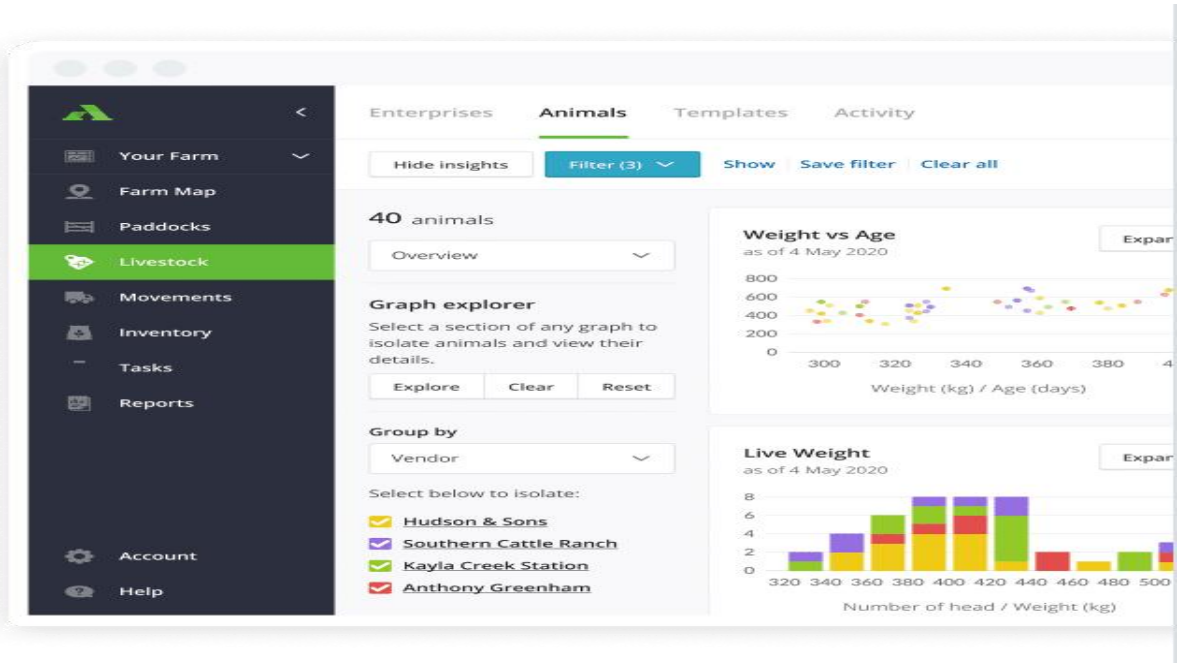


Figure 3: AgriWebb Livestock

Agriwebb is a livestock management software that helps farmers securely record data for farm decisions and audit purposes. (Department of Primary Industries and Regions, South Australia, 2024). In Figure 2, the AgriWebb dashboard displays the overall information such as the number of animals, field size, details of livestock and more. This saves time of inputting paper-based data from staff into a management system (Department of Primary Industries and Regions, South Australia, 2024). This user-friendly interface highlights the potential of leveraging digital tools for streamlined management and decision-making. Figure 3 highlights the "Livestock" tab in the AgriWebb dashboard, which provides detailed insights into livestock data such as movements, inventory, and live weight distribution. Interactive graphs and filtering options allow users to isolate specific data for better analysis, making it easier to track livestock performance and manage operations efficiently.

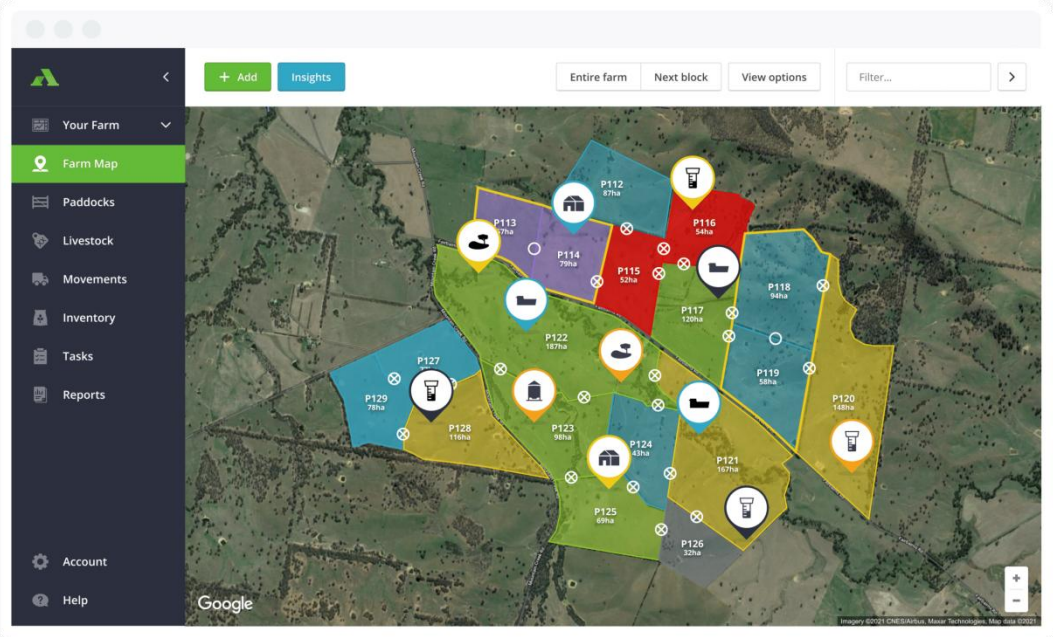


Figure 4: AgriWebb Map

Figure 4 displays the AgriWebb's map that provides a comprehensive view of the user's farm, allowing the user to track herd movements and essential tasks effortlessly (*AgriWebB: Revolutionizing Farm Mapping and Management*, 2024.).

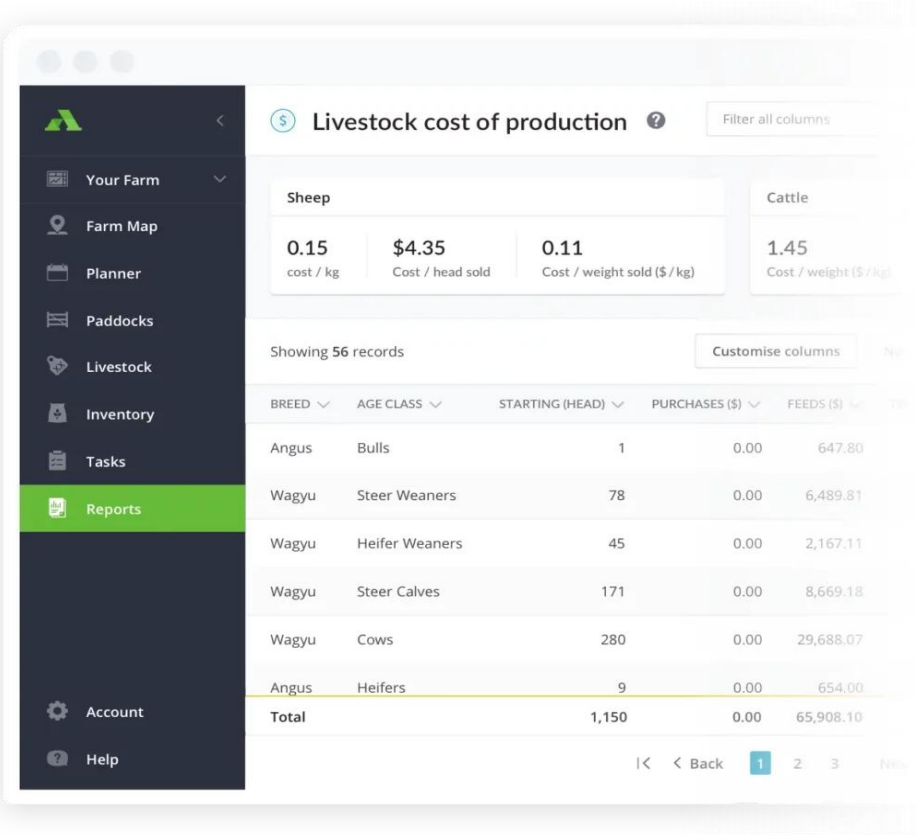


Figure 5: AgriWebb Reports

Figure 5 displays the Reports tab of the AgriWebb system, showcasing the Livestock Cost of Production for different animal categories, including sheep and cattle. The table provides detailed information on cost per kilogram, cost per head sold, and cost per weight sold for various breeds and age classes. This feature allows users to track and analyze the financial performance of their livestock operations by calculating and comparing the costs associated with production, purchases, and feed. It demonstrates how the system aids in financial planning and decision-making for livestock management (Pezzuolo et al., 2021).

2.2.2 Cropio

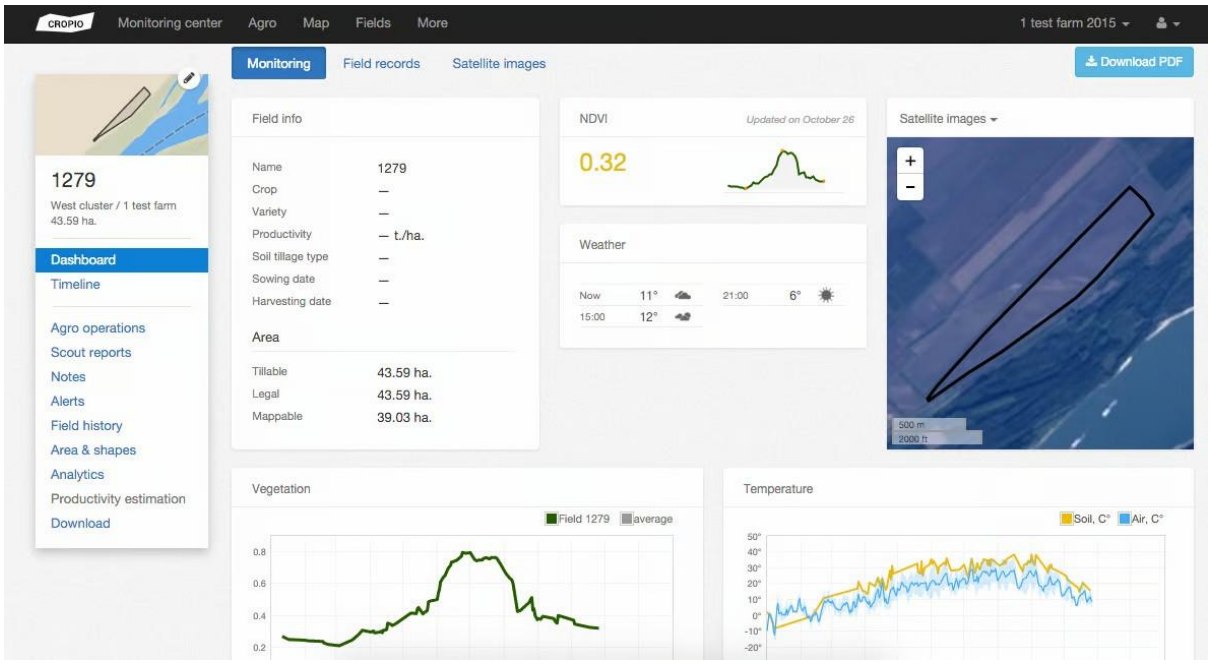


Figure 6: Cropio Dashboard

Figure 6 illustrates the dashboard of the Cropio platform, which is a digital solution designed for monitoring agricultural operations. The dashboard provides critical information on field monitoring, including NDVI (Normalized Difference Vegetation Index) for vegetation health, satellite imagery of the field area, and weather updates. It also offers data on field info such as crop type, productivity, soil tillage type, and harvest date, along with legal and mappable area statistics. The vegetation and temperature graphs allow users to assess crop health over time, making it a valuable tool for precision agriculture and decision-making. This system showcases how real-time data and analytics contribute to optimizing agricultural management practices.

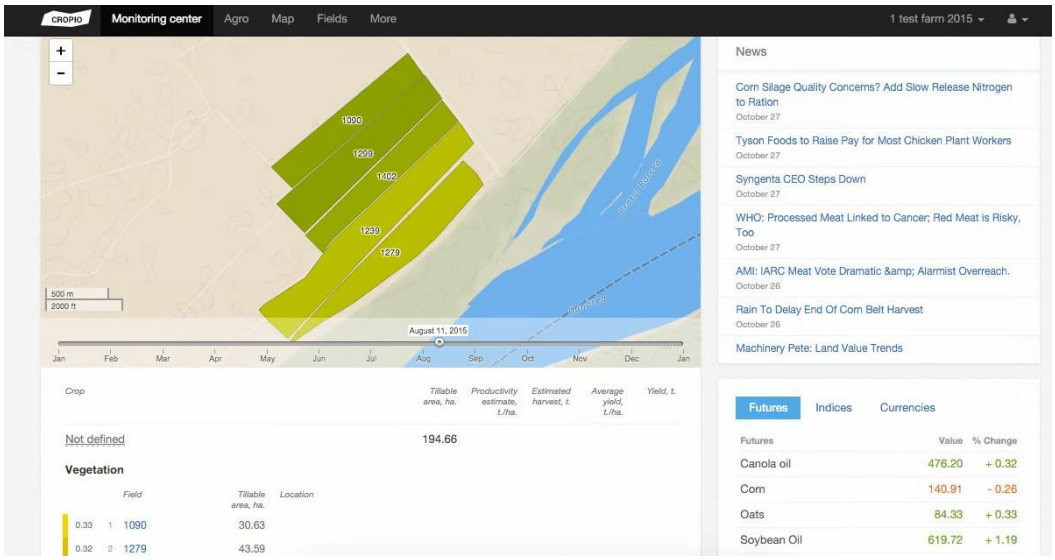


Figure 7: Cropio Monitoring Centre

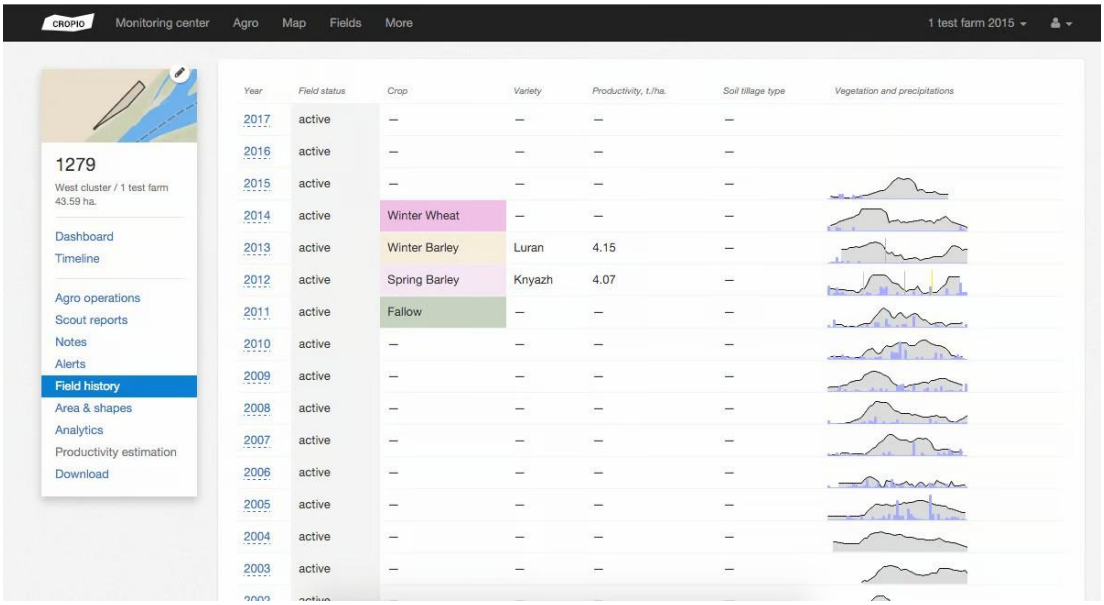


Figure 8: Cropio History

Figures 7 and 8 highlight Cropio's capabilities in monitoring and tracking agricultural activities. Figure 7 demonstrates the Monitoring Centre, which provides a geographic overview of farm fields, detailing vegetation indices, estimated yields, and field-specific conditions. This interface also integrates external agricultural news and updates on futures, indices, and commodity prices, supporting decision-making for farm productivity. Figure 8 illustrates the

History tab, offering a chronological record of field statuses and crop histories, including information on crops planted, varieties, productivity, soil tillage, and environmental factors like vegetation and precipitation trends.

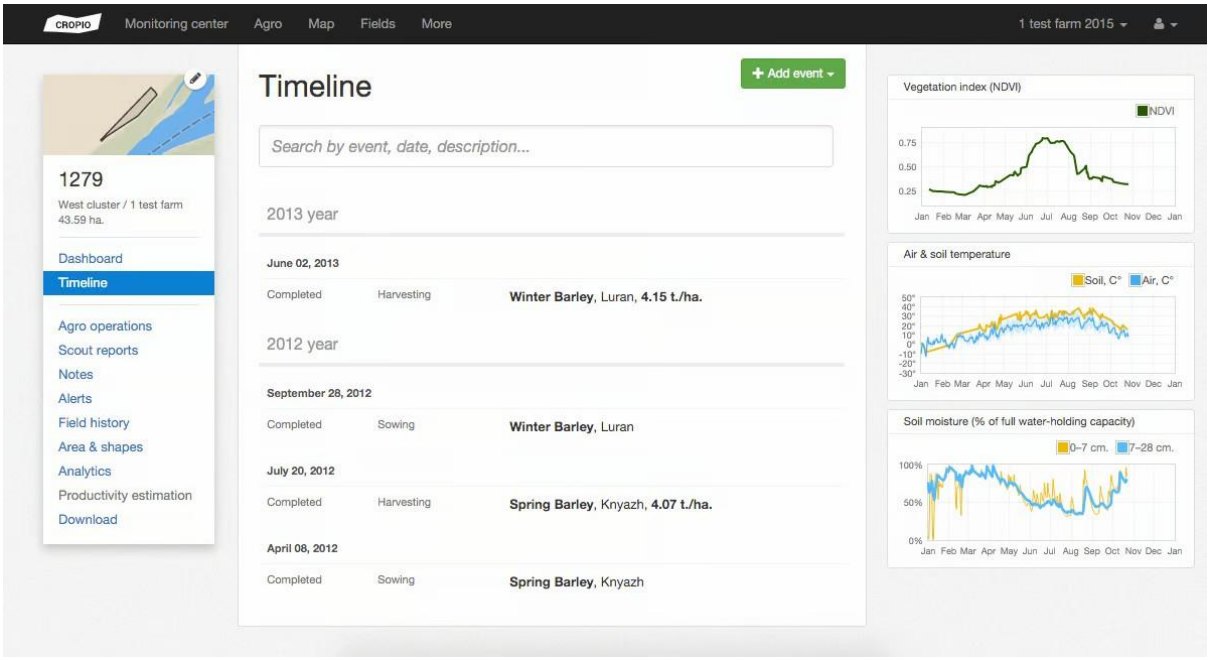


Figure 9: Cropio Timeline

Cropio's Timeline feature, displayed in Figure 9, showcases the potential of digital tools in precision agriculture. By providing a comprehensive historical record of field activities, including sowing and harvesting dates, crop rotations, and yield data, this feature empowers farmers to make data-driven decisions. The visual representation of this information facilitates the analysis of past performance, identification of trends, and optimization of future farming practices.

2.2.3 Herdwatch

Figure 10: Herdwatch Dashboard

HerdWatch (Figure 10) is a digital livestock management platform that empowers farmers with tools to optimize herd health and productivity. By leveraging technology, HerdWatch allows farmers to track and monitor various aspects of their livestock, including compliance, performance, breeding, farm management, and reporting. Real-time data access enables timely decision-making and proactive response to potential issues, ultimately contributing to improved efficiency and profitability in livestock farming.

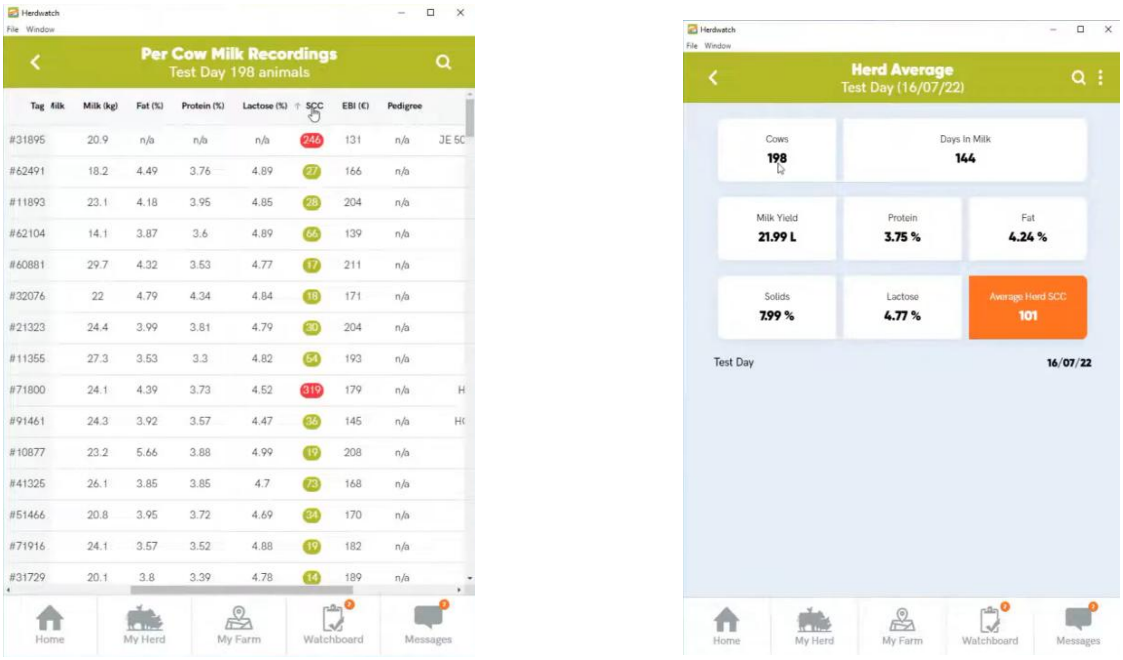


Figure 11: Herdwatch Records and Individual Records

HerdWatch provides detailed records of individual cows, including milk yield, protein, fat, and lactose content as displayed in Figure 11. This data enables farmers to monitor the performance of each animal and identify high-performing individuals for breeding purposes. Additionally, HerdWatch offers a comprehensive overview of herd performance, including average milk yield, protein, and fat content as displayed in Figure 11. This information helps farmers assess the overall health and productivity of their herd and make informed management decisions.

2.3 Comparison between proposed system and existing system

Table 1: Comparison between proposed system and existing systems

Feature	AgriWebb	Cropio	Herdwatch	Proposed System
Animal Monitoring	Yes	Yes	Yes	Yes
Alerts and Notifications	Yes	Yes	Yes	Yes
Real-time Data Analysis	Yes	Yes	No	Yes
Customizable Reports	Yes	Yes	No	Yes
Integration with Financial Management Tools	No	No	No	Yes
Enhanced Data Security	Yes	Yes	Yes	Yes

2.4 Discussion

The proposed Swiftlet Bird's Nest Harvesting Management Tracking System offers an optimized solution that improves traditional methods of tracking and managing bird nest

production. Unlike manual record-keeping, which can lead to disorganized records, scheduling inefficiencies, and difficulties in retrieving past data, the system provides a structured, centralized approach to nest tracking while ensuring that manual logging remains practical and efficient.

The process flow of the system involves farmers manually logging harvest-related data, including nest quantity, position, weight, quality, and environmental conditions (such as humidity and temperature) through an intuitive web interface. Since swiftlet farmers do not enter birdhouses daily, the system is designed to allow periodic updates that align with harvesting and cleaning schedules. All data, including nest weight measured using digital weighing scales, will be manually logged by the users, ensuring accuracy and control over data entry. Environmental conditions such as humidity and temperature will also be manually recorded based on farmers' observations.

To support efficient record-keeping, the system incorporates CRUD (Create, Read, Update, Delete) functionalities, allowing farmers to add, modify, and retrieve records seamlessly. This ensures flexibility in data management, prevents outdated or incorrect records, and maintains a well-organized and accessible database. The system's centralized platform is specifically designed for swiftlet birdhouse management, making it easier for farmers to track production cycles, analyze trends, and optimize harvesting schedules.

The system enhances tracking efficiency by incorporating automated reporting tools that analyze historical nest data and generate visual insights into productivity trends. These reports allow farmers to identify patterns in nest development, optimize harvest timing, and track changes in nest conditions over multiple cycles. The absence of automated IoT sensors ensures

that farmers retain full control over data accuracy, while real-time notifications assist in scheduling tasks effectively.

The system is accessible via multiple devices, including tablets, mobile phones, and computers, providing convenient data entry options for farmers in the field. Its mobile-responsive design enables on-the-go updates, ensuring that nest tracking remains structured, efficient, and adaptable to farming routines.

By enhancing, rather than replacing, manual logging, the system bridges the gap between traditional farming methods and modern digital solutions. Real-time inventory tracking ensures better monitoring of harvested nests, while data security measures maintain record integrity. The system is designed with scalability in mind, allowing for future enhancements, such as advanced analytics and automated reminders, to further improve tracking efficiency.

By addressing the limitations of manual tracking while preserving its practicality, the system ensures accurate, structured, and accessible data management. Its streamlined processes for data logging, tracking, and reporting help farmers focus on production planning while ensuring accuracy and traceability across the supply chain. This enhances profitability and positions the system as a reliable, scalable solution for the swiftlet farming industry.

2.5 Summary

In conclusion, this literature review has examined the current state of technology in agricultural management, focusing on systems like AgriWebb, Cropio, and HerdWatch, which have demonstrated the potential of technology to enhance efficiency, productivity, and sustainability in agriculture. While these systems are effective in general livestock and crop

management, niche agricultural sectors, such as the Swiftlet Bird's Nest industry, continue to face challenges that remain largely unaddressed.

The proposed Swiftlet Bird's Nest Harvesting Management Tracking System aims to overcome these challenges by integrating manual data logging with digital tracking and reporting tools. This hybrid approach allows for the accurate entry of data, such as nest quantity, weight, quality, and environmental conditions, while leveraging digital features like CRUD operations, secure databases, and automated reporting to streamline management processes. Additionally, the system offers scalability, real-time tracking, and accessibility across mobile devices, tablets, and computers, providing farmers with a flexible and centralized platform for operations.

By addressing the inefficiencies of traditional manual record-keeping and introducing advanced features tailored to the unique needs of the Swiftlet Bird's Nest industry, the system enhances operational efficiency, traceability, and decision-making. Its scalable and mobile-responsive design positions it as a reliable and sustainable solution, with the potential to revolutionize the swiftlet farming sector while contributing to the industry's long-term development.

Chapter 3: Requirement Analysis & Design

3.1 Introduction

Chapter 3 outlines the methodology used in the development of the Swiftlet Bird's Nest Harvesting Management Tracking System. The Agile Scrum framework, chosen for its flexibility and iterative approach, facilitates continuous refinement of requirements and adaptation to feedback. This methodology organizes development tasks into manageable sprints, ensuring system requirements are addressed effectively while promoting collaboration and regular stakeholder interaction. The chapter covers the analysis process, including requirement elicitation and system modelling, as well as the design process, focusing on architectural diagrams, algorithms, UI sketches, and database structures. This comprehensive approach lays the groundwork for the system's successful implementation.

3.2 Methodology

Figure 12: Agile Scrum Model

Agile Scrum methodology, referring to Figure 12, consists of six primary phases: Requirements, Design, Development, Testing, Deployment, and Review. In this chapter, we

focus on the Requirements and Design phases, which are crucial for defining the system's functionality and creating the necessary design artifacts.

Table 2: Summary of Agile Scrum Methodology

Phase	Activity	Technique	Tool
Requirements	Collect requirements from stakeholders.	Google Forms	Google Forms
Planning	Prioritize features and define sprint goals.	Sprint Planning, User Story Mapping	Scrum Board (e.g., Trello, Jira)
Design	Create system workflows and models.	UML Diagrams	Lucidchart, Visual Paradigm
Development	Build features incrementally in sprints.	Incremental Development, Coding Practices	Visual Studio Code, GitHub
Testing	Verify and validate functionality.	Unit Testing, Integration Testing	XAMPP Local Server
Deployment	Deploy the system to a cloud server for stakeholders.	Continuous Integration/Delivery (CI/CD)	000webhost
Review	Gather feedback	Sprint Reviews,	Google Meet,

	from stakeholders to improve the system.	Retrospectives	Feedback Forms

3.3 Elicitation and Analysis

3.3.1 Stakeholder Identification

The stakeholder interviewed for this project is a swiftlet birdhouse owner managing two medium-scale birdhouses, which yield approximately 150 nests per harvesting cycle. The stakeholder's name is Mark Arthur whom has over 14 years of experience in the industry and possesses extensive knowledge of swiftlet farming operations, pest management, and data tracking requirements. The stakeholder is responsible for monitoring environmental conditions, organizing harvest schedules, and overseeing periodic maintenance of the birdhouses, making them highly relevant for providing insights into practical system requirements. The photos of the stakeholder's birdhouse can be viewed in Appendix B.

3.3.2 Elicitation Method

To gather detailed requirements for the Swiftlet Bird's Nest Harvesting Management Tracking System, a structured interview was conducted with the stakeholder. This method was chosen over general surveys to obtain qualitative insights from an experienced swiftlet birdhouse owner, ensuring that the system is designed based on real industry needs rather than generic assumptions. A structured interview provides the flexibility to explore specific challenges and potential solutions while maintaining a focused approach to requirements gathering. The interview questions were designed to gather insights into three critical aspects of swiftlet bird

nest farming operations: operational challenges, desired features, and usability needs. The first focus area, operational challenges, aimed to identify the difficulties faced by swiftlet birdhouse owners in structuring and retrieving nest-related data. While manual tracking remains a common practice, the lack of a centralized system leads to disorganized records, difficulties in tracking nest conditions across multiple harvesting cycles, and missed scheduling for nest collection. Additionally, pest control and environmental monitoring were highlighted as recurring concerns that require better management solutions.

The second focus area, desired features, explored essential functionalities that the farmer expects from the system. The stakeholder emphasized the importance of CRUD capabilities for managing nest data efficiently, including logging nest positions, conditions, weight, and quality. Furthermore, the ability to analyze historical nest production data was requested to help swiftlet birdhouse owners make informed decisions regarding harvesting trends and farm productivity.

The final area, usability needs, assessed the stakeholder's preferences regarding system design and accessibility. The farmer expressed the need for a mobile-friendly interface that allows easy logging and retrieval of data during field visits. Since swiftlet farmers do not enter birdhouses daily, the system must be designed to enable quick updates whenever necessary without requiring frequent manual interventions. The interface should be user-friendly and intuitive, ensuring that farmers with limited technical knowledge can use the system effectively without requiring extensive training.

Sample Interview Questions:

- "What are the main challenges you face in structuring and retrieving bird nest data?"
- "What additional features or functionality would you like in the system?"

- “How do you currently keep track of nest conditions (egg presence, newly built, ready for harvest)?”
- “How often do you visit your birdhouses, and what activities do you usually perform (harvesting, cleaning, maintenance)?”
- “Would a mobile-friendly web-based system improve data entry and accessibility for you?”

The full list of interview questions is provided in Appendix A.

3.3.3 Interview Findings

The structured interview was conducted with a swiftlet birdhouse owner who also served as the primary stakeholder for this project. With over 14 years of experience managing multiple birdhouses, the stakeholder provided extensive insight into real-world operational challenges, particularly those related to manual nest logging, harvest planning, and environmental control. Due to limited access to birdhouse owners during the study period, only one participant was formally interviewed. However, this was supplemented by the author’s direct involvement in the stakeholder’s operations, including multiple visits to active birdhouses. During these visits, the author participated in routine tasks such as inspecting ceiling nest areas, changing humidifier water, and using printed log sheets to record nest harvesting data. This first-hand exposure provided valuable observational context that reinforced and validated the interview findings. Supporting photos of these on-site activities are included in Appendix C.

The stakeholder highlighted several inefficiencies in the current approach to nest management, such as unstructured paper-based logs, poor accessibility to historical records, and the absence of a centralized system for organizing nest harvesting data. Although the birdhouse

owner does not enter the birdhouse daily, there is a clear need for a structured system to periodically log harvest-ready nests and retrieve past records to plan schedules more effectively. At present, nest statuses are tracked manually using printed grid layouts, which are updated by hand to indicate nest readiness.

Regarding environmental monitoring, the stakeholder emphasized the importance of maintaining suitable conditions, particularly temperature and humidity, for optimal nest development and occupancy. While the stakeholder considered full IoT automation unnecessary due to its high cost and the intermittent nature of birdhouse activity, they acknowledged that timely notifications about environmental issues would be valuable. As a direct response, the system was enhanced to include a real-time environmental alert feature. This module, powered by an ESP32 microcontroller paired with a DHT22 sensor, collects temperature and humidity readings and triggers alerts when values fall outside optimal ranges. This implementation was guided by published data on swiftlet environmental requirements, as further discussed in Chapter 2.

The stakeholder also requested that the system include robust data handling capabilities, such as full CRUD (Create, Read, Update, Delete) operations for nest records, task input for harvest-related scheduling, and a mobile-friendly interface for in-field data logging. Another critical gap identified was the lack of analytics tools in existing manual workflows, which makes it difficult to observe long-term harvesting trends.

Although these insights were obtained from a single stakeholder, they were reinforced by direct field observations and technical implementation experience. The combination of stakeholder feedback and real-world validation provided a strong foundation for developing a

system that addresses both practical and technical needs. Future work is expected to include broader user engagement and formal observational studies to strengthen generalizability.

3.3.4 Elicitation Analysis

The findings from the elicitation process provided valuable insights into the key requirements for the Swiftlet Bird's Nest Harvesting Management Tracking System. The stakeholder's feedback highlighted critical operational challenges, such as inefficiencies in manual data tracking, the need for timely notifications, and difficulties in analyzing historical production data. These challenges formed the foundation for identifying both functional and non-functional requirements for the system.

The stakeholder's emphasis on CRUD (Create, Read, Update, Delete) functionalities confirmed the necessity of incorporating robust data management capabilities. CRUD operations will enable efficient handling of data logs, minimizing errors and ensuring accurate record-keeping. Additionally, the request for real-time notifications and alerts was prioritized to address scheduling inefficiencies and support timely decision-making processes. The stakeholder's preference for a user-friendly interface reinforced the need for simplicity in design, ensuring accessibility for users with minimal technical expertise. Mobile compatibility was identified as a critical feature to allow convenient data logging and access during fieldwork.

Based on this analysis, the elicitation results align closely with the system's objectives. The focus on operational efficiency and usability will drive the development of core features, including a centralized database for real-time data management, an intuitive user interface, and notification functionalities. These requirements are expected to address the stakeholder's pain points effectively while laying a scalable foundation for future system enhancements.

3.4 Software Requirement

The software requirements for the Swiftlet Bird's Nest Harvesting Management Tracking System include essential tools to streamline development, database management, and version control. These tools ensure the system is robust and efficient, addressing the needs for functionality such as data tracking and reporting. The required software is summarized in the following table.

Table 3: Table of Software Requirements

Software Requirement	Description
Visual Studio Code (VS Code)	Visual Studio Code is a lightweight and highly extensible integrated development environment (IDE) that will be used for coding and managing the project's source files. Its robust features, such as syntax highlighting, debugging tools, and support for extensions, make it ideal for developing web-based applications. Additionally, VS Code offers seamless integration with Git for version control, allowing efficient collaboration and code management.
Navicat	Navicat is a powerful database management tool that facilitates the design, querying, and administration of databases. It will be used to create and manage the Entity-Relationship Diagram (ERD) and to interact with the system's database, which is implemented using MySQL. Navicat's user-friendly interface and advanced features, such as data synchronization and SQL generation, streamline database development.

GitHub	GitHub is an online platform for version control and collaborative development. It will serve as a repository for the system's source code, enabling version tracking and collaborative coding among team members. By hosting the project's code on GitHub, developers can efficiently manage changes, maintain a history of modifications, and ensure the system's codebase is accessible and secure.
MySQL	MySQL is a relational database management system chosen to store and manage the project's data, including information about farmers, nests, expenses, and reports. It offers reliable and scalable database solutions, ensuring efficient storage and retrieval of data. MySQL's compatibility with Navicat and other development tools makes it a robust choice for this project.
Web Browser	A modern web browser, such as Google Chrome or Mozilla Firefox, is essential for testing and validating the web-based functionalities of the system. The browser will allow developers to simulate the end-user experience, test user interfaces, and identify and debug issues in real-time.
XAMPP	XAMPP is an open-source cross-platform server stack used for local development and testing. It provides a pre-configured environment that includes Apache, PHP, and MySQL, making it easy to host and test the system on a local server. XAMPP is critical during development to ensure the system functions

	correctly before deployment.

3.5 Hardware Requirements

The Swiftlet Bird's Nest Harvesting Management Tracking System requires a range of hardware components to support its functionalities effectively. These hardware elements are selected based on their ability to handle manual data logging, real-time updates, and efficient system management. Below is a table for the overview of the key hardware requirements.

Table 4: Table of Hardware Requirements

Hardware Component	Description	Specifications
Mobile Devices or Tablets	Portable devices used for logging bird nest data directly into the system during fieldwork. These provide internet connectivity and a responsive touchscreen for ease of data entry.	- Dual-core processor - 2 GB RAM - 16 GB storage - Internet connectivity
Laptops, Desktop Computers	Used by administrators to manage records, generate reports, and access dashboards. These devices provide the computational	- Quad-core processor - 8 GB RAM - 256 GB storage - High-resolution screen

	power needed for data analysis and handling.	
Wi-Fi Router or Access Points	Enables seamless data transfer between mobile devices, tablets, laptops, and the centralized server. Provides connectivity for real-time updates.	- High-speed router - Reliable network signal coverage
Internet Connection	Ensures smooth communication between user devices and the centralized server, especially for real-time updates and remote access.	- Minimum speed: 10 Mbps

The system is currently being extended with an environmental sensor module consisting of an ESP32 microcontroller and a DHT22 sensor. This enhancement will allow users to automatically monitor temperature and humidity within the birdhouse, reducing manual input and enabling better environmental control.

3.6 System Design

3.6.1 System Architecture Diagram

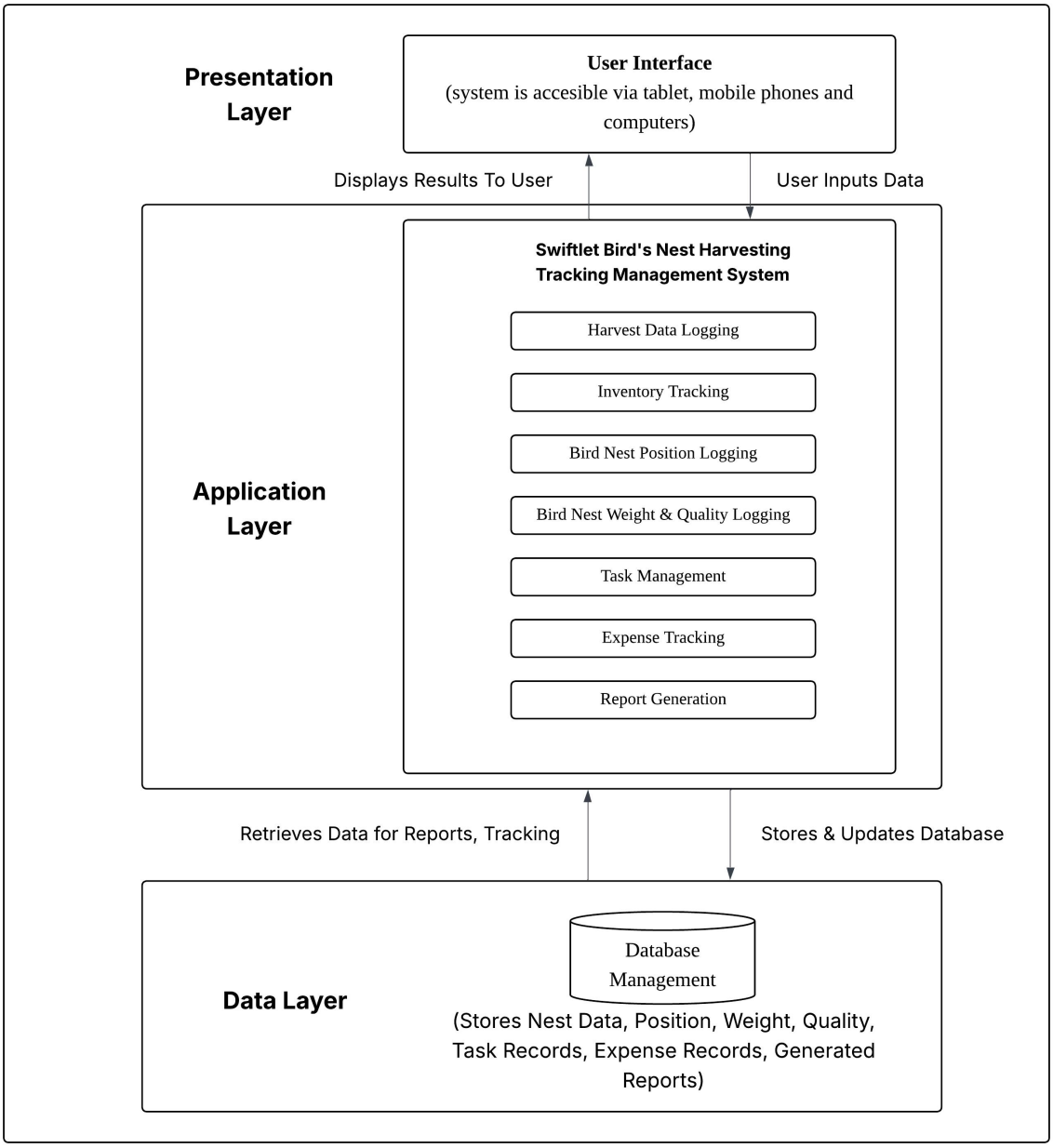


Figure 13: System Architecture Diagram

Based on Figure 13, The Swiftlet Bird's Nest Harvesting Management Tracking System follows a three-tier architecture model, which enhances system scalability, security, and efficiency in managing swiftlet farming operations. The architecture consists of the Presentation Layer, Application Layer, and Data Layer, each responsible for different aspects of system functionality.

The Presentation Layer serves as the user interface, allowing swiftlet farmers to interact with the system using tablets, mobile phones, or computers. This layer facilitates data entry and retrieval, enabling users to input key information such as nest position, weight, and quality. The system then processes this data and provides feedback, such as generating reports and displaying inventory updates. The bidirectional interaction between the Presentation Layer and the Application Layer ensures that users can effectively manage their farm records in real time.

The Application Layer is the core of the system, handling business logic and processing user requests. It includes several key functionalities such as Harvest Data Logging, Inventory Tracking, Bird Nest Position Logging, Bird Nest Weight & Quality Logging, Task Management, Expense Tracking, and Report Generation. Each of these modules ensures that farmers can maintain accurate records of nest production, organize farm-related tasks, track financial expenses, and analyze performance trends. This layer acts as the middleware, processing user inputs, executing CRUD (Create, Read, Update, Delete) operations, and ensuring data integrity before passing information to the database.

The Data Layer is responsible for securely storing and managing all system records. The Database Management System (DBMS) maintains information related to nest production, positional data, weight and quality attributes, task logs, expense records, and generated reports. The database is designed to ensure fast retrieval and secure storage, supporting the overall efficiency of the system. The Application Layer interacts with the Data Layer bidirectionally, meaning it can store and update database records while also retrieving stored data to generate meaningful insights.

The arrows in the diagram illustrate the flow of data between these three layers. When a farmer logs data through the Presentation Layer, the request is processed by the Application

Layer, which then stores it in the Data Layer. Similarly, when users need to retrieve information, such as generating reports, the Application Layer queries the database, retrieves relevant records, and presents them back to the User Interface.

This layered architecture ensures system scalability, maintainability, and security. The modular approach allows for future system enhancements, such as integrating automated data collection tools or expanding reporting features. Additionally, the clear separation of concerns between layers ensures that system performance remains optimized, with dedicated components handling specific tasks.

In conclusion, the System Architecture Diagram effectively illustrates the logical structure and data flow within the Swiftlet Bird's Nest Harvesting Tracking Management System. The three-tier model enhances operational efficiency, allowing farmers to log, track, and analyze farm production while ensuring secure and structured data management.

3.6.2 Justification for Manual Logging

While the automation of data logging (such as IoT-based tracking) was considered, the stakeholder confirmed that it is currently not feasible due to high implementation costs and the nature of swiftlet farming, where farmers do not enter birdhouses daily. Unlike other livestock farming systems, swiftlet farmers only access birdhouses periodically for nest harvesting or cleaning, making fully automated tracking unnecessary. Manual logging remains the most practical approach as it aligns with current farming routines where farmers already inspect nests visually to check for egg presence, newly built nests, or readiness for collection. Implementing automation would not only be costly but also redundant, as human verification remains essential for determining nest conditions accurately.

Thus, a well-structured manual logging system with CRUD functionalities, mobile compatibility, and real-time notifications is the most effective solution. This approach enhances data tracking efficiency without disrupting existing farming operations while ensuring accuracy and accessibility for farmers. The system optimizes current practices instead of attempting to replace them with unnecessary automation， ensuring feasibility and scalability in real-world swiftlet farming conditions.

3.6.3 Use Case Diagram

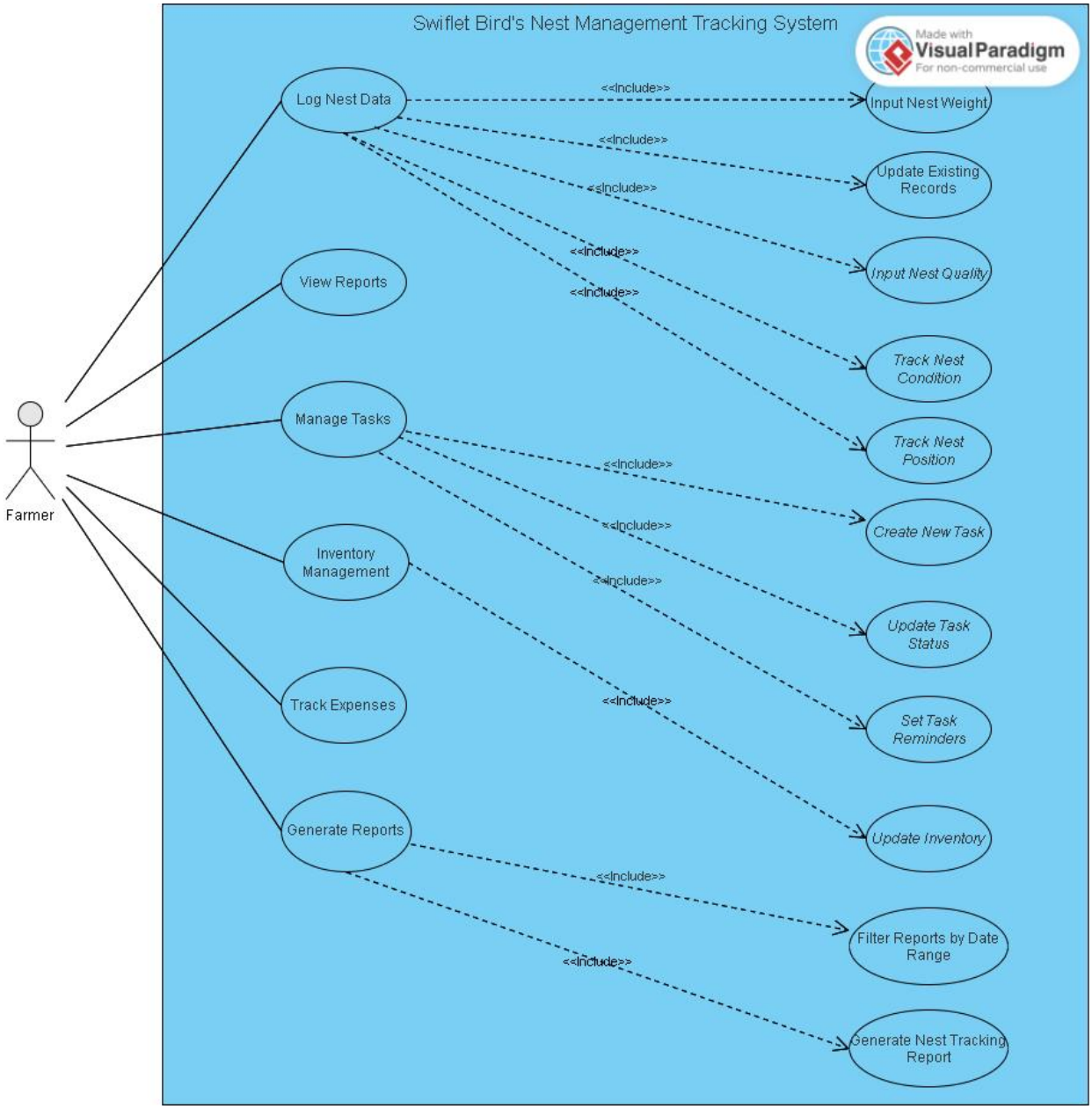


Figure 14: System Use Case Diagram

The Use Case Diagram, referring to Figure 16, for the Swiftlet Bird's Nest Management Tracking System illustrates the interactions between the Swiftlet Farmer (User) and the system's core functionalities. This diagram captures the essential use cases related to nest tracking, inventory management, reporting, task management, and financial tracking. The Swiftlet Farmer

is the primary actor, responsible for inputting data, managing tasks, updating records, and generating reports.

The Log Nest Data use case is central to the system, allowing farmers to log essential information about the nests, such as weight, quality, condition, and position. This use case includes specific functionalities like inputting nest weight and quality, tracking nest condition (e.g., egg present, newly built, or ready for collection), and recording nest position within the birdhouse. Additionally, the Update Existing Records function allows users to modify previously entered nest data if necessary.

Another key component is the Inventory Management use case, which ensures that farmers can update and track harvested nests. The Update Inventory function enables farmers to reflect changes in stock, preventing mismanagement of collected nests.

The Generate Reports use case enables farmers to extract valuable insights regarding nest production and trends. This includes generating Nest Tracking Reports, which summarize nest conditions and positions over time, as well as filtering reports by date range for more refined analysis. The ability to view reports ensures that farmers can monitor productivity and optimize harvesting decisions.

For improved organization, the system also includes a Manage Tasks function, allowing farmers to create new tasks, update task status, and set reminders for essential farming activities. This ensures that farmers stay on schedule with nest monitoring, maintenance, and harvesting.

Additionally, the Track Expenses use case allows farmers to log and track financial transactions related to farming operations, such as maintenance costs, labor wages, and

equipment purchases. This feature helps farmers maintain financial transparency and monitor profitability.

The Use Case Diagram effectively outlines how the system supports structured nest management, inventory tracking, reporting, task scheduling, and financial management, providing farmers with a comprehensive tool to streamline operations and improve efficiency in swiftlet farming. The Use Case Table Description is below.

Table 5: Use Case Table Description

Use Case	**Description**	**Actor**	**Pre-conditions**	**Steps**	**Post-conditions**	**Exceptions**
Log Nest Data	Farmers log essential nest details, including weight, quality, condition, and position.	Swiftlet Farmer	User is logged into the system.	1. Navigate to "Log Nest Data". 2. Input nest details. 3. Confirm submission.	Data is stored in the system.	Invalid inputs prompt an error message.
Generate Reports	Farmers generate reports on nest data and production trends.	Swiftlet Farmer	User has logged nest data.	1. Navigate to "Generate Reports".2. Select report type.3. Confirm.	Report is displayed.	No data available.

Update Inventory	Farmers update harvested nest inventory.	Swiftlet Farmer	User has collected nests.	1. Navigate to "Inventory Management". 2. Select "Update Inventory".3. Confirm.	Inventory data is updated.	No collected nests to update.
Track Expenses	Farmers log expenses related to farm operations.	Swiftlet Farmer	User is logged in.	1. Navigate to "Track Expenses".2. Enter transaction details.3. Confirm submission.	Expense data is stored.	Invalid inputs prompt an error.
Manage Tasks	Farmers create, update, and track task activities.	Swiftlet Farmer	User is logged in.	1. Navigate to "Manage Tasks".2. Select task action (Create,	Task is saved or updat	Invalid task inputs prompt an error.

				Update, Reminder).3. Confirm.	ed.	

3.6.4 Sequence Diagram

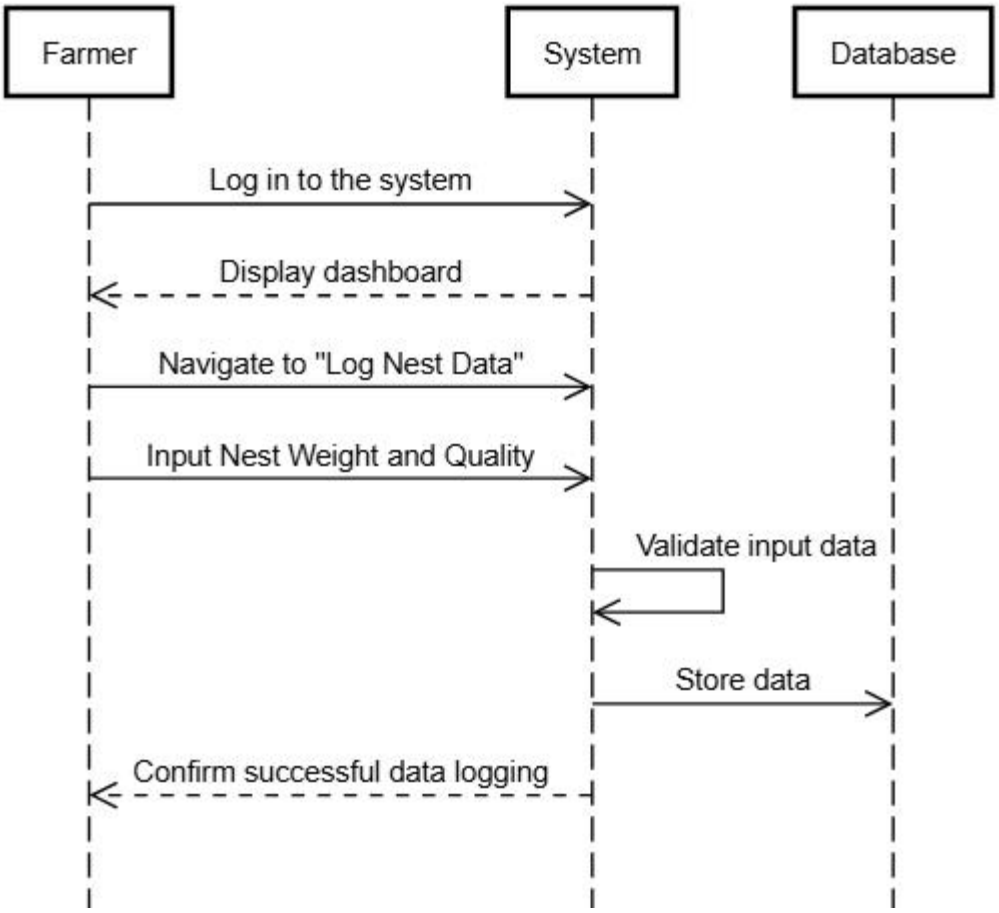


Figure 15: Nest Logging Sequence Diagram

Figure 17 illustrates how the farmer interacts with the system to record details about bird nests. The process begins with the farmer logging into the system and navigating to the "Log

Nest Data" module. The farmer inputs data such as nest weight and quality, which the system validates for accuracy. Once validated, the data is stored in the database, and the system confirms successful logging to the farmer, ensuring proper data entry for future use.

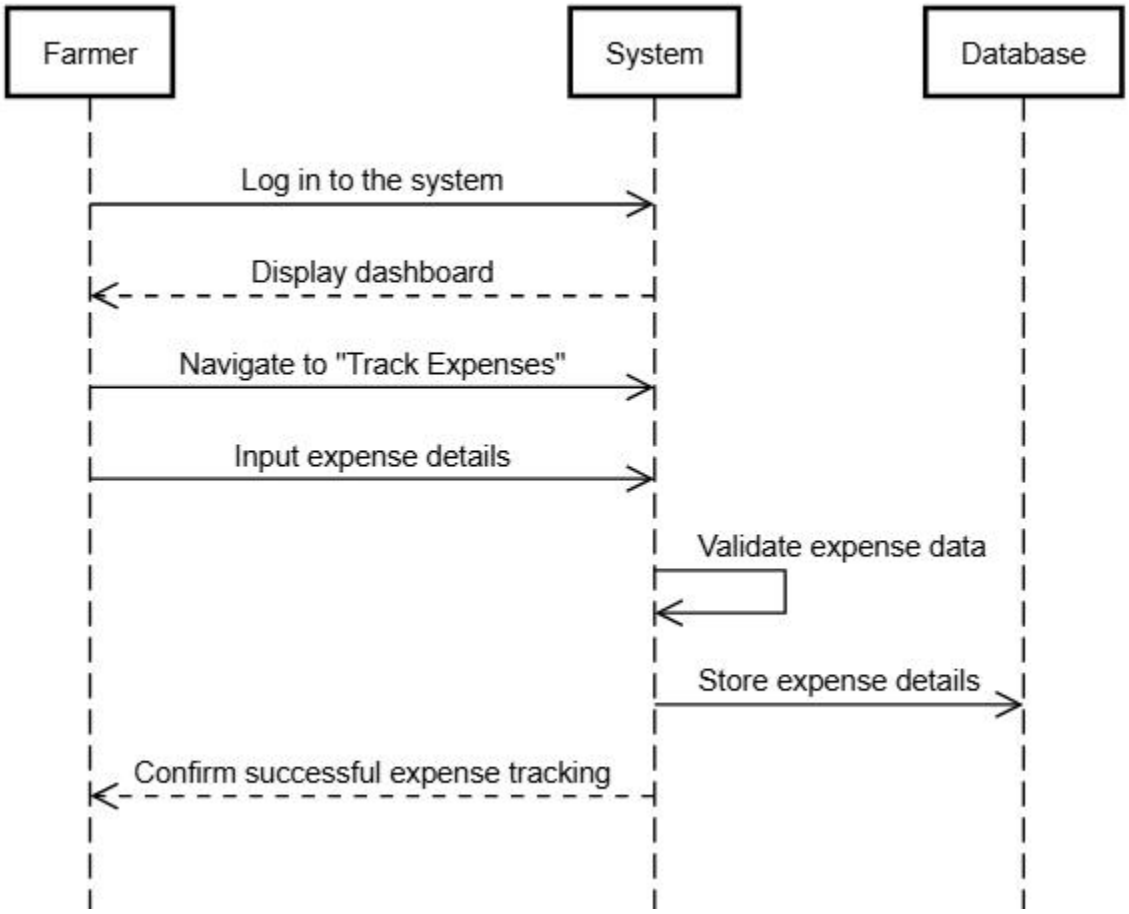


Figure 16: Expense Tracking Sequence Diagram

Figure 18 depicts the steps involved in recording operational expenses. The farmer logs into the system and navigates to the "Track Expenses" module. After entering expense details like description, amount, and date, the system validates the data to ensure it meets predefined

criteria. Upon successful validation, the system stores the expense details in the database and provides confirmation to the farmer, enabling accurate tracking of operational costs.

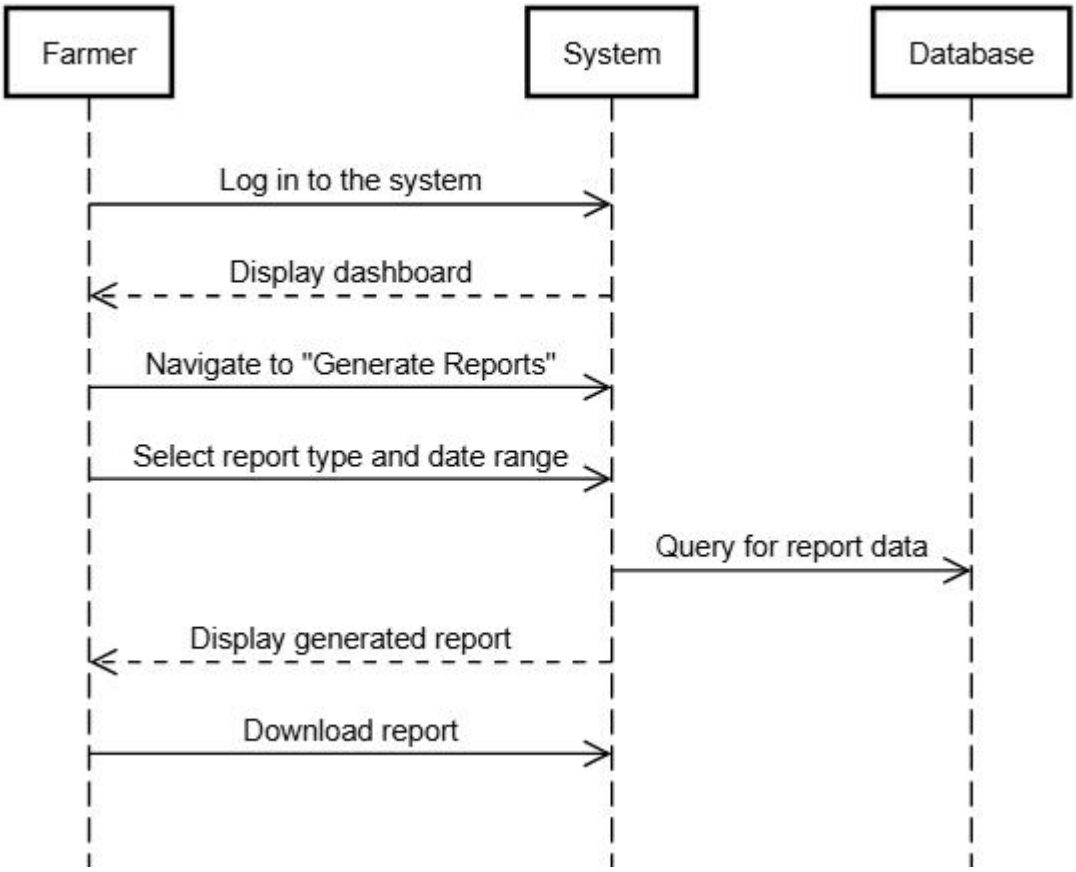


Figure 17: Generate Report Sequence Diagram

Figure 19 outlines the process of generating reports for analysis and decision-making. The farmer logs into the system and navigates to the "Generate Reports" module, where they specify the type of report and select a date range for filtering. The system retrieves the relevant data from the database, processes it, and generates the requested report. The generated report is displayed to the farmer, who has the option to download it for offline use or further analysis.

3.7 Database Design

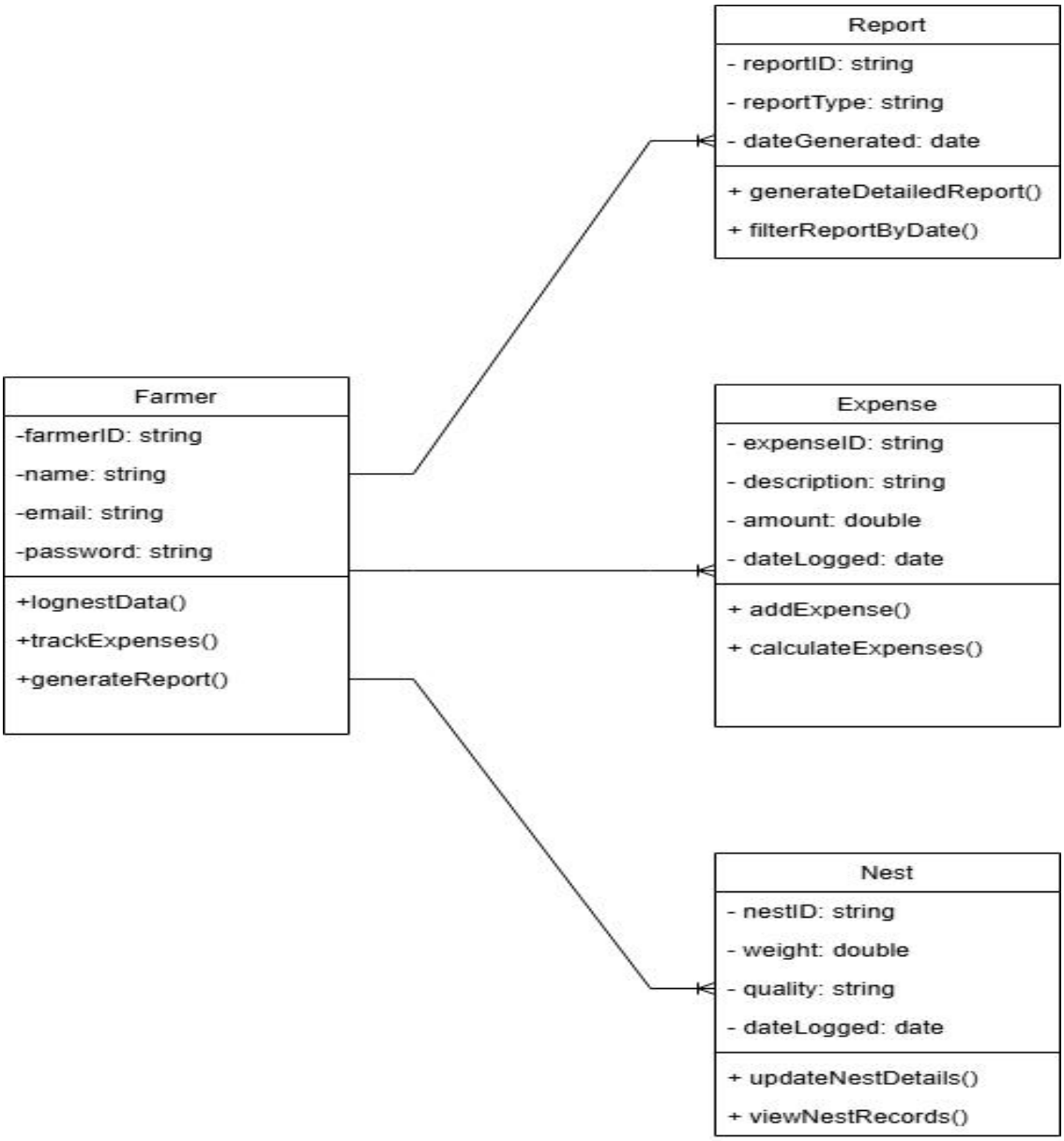


Figure 18: Class Diagram

Based on Figure 20, the “Farmer” class represents the primary user of the system, containing essential attributes such as farmerID, name, email, and password. These attributes are stored securely and allow each farmer to log in and manage their activities within the system.

The “Farmer” class includes methods like logNestData(), which allows farmers to record details about bird nests, trackExpenses(), which lets them monitor operational costs, and generateReport(), which automates the creation of insightful reports for better farm management.

The “Nest” class handles data related to swiftlet nests, with attributes such as nestID, weight, quality, and dateLogged. This class is directly associated with the Farmer class, as a farmer may log multiple nest entries. The “Nest class” includes methods like updateNestDetails() for editing logged information and viewNestRecords() for reviewing recorded nest data. This ensures farmers can efficiently track and manage swiftlet nest information over time.

The “Expense” class is responsible for managing financial records, with attributes like expenseID, description, amount, and dateLogged. It is associated with the “Farmer” class, allowing farmers to track multiple expenses tied to their operations. The “Expense” class provides methods such as addExpense(), which lets users log new expenses, and calculateExpenses(), which summarizes and analyzes the recorded data to aid financial management.

The “Report” class generates detailed reports for the farmer, containing attributes such as reportID, reportType, and dateGenerated. Reports can be customized to provide insights into nest productivity, operational expenses, and other relevant metrics. Methods like generateDetailedReport() enable the creation of comprehensive summaries, while filterReportByDate() allows farmers to narrow down report data for specific timeframes. This ensures that farmers have access to actionable insights for better decision-making.

The relationships between these classes reflect the operational flow of the system. The “Farmer” class serves as the central entity, connecting to the “Nest”, “Expense”, and “Report”

classes through one-to-many associations. A single farmer can log multiple nests, track numerous expenses, and generate multiple reports. These relationships ensure that the system is scalable, efficient, and tailored to the specific needs of swiftlet farmers.

3.8 User Interface Design

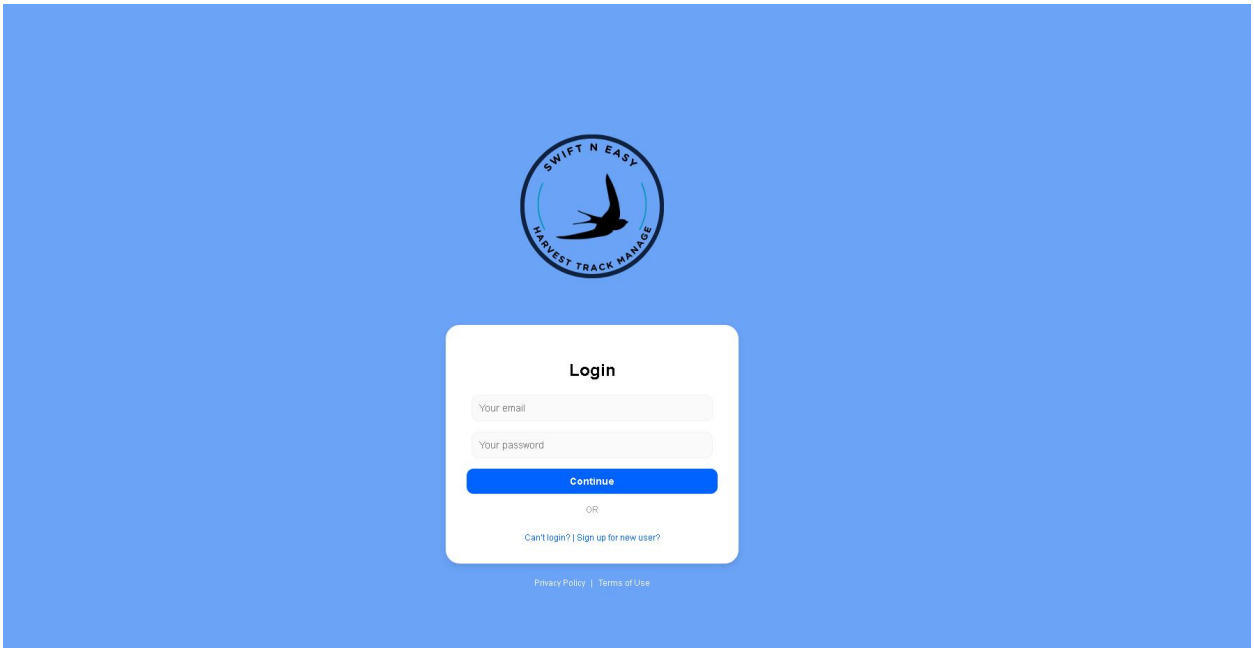


Figure 19: Login Page

The login page of Swiftlet Bird's Nest Harvesting Management Tracking System, referring to Figure 21, serves as the entry point for users to securely access the platform. It features a clean and professional design with a blue background, creating a visually appealing and calming user experience. At the top of the page is the system logo, prominently displaying the tagline “Swift N Easy: Harvest, Track, Manage,” which encapsulates the system’s core functionalities. The login interface includes input fields for users to enter their email and password, along with a primary “Continue” button for authentication. Additional links are provided for users who may encounter login issues or wish to register as new users, ensuring

inclusivity and ease of access. At the bottom of the page, links to the Privacy Policy and Terms of Use are displayed, emphasizing transparency and adherence to regulatory requirements. This user-friendly layout is designed to accommodate both technical and non-technical users, ensuring a seamless onboarding process.

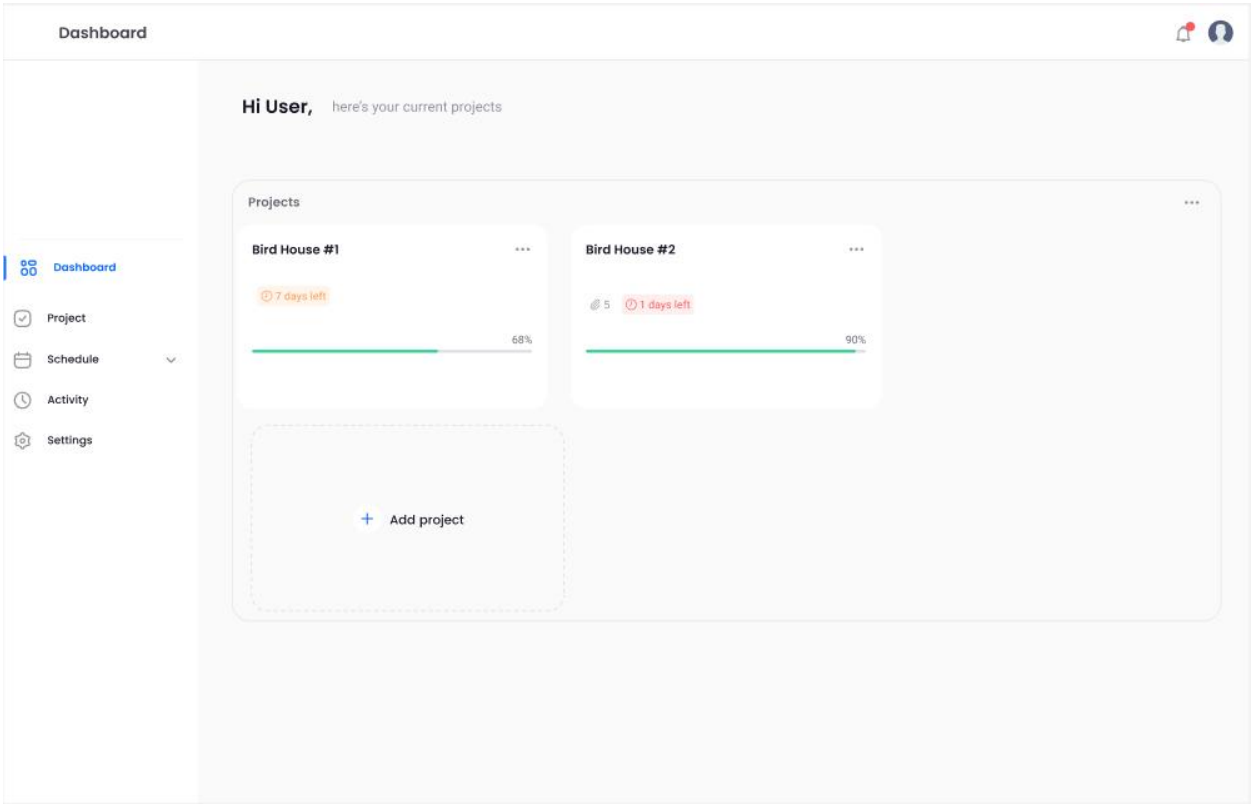


Figure 20: Dashboard

The user dashboard of the Swiftlet Bird's Nest Harvesting Management Tracking System, shown in Figure 22, provides a comprehensive overview of the system's key functionalities in a user-friendly interface. At the top, the dashboard welcomes the user and displays the current active projects. Each project is represented by a card that includes important details such as the project name, progress status, and a countdown indicating the number of days remaining for the

task or harvest schedule. Progress bars offer a visual representation of project completion percentages, aiding in quick assessments of ongoing tasks.

The left-hand navigation menu includes options such as Dashboard, Project, Schedule, Activity, and Settings, ensuring seamless access to various system modules. This intuitive menu layout allows users to navigate efficiently between tasks like managing individual projects, reviewing schedules, tracking activities, and adjusting personal or system-wide settings. The dashboard also includes a dedicated space for adding new projects, streamlining the process of expanding or organizing additional swiftlet houses.

Overall, the dashboard is designed with simplicity and clarity in mind, ensuring that users can monitor and manage multiple projects simultaneously while accessing key system functionalities with ease.

3.9 Summary

Chapter 3 outlined the Elicitation and Analysis process, system requirements, and the Agile Scrum methodology adopted for the development of the Swiftlet Bird's Nest Harvesting Management Tracking System. This chapter detailed the system’s functional and non-functional requirements, derived from structured interviews with an experienced swiftlet farmer. The stakeholder feedback provided insights into manual tracking challenges, leading to the justification for an optimized manual logging system rather than full automation.

The system design was presented using UML diagrams, including a Use Case Diagram, which illustrated the interactions between the farmer and the system, emphasizing core functionalities such as logging nest data, tracking nest position, managing inventory, updating expenses, and generating reports. The Sequence Diagrams further detailed the system processes

step-by-step, ensuring clarity in how the system handles CRUD operations, notifications, and data retrieval.

Additionally, Data Flow Diagrams (DFD) were used to visualize the flow of data between the farmer, system processes, and the database, ensuring structured information management. The Class Diagram was developed to define the database relationships between key entities: Farmer, Nest, Inventory, Expense, and Report, ensuring efficient data management and retrieval. The findings and analysis in this chapter lay the foundation for the system implementation, ensuring that the tracking and management functionalities align with real-world farming operations while maintaining scalability and usability.

Chapter 4: Implementation

4.1 Chapter Introduction

This chapter outlines the implementation process of the Swiftlet Bird's Nest Harvesting Management Tracking System. It describes the development environment, tools, technologies, and system architecture used to build the application. The goal of this phase was to transform the system's design into a fully functioning software solution capable of supporting users in managing swiftlet birdhouse operations.

The implementation covers the integration of major system modules such as birdhouse creation, grid labeling, nest logging, expense tracking, and report generation. It also presents key decisions made during development and the constraints faced. These insights provide a comprehensive understanding of how the proposed system was constructed.

4.2 Installation and Configuration

The development tools and environments required to begin the implementation are Visual Studio Code, XAMPP (which includes Apache, MySQL, and PHP), Navicat for MySQL database management, GitHub for version control, and FreeHosting for deployment. The following section will explain the usage of these tools and environments in the development of the system.

4.2.1 Visual Studio Code

Visual Studio Code is a lightweight yet powerful source-code editor developed by Microsoft. It supports various programming languages and offers features like syntax highlighting, debugging, version control integration, and extensions. In this project, Visual

Studio Code was used as the primary code editor for writing PHP scripts, HTML, CSS, and JavaScript for both the frontend and backend development.

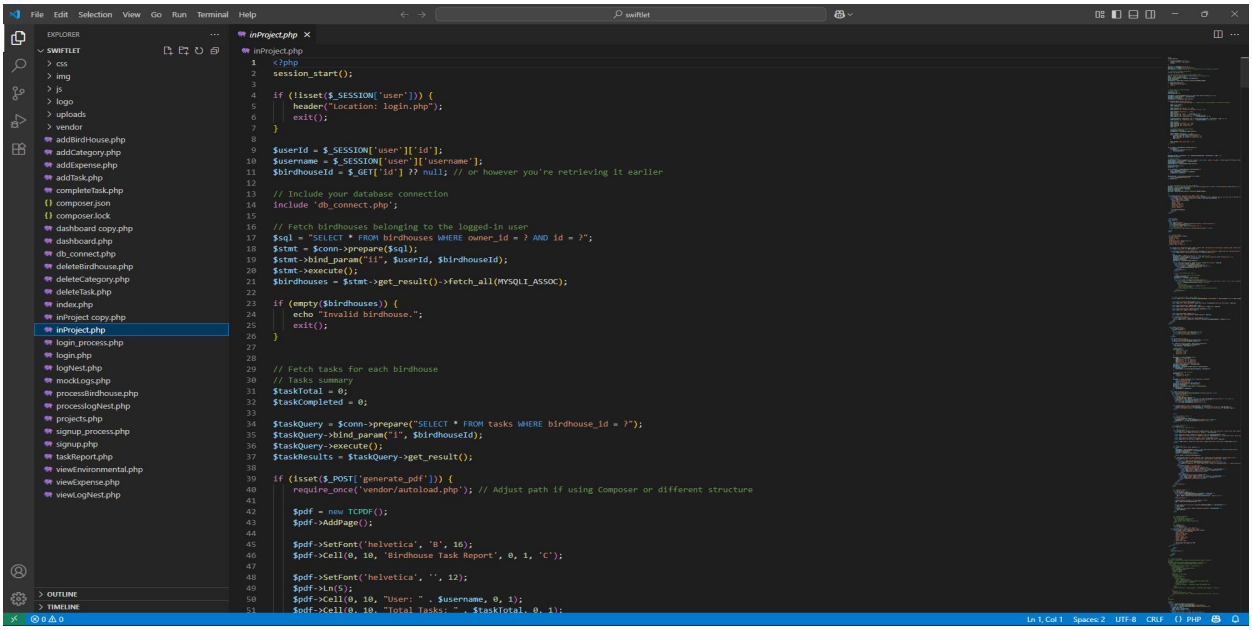


Figure 21: Visual Studio Code

4.2.2 XAMPP

XAMPP is an open-source cross-platform web server solution package developed by Apache Friends. It provides an easy-to-install environment that includes Apache HTTP Server, MySQL database, and PHP. XAMPP was used to locally host the Swiftlet Birdhouse system during development and testing phases. It allowed for running the PHP scripts and storing data in the MySQL database in an isolated local environment.

4.2.3 Navicat for MySQL

Navicat is a graphical database management and development software for MySQL and other databases. It was used to manage the Swiftlet project's MySQL database more efficiently. Navicat provided a visual interface for creating tables, executing queries, designing relationships,

and maintaining data integrity without having to write raw SQL scripts, which improved development productivity.

4.2.4 GitHub

GitHub is a cloud-based platform for version control and collaborative software development using Git. It was used to maintain and track changes in the codebase, collaborate on updates, and ensure code version integrity across development stages. GitHub also served as a backup platform for the project's source code.

4.2.5 FreeHosting

FreeHosting is a web hosting service that offers free and paid hosting plans. It was used to deploy the Swiftlet Bird's Nest Harvesting Management Tracking System to a live server environment. After completing local testing via XAMPP, the project files and database were migrated to FreeHosting, allowing users to access and test the system through the internet.

4.3 The System Interface Layout

4.3.2 Landing Page

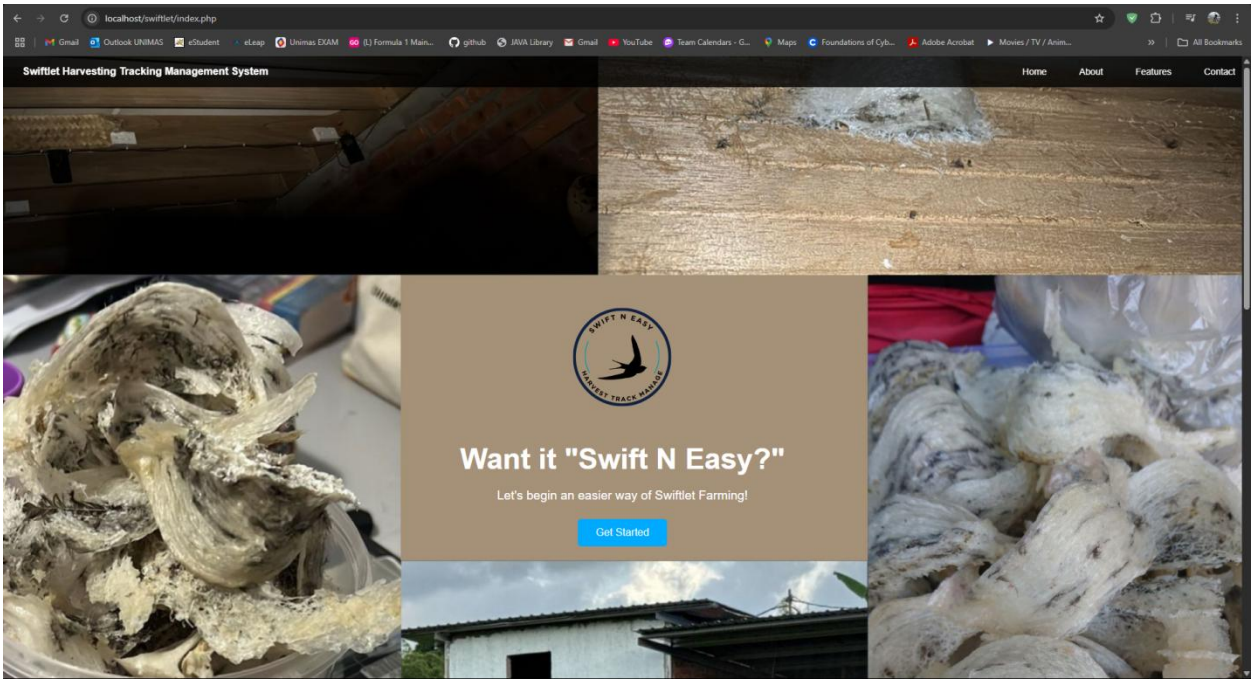


Figure 22: Landing Page

The landing page serves as the initial point of contact for users visiting the Swiftlet Harvesting Tracking Management System. It introduces the system's identity through a visually engaging layout that features high-quality background images related to swiftlet farming, along with the project's official logo and a clear callto-action.

The central message, "Want it 'Swift N Easy'?" accompanied by a prompt to "Get Started," sets the tone for the system's goal—to simplify and streamline swiftlet farm management. A navigation bar at the top right provides quick access to additional sections such as About, Features, and Contact, reinforcing standard usability practices.

This interface is designed to create an inviting and professional first impression, encouraging users to explore the system further while aligning with the project's branding and functional purpose.

4.3.3 Account Registration and Login

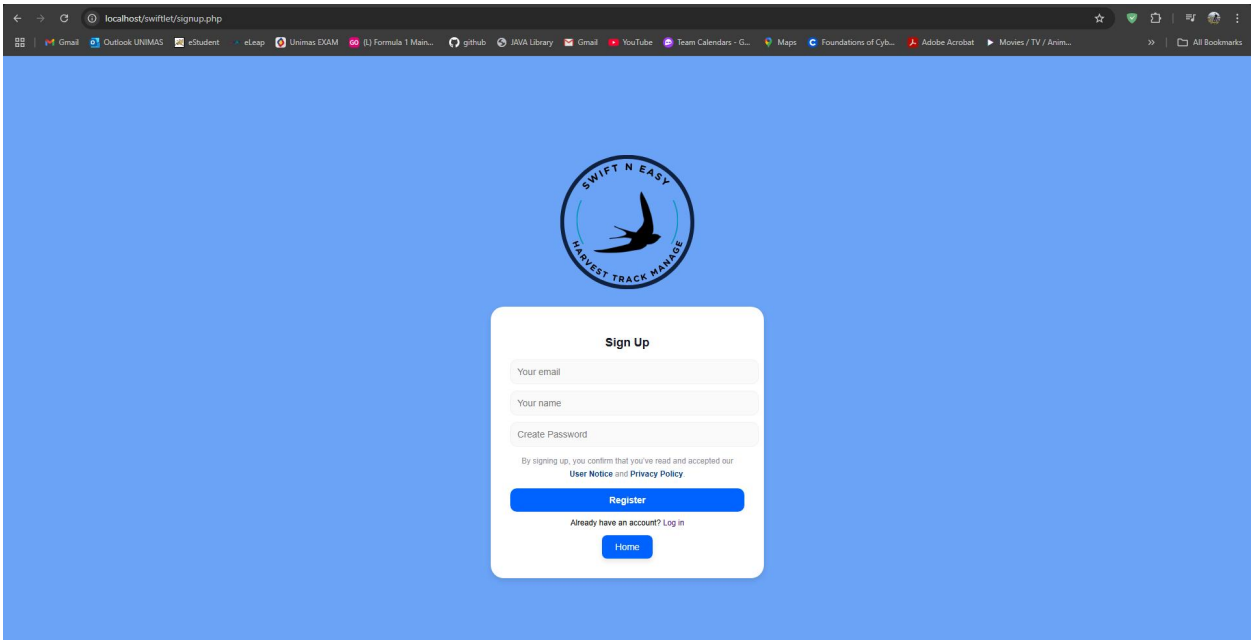


Figure 23: Account Registration Page

The user registration interface provides new users with a clean and intuitive form to create an account before accessing the system's main features. The layout is centered on the screen with a light blue background that enhances readability and maintains a consistent visual identity, reinforced by the project's logo prominently displayed above the form.

The form consists of three primary input fields: email address, username, and password. Below the inputs, a user agreement note is presented to inform users about the privacy policy and usage terms, promoting transparency and compliance. A prominent "Register" button is used to

submit the form, while secondary navigation links at the bottom offer options to log in or return to the homepage.

This interface plays a critical role in managing access control, ensuring only authorized users can proceed to use the system. The straightforward layout minimizes friction in the onboarding process, enhancing user experience and encouraging engagement with the application from the outset.

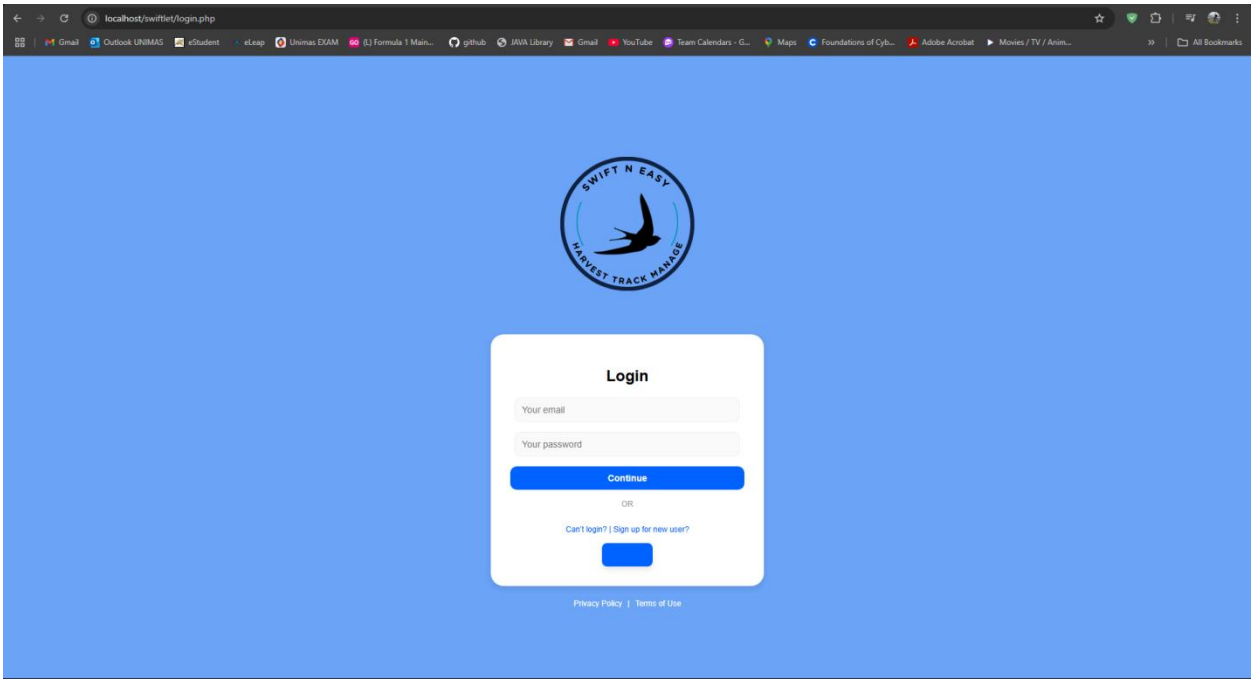


Figure 24: Login Page

The login interface enables registered users to securely access the system. The design remains consistent with the application's visual identity, featuring a centralized login panel on a calming blue background along with the official logo at the top to reinforce branding.

This interface includes two input fields—email and password—and a prominent "Continue" button for form submission. A helpful link is provided for users who may have forgotten their

credentials or who need to sign up for a new account. The inclusion of terms and privacy links at the bottom aligns with best practices for user consent and transparency.

The simplicity and clarity of this screen ensure a smooth and accessible login experience, forming a critical part of the user authentication flow while maintaining a professional aesthetic.

4.3.4 User Dashboard

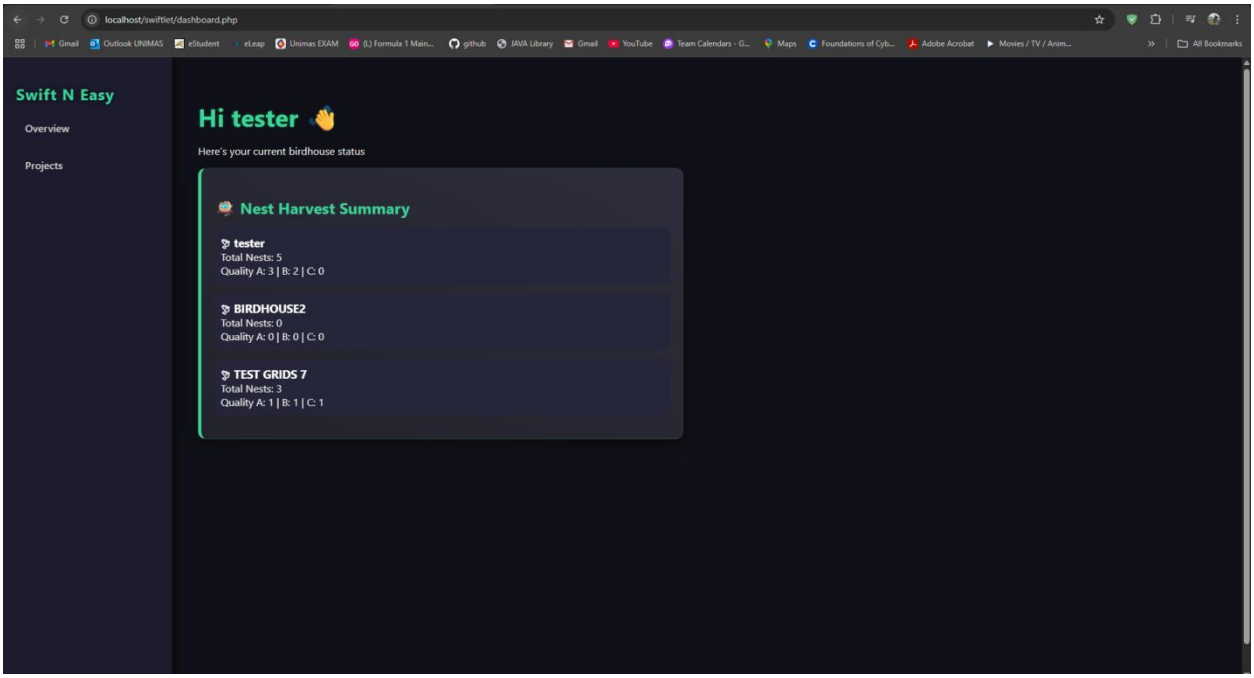


Figure 25: User Dashboard

The dashboard interface provides users with a personalized overview of their birdhouse data upon successful login. Featuring a dark-themed aesthetic for modern appeal and visual comfort, the dashboard greets the user by name and presents a summarized view of nest harvest activity across all registered birdhouses.

The main content area highlights the “Nest Harvest Summary” section, where each birdhouse entry displays the total number of nests along with their respective quality

classifications (A, B, and C grades). This compact format allows users to quickly assess performance and trends at a glance.

Navigation links such as “Overview” and “Projects” are placed on a collapsible sidebar for easy access to other modules. The layout emphasizes usability and quick data interpretation, serving as a central point for monitoring swiftlet farming operations.

4.3.5 Project Overview Page

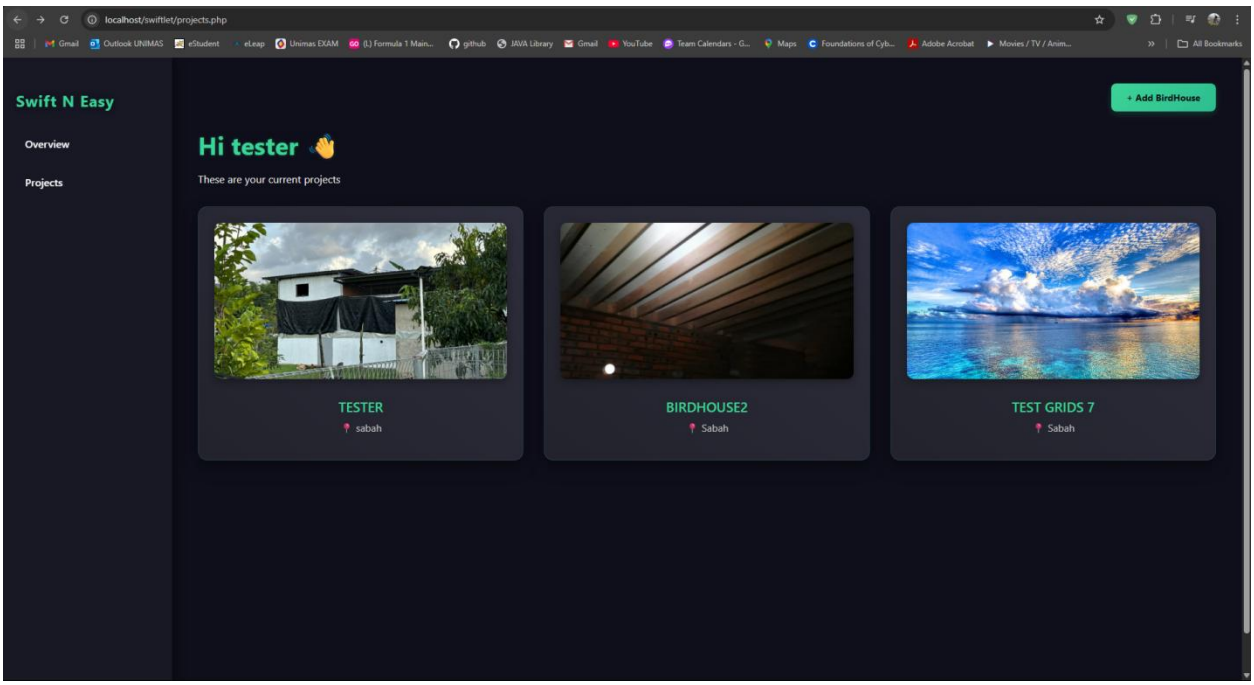


Figure 26: Project Overview Page

The project overview page presents a visual listing of all birdhouses registered by the user within the system. Each birdhouse is displayed as a card containing a cover image, project name, and location. This layout provides a clear summary of active projects and enhances navigation through visual identification.

A personalized greeting at the top reinforces user engagement, while a prominent “+ Add BirdHouse” button in the corner enables users to quickly register new birdhouses. This

streamlined design supports efficient project management by offering a user-friendly and visually organized interface.

The card-based layout reflects modern UI practices and aligns with the system's goal of simplifying the monitoring and maintenance of multiple swiftlet birdhouses.

4.3.6 Add New Birdhouse Interface

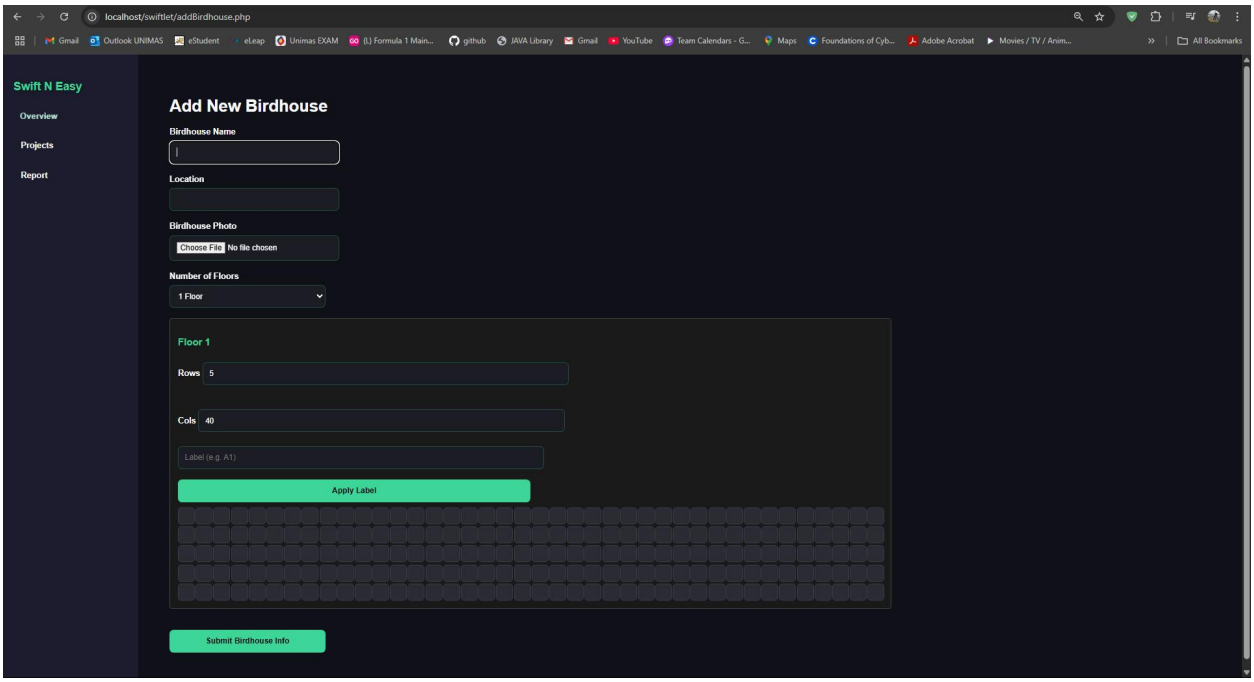


Figure 27: Add New Birdhouse Interface

The birdhouse entry interface allows users to register a new birdhouse within the system by submitting key configuration details. Users are prompted to enter the birdhouse name, its location, upload a representative photo, and specify the number of floors the birdhouse contains.

A dynamic grid configuration section appears based on the selected floor count, enabling users to define the number of rows and columns per floor. A labeling input field lets users assign custom labels (e.g., A1, B2) to specific grid positions. These labels are essential for nest tracking

accuracy in subsequent modules. The visual grid display beneath the form provides an interactive preview, enhancing usability and spatial understanding.

This interface ensures that birdhouse data is captured in a structured and customizable format, forming the foundation for all downstream tracking features in the system.

4.3.7 In-Project Overview Page

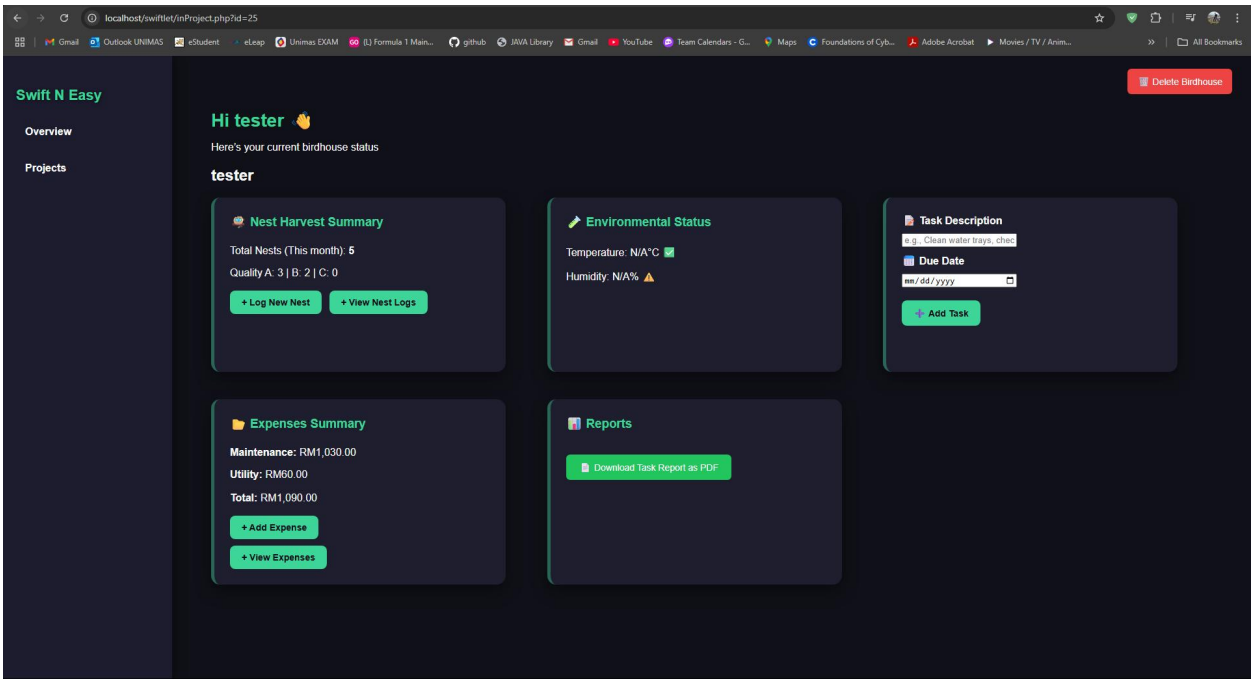


Figure 28: In-Project Overview Page

The birdhouse detail dashboard provides users with a comprehensive view of all data related to an individual birdhouse. Upon accessing this interface, users are greeted with a personalized heading that reinforces user identity and contextual awareness. The interface consolidates multiple operational components into one cohesive layout, allowing for effective monitoring and management. A section dedicated to nest harvesting displays the current month's total nest count along with their respective quality classifications, giving users a quick summary of productivity. Adjacent to it, the environmental status panel reflects current or most recently

logged temperature and humidity readings, aiding users in tracking habitat conditions. On the right, a task management form allows users to input scheduled activities or maintenance routines, complete with due dates to ensure operational upkeep. The dashboard also features an expenses summary that outlines categorized costs such as maintenance and utility, alongside the cumulative total expenditure. Additionally, a reporting panel enables users to generate downloadable reports in PDF format, enhancing traceability and documentation. With its modular layout and clean visual hierarchy, this interface efficiently brings together all relevant birdhouse data into a single accessible page, supporting users in making timely and informed decisions.

4.3.8 Nest Logging and Nest Logs Interface

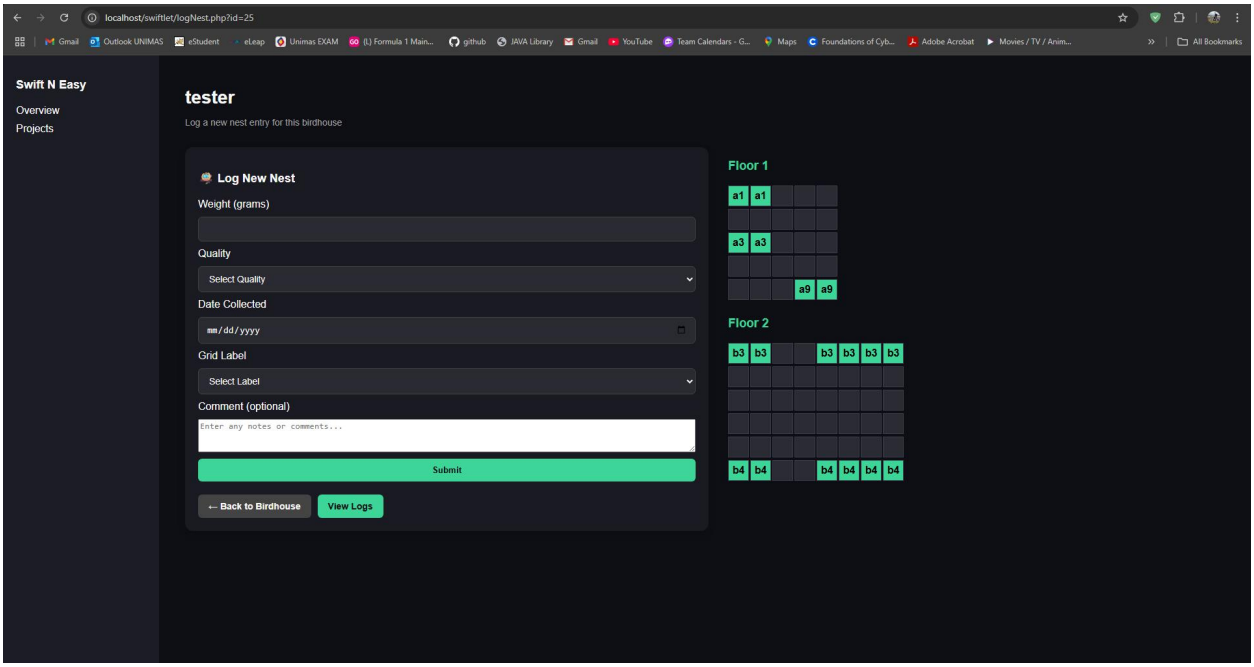


Figure 29: Nest Logging Interface

The nest logging interface enables users to record detailed information about newly harvested swiftlet nests within a specific birdhouse. Presented in a focused layout, the form captures critical data such as the nest's weight in grams, its quality rating (A, B, or C), the

collection date, and the corresponding grid label that indicates the nest's precise location within the birdhouse. An optional comment field allows users to add contextual notes or observations regarding the nest's condition or surroundings.

To assist with accurate positioning, a visual grid map is displayed on the right side of the interface, showing labeled cells across each floor. These labels correspond to previously defined grid positions, helping users quickly identify and confirm where the nest was harvested. The inclusion of a “Back to Birdhouse” button improves navigation, while the “View Logs” button offers access to historical data for review and analysis. This interface plays a crucial role in ensuring systematic record-keeping, supporting traceability, and facilitating future reporting and analysis related to nest harvesting operations.

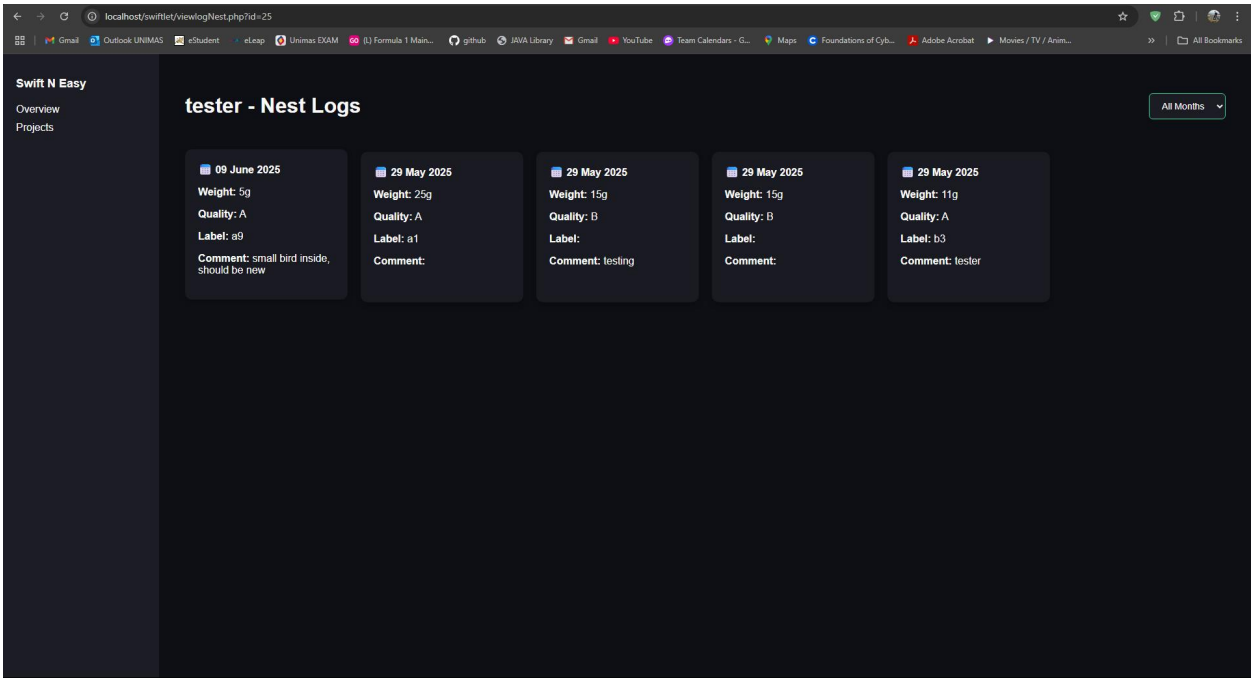


Figure 30: Nest Logs Interface

The nest logs interface displays a historical view of all recorded nest entries associated with a specific birdhouse. Presented in a card-style layout, each entry includes essential attributes

such as the collection date, nest weight, quality classification, grid label, and any comments provided during logging. This design enables users to quickly scan and review past harvest data in an organized and readable format. The cards are visually separated, improving clarity and enhancing the ability to distinguish between multiple logs at a glance. A dropdown filter at the top right corner allows users to refine the view based on specific time frames, such as monthly data, which supports more targeted analysis. This interface functions as a vital recordkeeping component, allowing users to track trends in nest harvesting, validate past entries, and evaluate changes in productivity or nest quality over time.

4.3.9 Add Expense and Expense Record Interface

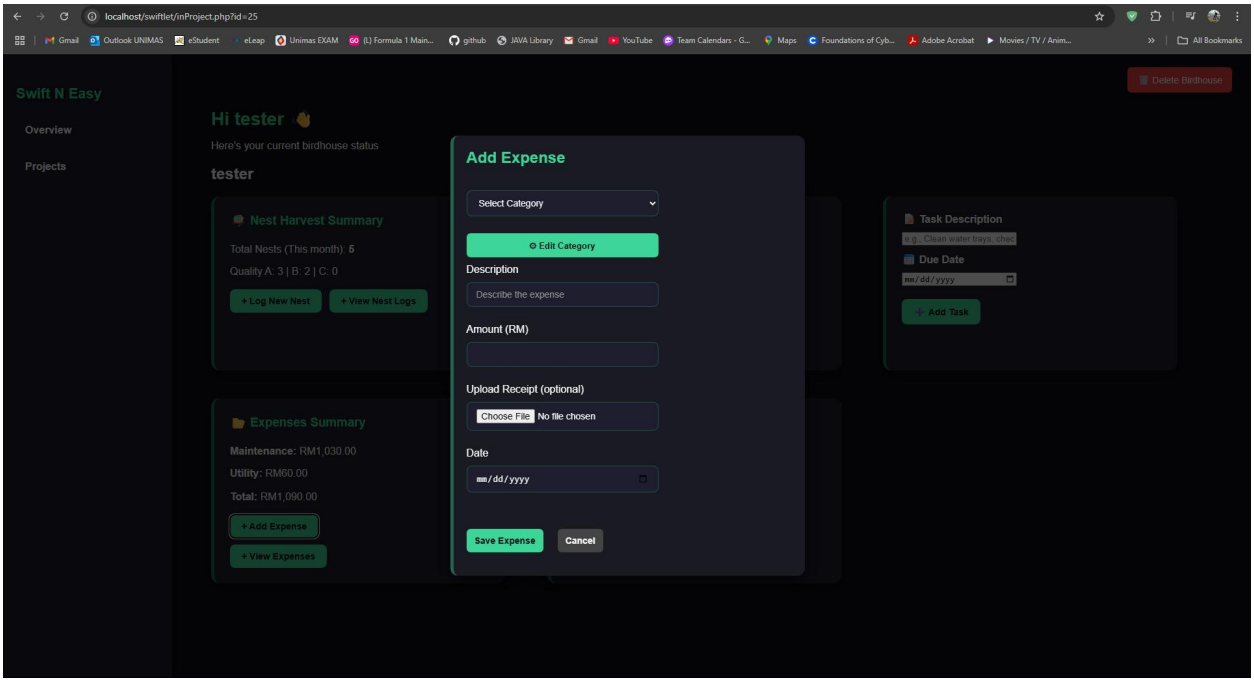


Figure 31: Add Expense Interface

The add expense interface allows users to log financial transactions related to birdhouse operations. Displayed as a modal form overlaying the main dashboard, this interface enables users to input expense details such as the category (e.g., maintenance or utility), description, amount in RM, and date of the transaction. It also supports optional receipt uploads to aid in

documentation and future auditing. A button is provided to edit or manage existing categories, allowing the system to adapt to a range of expense types specific to the user's operational needs. The layout is structured and intuitive, promoting accurate financial recordkeeping without overwhelming the user. Once submitted, the data integrates seamlessly into the system's backend, where it can later be reviewed or included in generated reports.

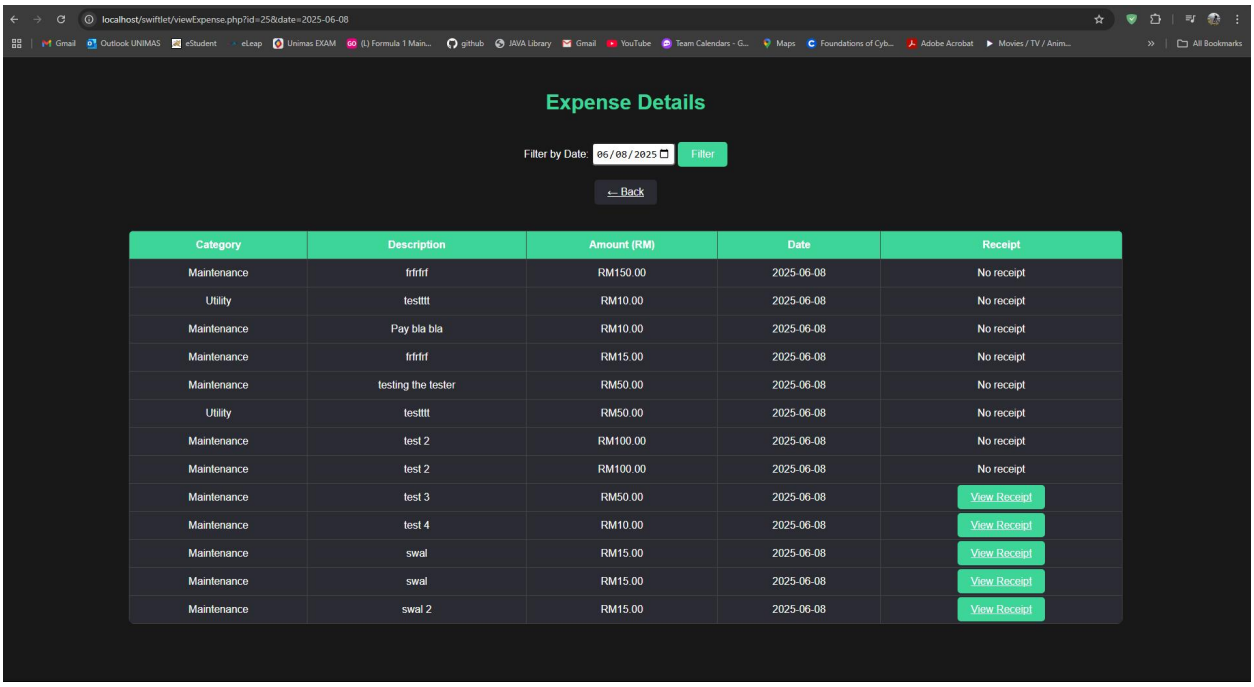


Figure 32: Expense Record Interface

The expense records interface presents a comprehensive table view of all logged expenses associated with a selected birdhouse. Each entry includes columns for the expense category, description, monetary amount, date, and receipt status. Users can filter the records by date using the input field and filter button located above the table, allowing for focused queries based on specific reporting periods. If a receipt has been uploaded for a particular transaction, a "View Receipt" button is displayed to facilitate quick access and verification. This structured layout supports transparency and accountability in financial management while enabling users to monitor and evaluate operational costs over time.

4.5 Backend Implementation & Code Explanation

This section introduces selected backend functions of the Swiftlet Bird's Nest Harvesting Management Tracking System. All backend components were developed using PHP and MySQL, with proper form validation, session handling, and database interactions. The system follows secure and structured CRUD operations for features such as user registration, nest logging, expense tracking, and PDF generation.

4.5.1 User Registration

```php
if ($_SERVER["REQUEST_METHOD"] === "POST") {
    $username = trim($_POST['username']);
    $email = trim($_POST['email']);
    $password = $_POST['password'];

    if (empty($username) || empty($email) || empty($password)) {
        header("Location: signup.php?error=failed");
        exit();
    }

    $stmt = $conn->prepare("SELECT id FROM account WHERE email = ?");
    $stmt->bind_param("s", $email);
    $stmt->execute();
    if ($stmt->get_result()->num_rows > 0) {
        header("Location: signup.php?error=email_taken");
        exit();
    }

    $verification_token = bin2hex(random_bytes(32));
    $hashed_password = password_hash($password, PASSWORD_DEFAULT);

    $stmt = $conn->prepare("INSERT INTO account (username, email, hash_password, is_verified, verification_token) VALUES (?, ?, ?, 0, ?)");
    $stmt->bind_param("ssss", $username, $email, $hashed_password, $verification_token);
```

Figure 33: Backend Logic for User Registration and Token Generation

This figure shows the PHP backend logic for handling the user registration process in the system. It begins by checking the HTTP request method and validating the submitted form fields for completeness. The email field is then checked against the database using a prepared statement to ensure that it has not already been registered, which helps prevent duplicate accounts. If the input is valid, the user's password is securely hashed using PHP's built-in password_hash() function before being stored in the database. A unique email verification token is also generated using random_bytes() and bin2hex() to support secure email verification. This token is saved along with the user record, enabling the system to send a personalized verification

email. This process ensures secure user registration, enhances system integrity, and supports a robust authentication mechanism by enforcing email confirmation before allowing system access.

```php
if ($stmt->execute()) {
    $mail = new PHPMailer(true);
    try {
        $mail->isSMTP();
        $mail->Host       = 'smtp.gmail.com';
        $mail->SMTPAuth   = true;
        $mail->Username   = 'ericgom200302@gmail.com';
        $mail->Password   = 'mvvjilnwjdsvhdwk';
        $mail->SMTPSecure = 'tls';
        $mail->Port       = 587;

        $mail->setFrom('ericgom200302@gmail.com', 'Swift N Easy');
        $mail->addAddress($email, $username);

        $verifyURL = "https://orange-gazelle-580360.hostingersite.com/swiftlet/verify.php?token=$verification_token";

        $mail->isHTML(true);
        $mail->Subject = "Verify Your Email";
        $mail->Body    = "Hi $username,<br><br>Please click the link below to verify your email:<br><a href='$verifyURL'>$verifyURL</a>";

        $mail->send();

        header("Location: signup.php?signup=success");
    } catch (Exception $e) {
        header("Location: signup.php?error=failed");
    }
} else {
    header("Location: signup.php?error=failed");
}

$stmt->close();
$conn->close();
```

Figure 34: PHPMailer Integration for Email Verification

This figure illustrates how the PHPMailer library is configured and utilized to send a verification email upon successful user registration. After a new user's information is securely stored in the database, a unique token is generated and appended to a verification URL. The system then uses SMTP settings such as the Gmail SMTP server, port, and authentication credentials to prepare and send an HTML email containing the verification link. This ensures that only valid email addresses are associated with user accounts and helps prevent unauthorized access. By requiring users to verify their email before login, the system adds an essential layer of security and accountability.

4.5.2 Login Authentication

```php
if ($_SERVER["REQUEST_METHOD"] === "POST") {
    $password = $_POST['password'];

    if (empty($email) || empty($password)) {
        echo "All fields are required.";
        exit();
    }

    $stmt = $conn->prepare("SELECT id, username, hash_password, is_verified FROM account WHERE email = ?");
    $stmt->bind_param("s", $email);
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result->num_rows === 1) {
        $user = $result->fetch_assoc();

        if ($user['is_verified'] != 1) {
            echo "Please verify your email before logging in.";
            exit();
        }

        if (password_verify($password, $user['hash_password'])) {
            $_SESSION['user'] = [
                'id' => $user['id'],
                'username' => $user['username'],
                'email' => $email
            ];
            header("Location: dashboard.php");
            exit();
        } else {
            echo "Invalid email or password.";
        }
    } else {
        echo "Invalid email or password.";
    }

    $stmt->close();
    $conn->close();
```

Figure 35: Backend Login Authentication with Email Verification

This figure illustrates how the PHPMailer library is integrated and configured to send a verification email immediately after successful user registration. The script sets up SMTP parameters such as the mail host, port, authentication credentials, and encryption protocol to establish a secure connection with Gmail's SMTP server. Once the email transport is configured, the system creates a verification message that includes a personalized greeting using the user's name and a unique verification link. This link is dynamically generated using the verification token created during registration and is embedded as an HTML anchor tag within the email body. PHPMailer is used here in HTML mode to support formatted messages, enhancing readability and user engagement. The email is sent to the registered address, and the user is redirected based on whether the email dispatch was successful or not. This automated email verification step

improves account security by ensuring only valid and accessible email addresses are registered in the system, while also providing a more professional onboarding experience.

4.5.3 Nest Logging

```php
$response = [];

if (!isset($_SESSION['user'])) {
    echo json_encode(['success' => false, 'message' => 'Not logged in.']);
    exit();
}

include 'db_connect.php';

$birdhouse_id = intval($_POST['birdhouse_id'] ?? 0);
$weight = floatval($_POST['weight'] ?? 0);
$quality = $_POST['quality'] ?? '';
$label = $_POST['label'] ?? '';
$log_date = $_POST['log_date'] ?? '';
$comment = trim($_POST['comment'] ?? null);

if (!$birdhouse_id || !$weight || !$quality || !$log_date) {
    echo json_encode(['success' => false, 'message' => 'Missing required fields.']);
    exit();
}

$stmt = $conn->prepare("INSERT INTO nests (birdhouse_id, weight, quality, label, log_date, comment) VALUES (?, ?, ?, ?, ?, ?)");
if (!$stmt) {
    echo json_encode(['success' => false, 'message' => 'SQL error: ' . $conn->error]);
    exit();
}

$stmt->bind_param("idssss", $birdhouse_id, $weight, $quality, $label, $log_date, $comment);


if ($stmt->execute()) {
    echo json_encode(['success' => true]);
} else {
    echo json_encode(['success' => false, 'message' => $stmt->error]);
}

$stmt->close();
$conn->close();
```

Figure 36: Nest Logging Backend Function for Harvest Data Entry

This figure shows the backend PHP logic responsible for processing nest log entries submitted via the frontend form. It first verifies the user session to ensure authentication, then retrieves and validates input data such as birdhouse ID, weight, quality, grid label, collection date, and optional comments. Using prepared statements with parameter binding, the data is securely inserted into the nests table. The script responds in JSON format, returning either a success or error message to the frontend, making it suitable for use with AJAX or other asynchronous

request methods. This approach ensures proper input handling, database integrity, and scalability for future integration with IoT data or mobile logging interfaces.

4.5.4 Expense Logging

```php
$birdhouse_id = $_POST['birdhouse_id'];
$category = $_POST['category'];
$description = $_POST['description'];
$amount = $_POST['amount'];
$log_date = $_POST['log_date'];
$receipt = '';

// Check if file is uploaded
if (isset($_FILES['receipt_image']) && $_FILES['receipt_image']['error'] === UPLOAD_ERR_OK) {
    // Path to store uploaded file
    $uploadDir = 'uploads/';
    if (!is_dir($uploadDir)) {
        mkdir($uploadDir, 0755, true);  // Create the directory if it doesn't exist
    }

    $fileName = time() . '_' . basename($_FILES['receipt_image']['name']);
    $targetPath = $uploadDir . $fileName;

    // Move the file to the server
    if (move_uploaded_file($_FILES['receipt_image']['tmp_name'], $targetPath)) {
        $receipt = $targetPath;  // Save the file path
    } else {
        // If file upload fails
        echo json_encode(['success' => false, 'message' => 'Error uploading the receipt image.']);
        exit();
    }
} else {
    $receipt = null; // No receipt uploaded
}

// Prepare and insert data into the database
$stmt = $conn->prepare("INSERT INTO expenses (birdhouse_id, category, description, amount, log_date, receipt_image) VALUES (?, ?, ?, ?, ?, ?)");
$stmt->bind_param("issdss", $birdhouse_id, $category, $description, $amount, $log_date, $receipt);

if ($stmt->execute()) {
    echo json_encode(['success' => true, 'message' => 'Expense added successfully!']);
} else {
    echo json_encode(['success' => false, 'message' => 'Error inserting data: ' . $stmt->error]);
}

$stmt->close();
$conn->close();
```

Figure 37: Expense Logging with Receipt Upload and Database Insertion

This figure demonstrates the backend logic for recording birdhouse-related expenses. It begins by collecting input values such as category, description, amount, and log date. If a receipt image is uploaded, the script validates the file, creates the upload directory if necessary, and saves the file with a unique name. The image file path is then stored in the database along with the other expense data using prepared statements to prevent SQL injection. The system responds with JSON messages indicating success or failure, making it compatible with AJAX-based

frontend calls. This approach improves data integrity, supports traceable expense records, and enhances the system's usability for mobile or on-site logging.

4.5.5 Task Scheduling

```php
$birdhouseId = intval($_POST['birdhouse_id'] ?? 0);
$description = trim($_POST['description'] ?? '');
$dueDate = $_POST['due_date'] ?? '';

if ($birdhouseId && $description && $dueDate) {
    $stmt = $conn->prepare("INSERT INTO tasks (birdhouse_id, description, due_date) VALUES (?, ?, ?)");
    $stmt->bind_param("iss", $birdhouseId, $description, $dueDate);
    $stmt->execute();
}
header("Location: inProject.php?id=" . $birdhouseId);
exit();
```

Figure 38: Backend Logic for Task Scheduling and Insertion

This figure displays the backend PHP code used to add scheduled tasks to a specific birdhouse within the Swiftlet Bird's Nest Tracking Management System. The script begins by retrieving the birdhouse_id, task description, and due_date from a submitted POST request. Basic input validation is performed to ensure that all required fields are present before proceeding. The task data is then inserted into the tasks table using a prepared SQL statement, with bind_param() employed to securely bind parameters, thereby mitigating the risk of SQL injection attacks. Upon successful execution, the user is redirected to the corresponding birdhouse's project page, where the newly added task is reflected in the system interface. This backend feature is essential for maintaining routine farm operations, allowing swiftlet farmers to log and monitor tasks related to nest harvesting, equipment maintenance, or hygiene schedules. It also lays the foundation for time-based task reminders or performance analytics, contributing to overall operational efficiency.

4.5.6 Report Generation (PDF)

```php
if (isset($_POST['generate_pdf'])) {
    require_once('vendor/autoload.php'); // Adjust path if using Composer or different structure

    $pdf = new TCPDF();
    $pdf->AddPage();

    $pdf->SetFont('helvetica', 'B', 16);
    $pdf->Cell(0, 10, 'Birdhouse Task Report', 0, 1, 'C');

    $pdf->SetFont('helvetica', '', 12);
    $pdf->Ln(5);
    $pdf->Cell(0, 10, "User: " . $username, 0, 1);
    $pdf->Cell(0, 10, "Total Tasks: " . $taskTotal, 0, 1);
    $pdf->Cell(0, 10, "Completed Tasks: " . $taskCompleted, 0, 1);

    $completionRate = ($taskTotal > 0) ? round(($taskCompleted / $taskTotal) * 100, 2) : 0;
    $pdf->Cell(0, 10, "Completion Rate: " . $completionRate . "%", 0, 1);

    $pdf->Ln(10);
    $pdf->SetFont('helvetica', 'B', 12);
    $pdf->Cell(60, 10, 'Task Name', 1);
    $pdf->Cell(60, 10, 'Status', 1);
    $pdf->Ln();

    // Re-fetch tasks for table display
    $taskQuery->execute();
    $taskResults = $taskQuery->get_result();

    $pdf->SetFont('helvetica', '', 11);
    while ($task = $taskResults->fetch_assoc()) {
        $pdf->Cell(60, 10, $task['description'], 1);
        $pdf->Cell(60, 10, $task['is_completed'], 1);
        $pdf->Ln();
    }

    $pdf->Output('task_report.pdf', 'I');
    exit();
}
```

Figure 39: Task Report Export Function Using TCPDF

This figure shows the PHP code used to export a birdhouse task report to a downloadable PDF using the TCPDF library. The script first initializes a new PDF document and formats the layout using custom fonts, cell spacing, and alignment to ensure readability. It then compiles summary statistics, including the total number of tasks, completed tasks, and an automatically calculated completion rate, which provides users with a quick overview of their birdhouse management performance. Following the summary, the code dynamically populates a table with individual task descriptions and their completion status by fetching data from the database. The

final output is rendered in-browser and can be saved or printed instantly by the user. This functionality enhances reporting, supports record-keeping, and allows stakeholders to generate structured documentation of operational activities with a single click.

4.5.7 Grid Labels

```php
$userId = $_SESSION['user']['id'];

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $birdhouse_name = trim($_POST['birdhouse_name'] ?? '');
    $location = trim($_POST['location'] ?? '');
    $floors = intval($_POST['floors'] ?? 1);
    $grid_labels = json_decode($_POST['grid_labels'] ?? '[]', true);
    // var_dump($grid_labels);
    if (!$birdhouse_name || !$location || !isset($_FILES['birdhouse_photo'])) {
        echo json_encode(['success' => false, 'message' => 'Missing required fields.']);
        exit();
    }
```

Figure 40: Retrieving and Validating Grid Label Input

```php
    // Insert labels if available
    if (!empty($grid_labels)) {
        $stmtLabel = $conn->prepare("INSERT INTO birdhouse_grid_labels
            (birdhouse_id, floor, row, col, label, total_rows, total_cols)
            VALUES (?, ?, ?, ?, ?, ?, ?)");

        if (!$stmtLabel) {
            echo json_encode(['success' => false, 'message' => 'Label prepare failed: ' . $conn->error]);
            exit();
        }

        mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT); // Add this to show real SQL errors

    foreach ($grid_labels as $label) {
        $floor = $label['floor'];
        $row = $label['row'];
        $col = $label['col'];
        $text = $label['label'];
        $total_rows = $label['total_rows'];
        $total_cols = $label['total_cols'];

        $stmtLabel = $conn->prepare("INSERT INTO birdhouse_grid_labels (birdhouse_id, floor, row, col, label, total_rows, total_cols) VALUES (?, ?, ?, ?, ?, ?, ?)");
        if (!$stmtLabel) {
            echo json_encode(['success' => false, 'message' => 'Label prepare failed: ' . $conn->error]);
            exit();
        }

        $stmtLabel->bind_param("iiiisii", $birdhouse_id, $floor, $row, $col, $text, $total_rows, $total_cols);

        if (!$stmtLabel->execute()) {
            echo json_encode(['success' => false, 'message' => 'Label insert failed: ' . $stmtLabel->error]);
            exit();
        }
    }

        $stmtLabel->close();
```

Figure 41: Grid Label JSON Parsing and Insert Preparation

These figures demonstrate the backend logic for managing custom grid labels during birdhouse creation. Figure 42 retrieves and decodes the grid_labels data from the frontend form

submission. Figure 43 validates and inserts each label into the birdhouse_grid_labels table using secure prepared statements. Each label is saved with detailed position attributes such as floor, row, and column, along with its total grid dimensions. This design supports accurate mapping of nests in multi-floor birdhouses and enhances usability for farmers when viewing or logging environmental or nest data.

4.5.8 Environmental Status Monitoring

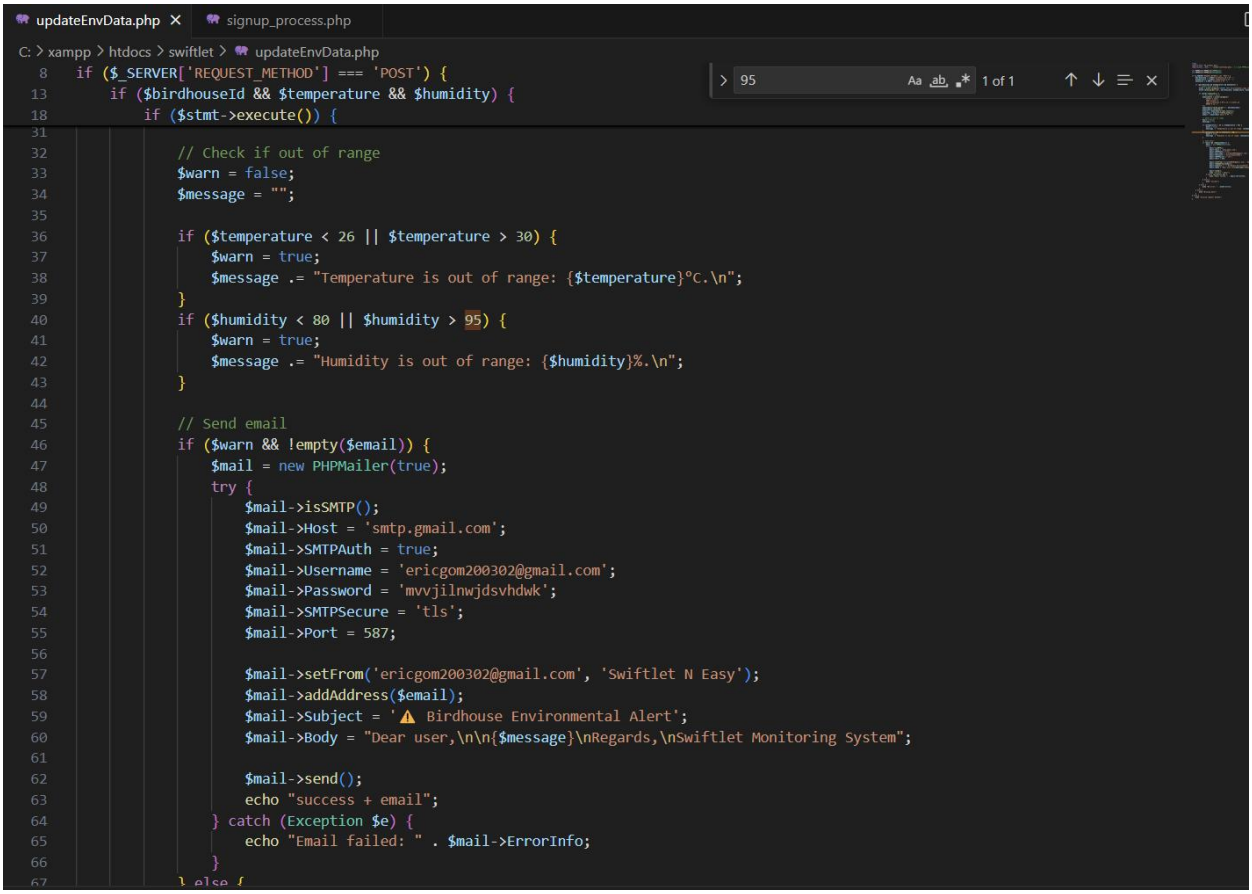


Figure 42: Automated Environmental Alert Logic in updateEnvData.php

To make the monitoring system more responsive, an email alert feature was added using PHPMailer. Whenever the ESP32 detects that the temperature or humidity goes beyond the ideal range—between 26°C and 30°C for temperature, and 80% to 95% for humidity—it automatically sends a warning email to the registered user. This helps ensure that any sudden changes in the

birdhouse environment can be addressed quickly, preventing conditions that could harm the swiftlets or disrupt their nesting. The threshold values are based on recommended environmental conditions for swiftlet farming. According to Sulaiman et al. (2014), swiftlets thrive best in environments where the temperature ranges between 26°C to 30°C and humidity ranges between 80% to 90%. These environmental conditions support ideal nesting and egg hatching within the birdhouse. Therefore, the system monitors and flags any values that fall outside these thresholds, ensuring users can take timely corrective action to maintain a healthy habitat.

4.6 Summary

Chapter 4 detailed the implementation of both frontend interfaces and backend functionalities that form the core of the Swiftlet Bird's Nest Harvesting Management Tracking System. The user interfaces were developed with a strong emphasis on usability, ensuring that farmers with minimal technical experience can interact with the system intuitively. Each page from landing and registration to project dashboards, grid labeling, nest logging, and expense tracking was designed to support specific user tasks while maintaining visual consistency and responsiveness across devices. On the backend, PHP and MySQL were used to power critical operations such as secure user authentication, data validation, nest and expense logging, and task scheduling. Features like receipt image uploads and real-time report generation demonstrate the system's practical value in organizing and documenting swiftlet farming activities. Secure database interactions using prepared statements were employed to ensure data integrity and prevent common vulnerabilities such as SQL injection. Additionally, the system supports scalable reporting through downloadable PDF reports for task management summaries, made possible via integration with the TCPDF library. These reporting features improve transparency and allow stakeholders to maintain a documented history of birdhouse operations. Altogether, the

implementation of both user-facing interfaces and backend modules demonstrates a strong alignment with the system's objectives. The chapter showcased how practical design and structured coding were combined to support streamlined, efficient, and scalable digital management of swiftlet farming tasks.

Chapter 5: Testing

5.1 Introduction

This chapter presents the evaluation of the Swiftlet Bird's Nest Harvesting Management Tracking System, focusing on the system's functionality, design performance, and user interaction. The evaluation is based on the implementation results discussed in Chapter 4, particularly how effectively the system meets the objectives defined in earlier stages of the project. Through this discussion, the strengths and weaknesses of the prototype, interface design, database structure, and overall user experience are identified. Additionally, this chapter considers feedback on the system's usability, efficiency, and any limitations encountered during testing. By reflecting on these findings, recommendations for future improvements and refinements are proposed to enhance the system's overall effectiveness and reliability.

5.2 Functional Testing

Functional testing was conducted to ensure that each module in the Swiftlet Bird's Nest Harvesting Management Tracking System performs its intended operations correctly and consistently. The testing process verified that all components operate without critical errors and meet the requirements outlined during the design and analysis phase.

This testing was done from both admin and regular user perspectives. It includes the evaluation of user registration and login, birdhouse management, labeling grids, logging nests and expenses, managing task inputs, and generating reports. The results are summarized below, covering both valid and invalid input scenarios to assess the system's robustness.

User Module:

- Sign Up & Login
- Birdhouse Creation
- Grid Labeling
- Nest Logging
- Expense Logging
- Task Input
- Report Generation

Each function was tested for both positive and negative cases to ensure reliable input handling and system feedback.

5.2.1 Swiftlet Bird's Nest Harvesting Management Tracking System Positive Test Cases

Table 6: Sign Up Test Case

Test Case: 1 **Test Case Name:** User Registration **System:** Swiftlet Bird's Nest Harvesting Management Tracking System **Short Description:** Verifies that a new user can successfully register using valid credentials and is redirected to the login page				
Pre-condition(s): 1. The user has not registered before and the email is unique				
Step	**Action**	**Expected Outcome**	**Result**	**Comment**
1	User opens the	Sign Up form is	Pass	Page loads

	Sign Up Page.	displayed with required input fields		successfully
2	User fills in valid email, username, and password	Form accepts input and enables the Register button	Pass	Input validation is triggered
3	User clicks the Register button	User is registered and redirected to the Login Page	Pass	Data is saved and redirected properly
Post-condition(s): 1. The user account is created and the onboarding process is complete.				

Table 7: Add Birdhouse Test Case

Test Case: 2 **Test Case Name:** Birdhouse Creation with All Valid Inputs **System:** Swiftlet Bird's Nest Harvesting Management Tracking System **Short Description:** Confirms that a birdhouse is created when the user provides all required information correctly.				
Pre-condition(s): 1. User is logged in and on the "Add Birdhouse" page. 2. Fields are all initially empty.				
Step	**Action**	**Expected Outcome**	**Result**	**Comment**
1	Fields are all initially empty.	Field accepts input	Pass	Text field accepts string
2	User selects a location	Dropdown selection is saved	Pass	Location list works
3	User uploads a valid photo file	File appears in upload preview	Pass	Image upload preview shows
4	User selects number of floors (e.g., 2)	Floor sections are dynamically displayed	Pass	JavaScript function triggers
5	User sets grid rows and	Grid preview updates	Pass	UI reflects inputs

	columns for each floor	accordingly		
6	User clicks "Submit Birdhouse Info"	System creates birdhouse and redirects to projects page	Pass	Redirect + DB insert success
Post-condition(s): 1. Birdhouse record is saved and shown in the project overview.				

Table 8: Grid Label Test Case

Test Case: 3 **Test Case** Name: Birdhouse Creation with All Valid Inputs **System:** Swiftlet Bird's Nest Harvesting Management Tracking System **Short Description:** Confirms that a birdhouse is created when the user provides all required information correctly.				
Pre-condition(s): 1. User is logged in and on the "Add Birdhouse" page. 2. Fields are all initially empty.				
Step	**Action**	**Expected Outcome**	**Result**	**Comment**
1	Fields are all initially empty.	Field accepts input	Pass	Text field accepts string
2	User selects a location	Dropdown selection is saved	Pass	Location list works
3	User uploads a valid photo file	File appears in upload preview	Pass	Image upload preview shows
4	User selects number of floors (e.g., 2)	Floor sections are dynamically displayed	Pass	JavaScript function triggers
5	User sets grid rows and columns for each floor	Grid preview updates accordingly	Pass	UI reflects inputs
6	User clicks "Submit	System creates birdhouse and	Pass	Redirect + DB

	Birdhouse Info"	redirects to projects page		insert success
Post-condition(s): 1. Birdhouse record is saved and shown in the project overview.				

Test Case: 4 **Test Case Name:** Label Grouping **System:** Swiftlet Bird's Nest Harvesting Management Tracking System **Short Description:** Verifies that a user can intentionally apply the same label to more than one cell to represent grouped nest positions (e.g., shared nests).				
Pre-condition(s): 1. The user has already labeled one or more cells on the grid. 2. The label (e.g., A1) is already used on one cell.				
Step	**Action**	**Expected Outcome**	**Result**	**Comment**
1	User selects a different cell on the same floor	Cell is highlighted.	Pass	Cell selection works as expected
2	User enters an existing label (e.g., A1) into the label field	Input is accepted	Pass	System allows repeated label
3	User clicks "Apply Label"	Label is added to the second cell, same as the first	Pass	System reflects group labeling
Post-condition(s): 1. Multiple cells display the same label. This is treated as a group label (if supported), or simply a repeated reference.				

Table 9: Log Nest All Valid Inputs Test Case

Test Case: 5

Test Case Name: Log Nest Successfully with All Valid Inputs

System: Swiftlet Bird's Nest Harvesting Management Tracking System

Short Description: Verifies that a user can successfully submit a nest log when all required fields are correctly filled.

Pre-condition(s):

1. User is logged in.
2. Birdhouse and labeled grid exist.

Step	Action	Expected Outcome	Result	Comment
1	User selects birdhouse and grid cell	Cell is activated.	Pass	Cell selection registered
2	User enters weight, quality, date , comment	All inputs are accepted	Pass	Validation passed
3	User clicks "Save Nest"	System saves nest log and confirms	Pass	Entry recorded

Post-condition(s):

1. Nest record is saved and visible in the system.

Table 10: Log Nest Without Comment Test Case

Test Case: 6

Test Case Name: Log Nest Without Comment

System: Swiftlet Bird's Nest Harvesting Management Tracking System

Short Description: Confirms that comment field is optional and nest can be logged without it.

Pre-condition(s):

1. User accesses the nest logging page.

Step	Action	Expected Outcome	Result	Comment
1	User fills all fields except	Form validates	Pass	Optional field

	comment	successfully		handled properly
2	User submits the form	Nest log saved	Pass	Entry accepted without comment
Post-condition(s): 1. Nest log saved with null or empty comment.				

Table 11: Log Nest on Different Floor Test Case

Test Case: 7 **Test Case Name:** Log Nest on Different Floor **System:** Swiftlet Bird's Nest Harvesting Management Tracking System **Short Description:** Confirms that user can log nests on a specific floor of a multi-floor birdhouse.				
Pre-condition(s): Birdhouse has 2+ floors configured.				
Step	**Action**	**Expected Outcome**	**Result**	**Comment**
1	User selects floor 2 and a grid cell	Grid updates and accepts label	Pass	Floor logic works
2	User enters other details and submits	Nest log saved for correct floor	Pass	Data mapped to floor correctly
Post-condition(s): 1. Nest appears under the right floor in logs.				

Table 12: Log Expense With All Valid Input Test Case

Test Case: 8 **Test Case Name:** Log Expense with All Valid Input

System: Swiftlet Bird's Nest Harvesting Management Tracking System

Short Description: Confirms that a user can successfully submit an expense record with all required fields filled.

Pre-condition(s):

1. User is logged in.

2. Birdhouse exists with at least one expense category (e.g., Maintenance).

Step	Action	Expected Outcome	Result	Comment
1	User navigates to the inProject page and clicks "Add Expense"	Expense modal appears	Pass	Modal loaded
2	User selects a category, enters a description (e.g., "Wire Fix"), and enters amount (e.g., 100)	Fields accept valid input	Pass	Inputs processed
3	User selects a valid date	Date selected correctly	Pass	No issues with date picker
4	User uploads a valid image (e.g., .jpg receipt)	Image preview appears or filename shown	Pass	Upload successful
5	User clicks "Save Expense"	Data saved; confirmation alert shown	Pass	Record stored successfully

Post-condition(s):

1. The expense is stored in the database and appears in the birdhouse's expense records.

Table 13: Log Expense Without Receipt Image Test Case

Test Case: 9

Test Case Name: Log Expense Without Receipt Image

System: Swiftlet Bird's Nest Harvesting Management Tracking System

Short Description: Verifies that users can successfully submit an expense without uploading a

receipt.				
Pre-condition(s): User is logged in and accessing the expense modal.				
Step	**Action**	**Expected Outcome**	**Result**	**Comment**
1	User clicks “Add Expense” in dashboard	Modal appears	Pass	Form loaded
2	User fills in category, description, amount, and date, but skips image upload	All required inputs accepted	Pass	Optional field ignored
3	User clicks “Save Expense”	Form submitted successfully	Pass	Image not required
Post-condition(s): 1. Expense is stored in the database with a null or empty image field.				

Table 14: Generate Task Report Test Case

Test Case: 10 **Test Case Name:** Generate Task Report **System:** Swiftlet Bird’s Nest Harvesting Management Tracking System **Short Description:** Confirms that the system can generate and download a task report in PDF format.				
Pre-condition(s): User is logged in. At least one task is recorded under the selected birdhouse.				
Step	**Action**	**Expected Outcome**	**Result**	**Comment**
1	User opens the birdhouse detail	Task section and report button are	Pass	UI loaded successfully

	page	visible		
2	User clicks "Download Task Report" button	System processes report request	Pass	Button triggers backend
3	PDF is generated and downloaded to device	PDF contains all task data	Pass	File downloaded with content
Post-condition(s): 1. PDF report is generated and downloaded with current data.				

Table 15: Report Includes Completed and Pending Tasks Test Case

Test Case: 11 **Test Case Name:** Report Includes Completed and Pending Tasks **System:** Swiftlet Bird's Nest Harvesting Management Tracking System **Short Description:** Verifies that both completed and uncompleted tasks appear in the PDF with correct status.				
Pre-condition(s): 1. Birdhouse has multiple tasks with different completion statuses.				
Step	**Action**	**Expected Outcome**	**Result**	**Comment**
1	User ensures tasks are marked both completed and pending	Task list updated	Pass	Data prepared
2	User downloads report	Report includes all tasks	Pass	Task status visible
3	User opens PDF	Status for each task is correct	Pass	Data reflects system status
Post-condition(s): 1. Report shows full task list with accurate status indicators.				

Table 16: Generate Report Without Image or Styling Errors Test Case

Test Case: 12 **Test Case Name:** Generate Report Without Image or Styling Errors **System:** Swiftlet Bird's Nest Harvesting Management Tracking System **Short Description:** Verifies that the report layout is structured and does not break visually on PDF export.				
Pre-condition(s): 1.System is using properly formatted HTML-to-PDF conversion.				
Step	**Action**	**Expected Outcome**	**Result**	**Comment**
1	User clicks report download	PDF generated as usua	Pass	No UI crash
2	User opens file in PDF reader	Layout, fonts, spacing are intact	Pass	PDF export styled properly
Post-condition(s): 1. PDF displays in a clean, readable format.				

Table 17: Add Task with Valid Description and Due Date Test Case

Test Case: 13 **Test Case Name:** Add Task with Valid Description and Due Date **System:** Swiftlet Bird's Nest Harvesting Management Tracking System **Short Description:** Verifies that the user can add a task successfully using a valid description and future due date.				
Pre-condition(s): User is logged in. User is viewing a birdhouse detail page (inProject interface).				
Step	**Action**	**Expected**	**Result**	**Comment**

		Outcome		
1	User scrolls to the "Task" section of the inProject page	Task input form is visible	Pass	UI loads correctly
2	User types a valid task description (e.g., "Clean floor on Level 2") into the text box	Field accepts and displays input	Pass	Input is responsive
3	User selects a valid future date using the date picker (e.g., tomorrow's date)	Calendar date is accepted	Pass	Date input is functional
4	User clicks the "Add Task" button	Task is saved and added to the task list immediately	Pass	Data inserted and displayed in real-time
Post-condition(s): 1. Task is saved in the database and displayed under the task list.				

Table 18: Add Multiple Tasks Back-to-Back Test Case

Test Case: 14 **Test Case Name:** Add Multiple Tasks Back-to-Back **System:** Swiftlet Bird's Nest Harvesting Management Tracking System **Short Description:** Verifies that the system allows users to submit multiple tasks in a row without errors.				
Pre-condition(s): User is already in the task section of the birdhouse detail page.				
Step	**Action**	**Expected Outcome**	**Result**	**Comment**
1	User enters a task and due date, then clicks	First task is added	Pass	UI updates with new entry

	"Add Task"			
2	User immediately enters a second task and date	Form resets and accepts new input	Pass	Reusable form
3	Clicks "Add Task" again	Second task is saved and appears	Pass	System handles consecutive input
Post-condition(s): All submitted tasks are listed and stored in the database.				

Table 19: Environmental Data Logging Without Alert

Test Case: 15 **Test Case Name:** Environmental Data Logging Without Alert **System:**Swiftlet Bird's Nest Harvesting Management Tracking System **Short Description:** Verifies that environmental data is logged correctly and no email alert is sent when temperature and humidity values are within the acceptable range (Temperature: 26°C–30°C, Humidity: 80%–95%).				
Pre-condition(s): 1. The ESP32 device is connected and actively sending data. 2. The birdhouse has a registered user with a valid email address.				
Step	**Action**	**Expected Outcome**	**Result**	**Comment**
1	ESP32 sends Temperature = 28°C, Humidity = 85% to server	Data is saved to environmental_logs	Pass	Values are within safe range
2	Server evaluates the values	No email alert triggered	Pass	System behaves as expected
3	User checks environmental status in UI	Values are displayed correctly	Pass	UI reflects latest data

Post-condition(s): 1. No alert email is sent. 2. Environmental log entry is added successfully. 3. User sees normal status in the frontend.

5.2.2 Swiftlet Bird's Nest Harvesting Management Tracking System Negative Test Cases

Table 20: Sign Up Negative Test Case

Test Case: 16 **Test Case Name:** Sign Up with Empty Fields **System:** Swiftlet Bird's Nest Harvesting Management Tracking System **Short Description:** Checks whether the system properly prevents registration when required fields are left empty.				
Pre-condition(s): 1. The user has not entered any input in the sign-up form.				
Step	**Action**	**Expected Outcome**	**Result**	**Comment**
1	User opens the Sign Up Page	Sign Up form is displayed with input fields	Pass	Page loads without error
2	User leaves all fields empty and clicks Register	System displays error or validation message; registration blocked	Pass	SweetAlert2 or form validation triggers
3	User remains on the same page	No redirection; no data submitted to database	Pass	System correctly prevents submission
Post-condition(s): 1. No user account is created and the user remains on the sign-up page.				

Table 21: Grid Empty Label Input

Test Case: 17 **Test Case Name:** Grid Empty Label Input **System:** Swiftlet Bird's Nest Harvesting Management Tracking System **Short Description:** Checks whether the system prevents saving when the label field is empty.				
Pre-condition(s): 1. Grid cell is selected. 2. Label input field is left empty.				
Step	**Action**	**Expected Outcome**	**Result**	**Comment**
1	User selects a cell	Cell is highlighted.	Pass	Normal behaviour.
2	User leaves label field blank and clicks "Apply Label"	System shows validation error or does nothing.	Pass	Label is not applied.
3	User remains on the same screen.	No label saved or shown.	Pass	System enforces required field.
Post-condition(s): 1. No label is applied or stored.				

Table 22: Log Nest Empty Required Fields Test Case

Test Case: 18 **Test Case Name:** Submit with Empty Required Fields **System:** Swiftlet Bird's Nest Harvesting Management Tracking System **Short Description:** Ensures the form prevents submission if required fields are blank.
Pre-condition(s):

1. Nest logging form is open.				
Step	**Action**	**Expected Outcome**	**Result**	**Comment**
1	User leaves weight and label empty	Form is blocked	Pass	Validation triggers correctly
2	User tries to submit	Error message shown	Pass	Form not processed
Post-condition(s): 1. No data saved; user prompted to complete fields.				

Table 23: Submit Invalid Nest Weight Test Case

Test Case: 19 **Test Case Name:** Submit Invalid Nest Weight (negative weight) **System:** Swiftlet Bird's Nest Harvesting Management Tracking System **Short Description:** Verifies that system blocks invalid weight input (e.g., "-10" or "abc").				
Pre-condition(s): 2. User is on the nest form.				
Step	**Action**	**Expected Outcome**	**Result**	**Comment**
1	User enters "-10" or "abc" into weight field	System blocks input or shows error	Pass	Type & range validation works
2	Submission is attempted	Form shows validation message	Pass	Invalid data blocked
Post-condition(s): 1. Invalid values not accepted or saved.				

Table 24: Submit Without Selecting Grid Label Test Case

Test Case: 20 **Test Case Name:** Submit Without Selecting Grid Label **System:** Swiftlet Bird's Nest Harvesting Management Tracking System **Short Description:** Ensures the form cannot be submitted unless a grid label is selected.				
Pre-condition(s): 3. User skips cell selection.				
Step	**Action**	**Expected Outcome**	**Result**	**Comment**
1	User leaves grid label unselected	Label field marked as required	Pass	Grid selection enforced
2	User tries to save	Error prevents submission	Pass	Submission blocked
Post-condition(s): 1. Nest log not saved; user remains on page.				

Table 25: Submit with Empty Required Fields Test Case

Test Case: 21 **Test Case Name:** Expense Logging Submission with Empty Required Fields **System:** Swiftlet Bird's Nest Harvesting Management Tracking System **Short Description:** Ensures that the system prevents submission if category or amount is missing.				
Pre-condition(s): 4. User is logged in and opens the Add Expense form.				
Step	**Action**	**Expected Outcome**	**Result**	**Comment**
1	User clicks "Add Expense"	Modal appears	Pass	UI functional

2	User leaves category or amount field empty and clicks "Save"	System shows validation error	Pass	Required field check triggered
3	User remains on the same screen	No data stored	Pass	Submission blocked as expected
Post-condition(s): 1. Expense is not saved; system blocks submission.				

Table 26: Expense Log Enter Invalid Amount Format

Test Case: 22 **Test Case Name:** Enter Invalid Amount Format **System:** Swiftlet Bird's Nest Harvesting Management Tracking System **Short Description:** Ensures that the system restricts non-numeric inputs in the amount field.				
Pre-condition(s): 1. User accesses the expense form.				
Step	**Action**	**Expected Outcome**	**Result**	**Comment**
1	User enters "abc" or "$10" in the amount field	System flags input or prevents submission	Pass	Data type validation works
2	System flags input or prevents submission	System shows warning or error	Pass	Invalid input blocked
Post-condition(s): 1. Expense not saved; validation message displayed.				

Table 27: Expense Log Upload Invalid Receipt File Type Test Case

Test Case: 23

Test Case Name: Upload Invalid Receipt File Type

System: Swiftlet Bird's Nest Harvesting Management Tracking System

Short Description: Verifies that the system only allows valid image file types for receipt uploads.

Pre-condition(s):

1. User accesses Add Expense form and attempts to upload file.

Step	Action	Expected Outcome	Result	Comment
1	User uploads a .exe or .txt file as receipt	System shows error or does not accept file	Pass	File extension filter working
2	User attempts to save expense	Submission blocked or alert triggered	Pass	Upload validation passed

Post-condition(s):

1. Form is blocked or file rejected; expense not saved.

Table 28: Generate Report With No Tasks Logged

Test Case: 24

Test Case Name: Generate Report With No Tasks Logged

System: Swiftlet Bird's Nest Harvesting Management Tracking System

Short Description: Ensures system handles report generation gracefully when no task data exists.

Pre-condition(s):

1. Birdhouse has no tasks recorded.

Step	Action	Expected Outcome	Result	Comment
1	User visits birdhouse without tasks	UI still shows report button	Pass	File extension filter working

2	User clicks "Download Task Report"	System creates blank or informative PDF	Pass	Upload validation passed
3	PDF opens with message	Displays "No tasks to display"	Pass	Gracefull fallback behavior
Post-condition(s): 1. Report is generated with a message such as "No tasks available."				

Table 29: Add Task With Invalid Due Date

Test Case: 25 **Test Case Name:** Add Task Without Entering Description **System:** Swiftlet Bird's Nest Harvesting Management Tracking System **Short Description:** Ensures the system prevents submission if the task description is left empty.				
Pre-condition(s): 1. User is on the inProject task input form.				
Step	**Action**	**Expected Outcome**	**Result**	**Comment**
1	User selects a due date but leaves the description blank	System flags input as incomplete	Pass	Description is required
2	User clicks "Add Task""	No task is added	Pass	Validation prevents submission
3	User remains on the same page	Input field remains focused	Pass	Encouraged to correct input
Post-condition(s): 1. Task is not saved; user is shown an error or blocked from submitting.				

Table 30: Environmental Alert Email Trigger

Test Case: 26

Test Case Name: Environmental Alert Email Trigger

System: Swiftlet Bird's Nest Harvesting Management Tracking System

Short Description: Verifies that the system sends an alert email to the user when the temperature or humidity falls outside the recommended range (Temperature: 26°C–30°C, Humidity: 80%–95%).

Pre-condition(s):

1. The ESP32 is powered and connected to Wi-Fi.
2. The birdhouse has a registered owner with a valid email address.
3. The PHPMailer setup is correctly configured and functional.

Step	Action	Expected Outcome	Result	Comment
1	ESP32 sends Temperature = 24.5°C, Humidity = 97% to the server	Data is saved in environmental_logs	Pass	Out-of-range values received
2	Server evaluates data and detects environmental violation	Alert email is triggered	Pass	Alert logic works as intended
3	Registered user checks email	Alert message is received	Pass	Email includes temperature and humidity warnings

Post-condition(s):

1. A new entry is saved in environmental_logs.
2. The registered user receives an environmental warning email.
3. Warning message also appears in the frontend dashboard under "Environmental Status."

5.3 Acceptance Testing

The acceptance testing for the Swiftlet Bird's Nest Harvesting Management Tracking System was conducted with a swiftlet birdhouse owner who was previously involved during the requirement analysis phase. As an experienced stakeholder, their insights were used to validate both the functionality and usability of the system. The results are documented in two parts: functional acceptance and usability evaluation.

5.3.1 Functional Testing Feedback

The table below summarizes the stakeholder's evaluation of each major system function based on whether it worked correctly and fulfilled its intended purpose.

Table 31: Stakeholder Feedback Table

Feature	**Stakeholder Feedback**	**Status**
User Registration/ Login	Simple and working as expected	Accepted
Birdhouse Creation	Intuitive and effective for entering house details	Accepted
Grid Labeling	Very helpful; improved visibility and nest tracking	Accepted
Nest Logging	Easy to use and accurate	Accepted
Expense Logging	Useful for keeping cost records	Accepted
Task Input	Helpful for organizing cleaning and harvest dates	Accepted
PDF Report Generation	Works well, useful for reviewing summary data	Accepted

5.3.2 Usability Evaluation

While no formal usability scale (like SUS or a 10-person Google Form) was used, the stakeholder gave detailed feedback during an interactive walkthrough of the system. The table below summarizes their qualitative evaluation.

Table 32: Stakeholder Usability Evaluation Table

Usability Aspect	**Stakeholder Observation**
Ease of Use	Very easy to navigate, user-friendly
Interface Clarification	Clean layout and clear input labels
Satisfaction with features	System covers exactly what the swiftlet farmers require for their routinely birdhouse visit.
Areas for improvement	Suggested improvements included support for light mode and the addition of task alert reminders
Overall impression	System is good for farming practice because of its record organization.

Figure 43: Stakeholder Usability Ratings Based on Walkthrough Experience

5.4 Summary

This chapter presented a comprehensive evaluation of the Swiftlet Bird's Nest Harvesting Management Tracking System. Through functional testing, each core feature—including user registration, birdhouse creation, grid labeling, nest logging, expense tracking, task input, and report generation—was tested and confirmed to operate according to the intended requirements. Both positive and negative test cases were executed to ensure the system handled valid inputs effectively and prevented invalid submissions through proper validation.

In addition to developer testing, usability was assessed through an acceptance walkthrough with a real-world stakeholder. The feedback received highlighted the system's ease of use, practicality, and usefulness for swiftlet farming operations, particularly for users with limited technical experience. The stakeholder confirmed that the system simplified their data management process and aligned well with their daily routines.

Although minor limitations such as mobile responsiveness and the absence of notification features were noted, they do not affect the core functionality. Overall, the system was found to be stable, user-friendly, and suitable for its intended purpose, with room for future enhancements based on user needs.

CHAPTER 6 : CONCLUSION AND FUTURE WORK

6.1 Introduction

This chapter provides a concluding overview of the project and reflects on the overall outcome of the Swiftlet Bird's Nest Harvesting Management Tracking System. It summarizes the degree to which the project met its objectives, outlines the limitations faced during development, and suggests potential improvements that could be implemented in the future to enhance the system's functionality and usability.

6.2 Achievement of Objectives

All primary objectives of this project were successfully achieved. A functional, web-based platform was developed to support swiftlet farmers in managing their birdhouses more effectively. The system allows users to create birdhouse entries, define grid-based floor plans, log nest data with precise location tagging, record operational expenses, schedule cleaning and harvesting tasks, and generate downloadable summary reports in PDF format.

Table 33: Achievement of Objectives

Objectives	Status	Description of Achievement
To develop a system for registering and organizing birdhouse information	Achieved	Users can input birdhouse name, floors, and upload photos
To implement a grid labeling system for nest tracking	Achieved	Users can dynamically define floors and grid cells per birdhouse.
To enable logging of nest data including type, weight, and location	Achieved	Nest data can be logged with quality and location labels
To allow expense tracking and reporting	Achieved	Users can input expenses by category and view summaries
To support task scheduling	Achieved	Users can input tasks with due

		dates.
To generate PDF reports summarizing nest or task records	Achieved	System generates downloadable PDF reports for user tasks and harvest data

6.3 Limitations

Despite meeting its key objectives, the system has several limitations. First, all data inputs are handled manually by users, which can be time-consuming and may introduce human error. The system does not currently support any automated sensor input or integration with IoT devices, which could enhance data accuracy and reduce effort. Second, mobile responsiveness could be further improved to provide a smoother experience for users accessing the system on smartphones or tablets. Additionally, the system lacks real-time notifications or reminders for upcoming tasks, which some users expressed would be useful for improving daily management.

Finally, due to limited access to a large user group, the system was primarily evaluated by a single stakeholder. Broader usability testing across multiple swiftlet farmers would be beneficial for identifying additional pain points and areas of improvement.

Table 34: Limitations Table

Limitation Area	Description
Manual Data Entry	Users must input all data manually, which may introduce errors or take more time
Mobile Responsiveness	Basic mobile support exists, but layout could be optimized further
Limited User Testing	Feedback was based on a single stakeholder; broader testing would improve validation
No Task Notifications	Tasks must be checked manually; the system does not send SMS or email reminders

6.4 Future Works

Several enhancements can be explored in future iterations of the system. One of the most impactful would be to integrate mobile push notifications or SMS/email alerts for scheduled tasks such as cleaning, harvesting, or monthly maintenance. This feature would help users stay informed without the need to log in constantly.

The system could also benefit from real-time data visualization, such as dashboards displaying nest production trends or expense summaries using charts and graphs. Additionally, support for photo tagging or uploading images of nest locations may further enhance traceability and reporting quality.

Long-term improvements may include developing a native mobile app or integrating basic AI to predict harvest schedules based on past data trends.

Table 35: Future Works Table

Suggested Improvement	Purpose/Benefit
Add SMS/email task notifications	Keep users informed about upcoming tasks without logging in
Improve mobile responsiveness	Allow smooth access from smartphones in birdhouse environments
Add visual dashboards (charts/graphs)	Help users visualize nest production trends and expense summaries
Develop native mobile app (long term)	Allow offline data entry

6.5 Conclusion

In conclusion, the Swiftlet Bird's Nest Harvesting Management Tracking System successfully provides a practical, user-friendly solution for managing swiftlet farming operations. Through its modular design and intuitive interface, it streamlines the core tasks that farmers

face—nest logging, expense tracking, task scheduling, and reporting. The system's successful implementation and positive stakeholder evaluation reflect its potential to be adopted in real-world settings. With further refinement and extended features, this system can evolve into a comprehensive digital tool that supports modern and scalable swiftlet farming practices.

References:

AgriWebb. (2024). Revolutionizing farm mapping and management. https://staging.aidoos.com/kb/products-agriwebb-agriwebb-revolutionizing-farm-mapping-and-management/

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Thomas, D. (2001). Manifesto for agile software development. Agile Alliance. https://agilemanifesto.org/

Department of Primary Industries and Regions, South Australia. (2024, March 27). Livestock management software—AgriWebb. https://pir.sa.gov.au/research/agtech/find_solutions/products/agriwebb

Pezzulo, A., Guo, H., Marchesini, G., Brscic, M., Guercini, S., & Marinello, F. (2021). Digital technologies and automation in livestock production systems: A digital footprint from multisource data. *Digital Technologies and Automation in Livestock Production Systems*, 258–262. https://doi.org/10.1109/metroagrifor52389.2021.9628544

Sulaiman, S., Zulkifli, R., Omar, D., & Hussain, N. H. (2014). Design requirements for successful swiftlet farming in Malaysia. Journal of Asian Scientific Research, 4(2), 81–91.

Sutherland, J., & Schwaber, K. (2017). *The Scrum Guide*. Scrum.org. https://scrumguides.org/

VersionOne. (2020). State of agile report. https://www.versionone.com/state-of-agile/

Appendix A: Questionnaire

Operational Challenges (Data Tracking & Logging)

- What are the main challenges you face in structuring and retrieving bird nest data?
- How do you currently keep track of nest conditions (egg presence, newly built, ready for harvest)?
- Have you ever experienced difficulties in retrieving past nest records?
- What are the most common issues you face when scheduling nest harvests?
- How often do you visit your birdhouses, and what activities do you usually perform (harvesting, cleaning, maintenance)?

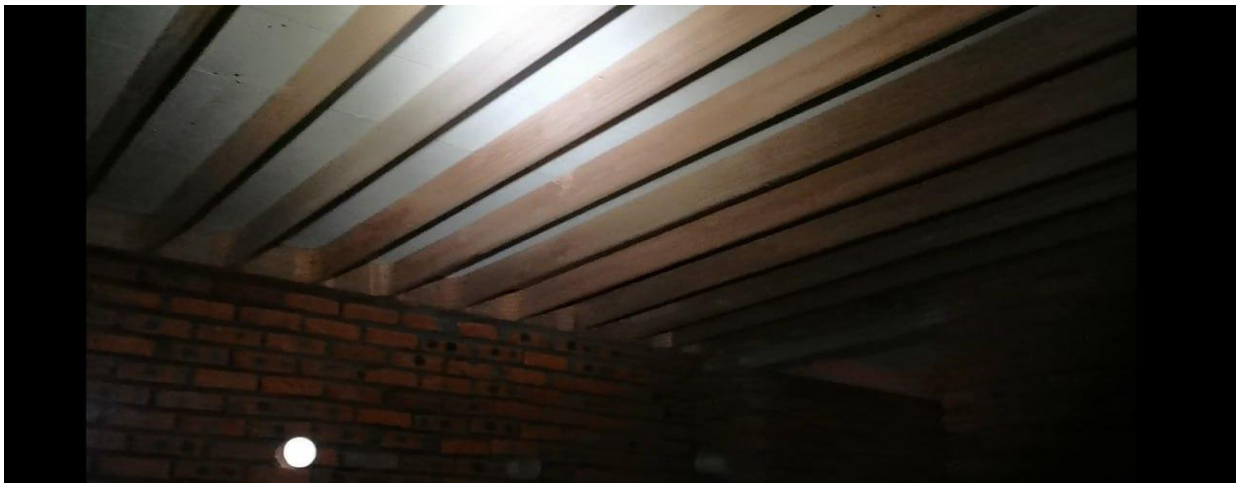
Desired Features (System Functionalities)

- What additional features or functionality would you like in the system?
- Would an automated reminder system for harvest schedules be beneficial to you?
- Would you prefer a dropdown-based selection for logging nest conditions, or would manual input be more convenient?
- Do you require a feature to generate reports on nest production over different time periods?
- Would integrating simple analytics feature help in planning future harvests based on past production trends?

Usability & Accessibility (System Design Preferences)

- Would a mobile-friendly web-based system improve data entry and accessibility for you?
- Do you prefer using a mobile application, a web-based system, or both for logging nest data?
- What level of technical experience do you have with digital tools and data entry systems?
- Would a system that allows offline data entry (with later synchronization) be useful for you?
- Would you find value in a feature that allows you to upload nest images for reference?

Appendix B: Stakeholder Swiftlet Bird House



Appendix C: Author Observations