CHUKA UNIVERSITY
FACULTY OF SCIENCE ENGINEEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE
EUGINE ONYANGO ODHIAMBO
EBI/61390/22
COSC 482
COMPUTER SYSTEM PROJECT 1
BERNARD ONG'ERA OSERO

TABLE OF CONTENTS

Contents

LIST OF TABLES

LIST OF FIGURES

Design and Implementation of a Smart Ranch Management System Using Flutter and Fast API

Abstract

The profitability and sustainability of smallholder livestock farming in Kenya are often constrained by reliance on manual or sporadic record-keeping, resulting in poor tracking of expenses, inefficient feed utilization, and limited visibility into individual animal health (Wambui, 2020). This project proposes the design and implementation of a Smart Ranch Management System (SRMS), delivered as a robust, cross-platform mobile application, to address these critical deficiencies. The system utilizes a modern, high-performance technology stack: Flutter for the frontend, ensuring broad accessibility across mobile operating systems; Fast API for the backend API, leveraging its asynchronous capabilities for superior concurrency and low latency; and PostgreSQL, selected for its advanced ability to support Hybrid Transactional/Analytical Processing (HTAP). The core innovation lies in the system's ability to seamlessly support both high-level batch management (for financial and aggregated analysis) and granular, real-time tracking of individual animals (for health and performance monitoring). Critically, the architecture is explicitly designed for future readiness, providing a robust, scalable foundation capable of integrating high-volume IoT sensor data and complex machine learning models for predictive analytics. By centralizing operational data, the SRMS aims to enable Kenyan farmers to transition to data-driven decision-making, thereby enhancing operational efficiency and bolstering resilience against structural challenges inherent in the digital transformation of smallholder agriculture (Kiaka, 2024).

Chapter One: Introduction

1.1 Background to the Study

The global agricultural sector is undergoing a profound digital transformation, often referred to as Agri Tech, characterized by the integration of information technology, sensors, and data analytics. This movement aims to introduce precision agriculture, optimizing resource consumption and boosting productivity through real-time monitoring and data-driven insights (Farmonaut, 2023).In the African and Kenyan context, the agricultural sector remains the economic backbone, yet it faces persistent structural challenges, including vulnerability to climate change and restricted access to timely market information and inputs (Mhlanga, 2024). Digital solutions are viewed as essential tools to overcome these barriers, enhancing financial inclusion and increasing income for smallholder farmers (Kiaka, 2024). Small-scale farming operations are critical to Kenya, accounting for approximately 78% of the country's total agricultural production (Anastasia Mumbi, 2024). Within this sector, livestock is particularly important, increasingly regarded as more resilient than crop farming in mitigating climate change shocks such as drought (Njarui, 2016).

However, the effective management of these livestock holdings is severely hampered by inadequate digital infrastructure and low adoption of sophisticated management tools (Mhlanga, 2024). Existing solutions often fail to account for the constraints faced by smallholders, particularly the diversity of mobile devices and the necessity for simple, intuitive interfaces (Anastasia Mumbi, 2024).This project recognizes that for a solution to be successful in the local market, it must adopt a mobile-first design philosophy, justifying the use of Flutter to ensure a robust, cross-platform experience across the prevalent mobile operating systems regardless of the user’s device specifications (Technology, 2021).

1.2 Problem Statement

Smallholder livestock management in Kenya is critically hindered by poor record-keeping practices. Evidence suggests that manual or biannual recording of business transactions is common, which leads directly to execution delays, inaccurate tracking of operational activities, and eventual business struggle or low profitability (Wambui, 2020). This results in a significant management vacuum where farmers lack the granular data necessary to make informed operational decisions. Specifically, farmers struggle to consistently track individual animal health status, accurately quantify feed consumption for efficiency analysis, and maintain detailed, categorized financial expenses. The inability to analyze these metrics prevents optimized resource allocation, breeding strategies, and timely interventions (WHO, 2017).

Furthermore, systemic issues compound this data deficit. The research indicates that qualified veterinary services are predominantly utilized by large commercial farms, while small-scale farmers often rely on unqualified health managers or forgo treatment entirely (WHO, 2017). This gap in professional extension services means that vital health events are often poorly documented or lost. The proposed system must therefore standardize and centralize the logging of symptoms, treatments, and outcomes. By enforcing structured data capture, the software creates a verifiable audit log, effectively generating the data required for future remote veterinary consultation or AI-aided diagnostics, mirroring successful models seen elsewhere in the continent (Marie McCampbell(PhD), 2023). The development of this structured, accurate data foundation is a necessary step to bridge the gap between smallholders and professional expertise, a precursor to enhancing animal welfare and productivity.

Finally, a technical gap exists in affordable software that can seamlessly handle the dual demands of livestock management: providing aggregated batch-level financial reports alongside detailed, longitudinal individual animal tracking (e.g., specific health history or breeding performance). Designing a robust database architecture to support this hybrid requirement without sacrificing performance is central to solving the current data management crisis.

1.3 Objectives of the Project

General Objective

The general objective is to design and implement a scalable, mobile-based Smart Ranch Management System using Flutter and Fast API to enhance operational efficiency, record-keeping accuracy, and profitability for smallholder livestock farmers in Kenya.

Specific Objectives

1. To analyze the functional and non-functional requirements for comprehensive livestock management, specifically focusing on supporting both batch and individual tracking within the Kenyan context.
2. To design and implement a robust, asynchronous API backend using the Fast API framework and a PostgreSQL hybrid data model capable of handling high-frequency transactional data and complex analytical queries efficiently.
3. To develop a cross-platform mobile application utilizing Flutter for intuitive recording of livestock details, tracking individual health events, monitoring feed consumption against batches, and logging financial transactions (expenses and income).
4. To implement a system architecture that is explicitly designed for future readiness, ensuring seamless integration pathways for IoT sensor data and machine learning predictive analytics models.
5. To rigorously test and evaluate the system's performance, usability, and functional completeness against the defined user requirements.

1.4 Scope and Limitations

Scope

The scope of this six-month project phase is clearly delineated:

1. **Functional Scope:** The system will encompass four essential modules: Livestock Inventory (supporting both individual identification and batch grouping), Health Tracking (events, treatments, diagnostics), Feed Management (consumption logs for batches), and Financial Tracking (categorized expenses and income).
2. **Technology Scope:** Development is strictly confined to the Flutter framework for the frontend application, the Fast API framework for the RESTful API backend, and the PostgreSQL relational database management system.
3. **Target User Base:** The system is designed for any livestock holders, typically those operating in peri-urban or rural Kenyan settings, and those managing fewer than 5 dairy cows or fewer than 10 small ruminants, as characterized in regional studies (WHO, 2017).

Limitations

1. **Phase I Limitation:** This Bachelor of Science project, Phase I, explicitly **excludes** the physical integration of actual IoT devices (e.g., smart collars or environmental sensors) and the final implementation of complex Machine Learning models for predictive analytics. The focus remains on building the fully functional, scalable architectural foundation.
2. **Data Limitation:** System validation and testing will rely on simulated and manually collected mock operational data, rather than large-scale, real-time datasets derived from a

continuous production environment.

3. **Hardware Limitation:** The project budget and timeframe preclude the procurement of specialized hardware (such as dedicated cloud servers or external data storage) beyond standard academic development resources. Performance optimization will thus concentrate entirely on the efficiency of the software stack.

1.5 Significance of the Study

The successful implementation of the Smart Ranch Management System holds substantial significance across several domains.

To Local Farmers: The most direct benefit is providing an accessible, mobile tool that facilitates the crucial shift from manual to digital record-keeping. This transition promises data-driven operational improvements, better resource utilization through detailed feed tracking, and significantly enhanced financial oversight, directly addressing the deficiencies that currently limit profitability.[1]

To Chuka University and Academia: The project serves as a practical demonstration of advanced application development, showcasing technical mastery in database design, high-performance API development, and cross-platform mobility. This aligns directly with the Faculty of Science Engineering & Technology's mission to generate and transmit quality knowledge through applied research and developing market-driven programs (University, n.d.).

To the Agri tech Sector: By intentionally designing an architecture that prioritizes scalability and future readiness, the project provides a crucial reference model for digitalizing livestock management across Sub-Saharan Africa (Mhlanga, 2024). The architectural spine, based on Fast API and PostgreSQL, explicitly addresses the known scalability gap associated with integrating high-volume IoT and ML in African aggrotech environments (Industries, 2023).

Furthermore, the selection of Flutter and Python/Fast API is a strategic advantage. These are highly marketable and in-demand technologies [10], ensuring I not only meets the academic goal of

demonstrating technical skill mastery (Lemayian, 2017) but also develops a tangible, industry-relevant product that maximizes the project's utility and employability post-graduation (University U. , 2024).

Chapter Two: Literature Review

2.1 Review of Existing Ranch Management Systems and Gap Analysis

A review of commercial ranch management software, such as ShambaPro–Rwanda, CattleMax – USA and AgriWebb–Australia, reveals a focus on core features including inventory tracking, herd health management, and performance analytics (Farmonaut, Top ranch & farm management software tools, 2023). While effective for basic management tasks, many of these systems often lack the necessary architectural spine to efficiently accommodate African AgriTech levels and the style or the extent of the farms in Kenya ,East Africa or Africa in its own.

A major analytical gap identified in the current software ecosystem, particularly when considering the future of African agriculture, is the failure to explicitly plan for the convergence of high-volume IoT telemetry and complex Machine Learning models (Bipesh Subedi, 2021). For a system to truly evolve toward Agriculture 5.0, it must be founded on an architecture that supports dynamic workloads and massive data ingestion (Mehendale, 2024). Existing systems ie ShambaPro often provide standard reporting capabilities (Farmonaut, Smart farming in Kenya: Boosting small scale agriculture, 2023) but may not detail the underlying structure required for achieving the massive elasticity necessary for real-time analytics.

A contextual gap also exists. Many advanced, subscription-based solutions are tailored for large-scale commercial operations in developed economies. These systems often overlook the key constraints of limited digital literacy and variable infrastructure availability faced by Kenyan smallholders and not just Kenyan but Africa at large (Mhlanga, 2024). This project, therefore,

addresses the critical requirement of building a foundational architecture that is natively ready for the future addition of real-time analytics and predictive modeling (Bernard Ijesunor Akhigbe, 2021). This strategic readiness, often overlooked in simpler management systems (Mehendale, 2024), is a necessary precondition for the long-term success of digitalization efforts in the region.

2.2 Theoretical Framework: Agile Software Development

This project will adopt the Agile software development methodology, specifically leveraging the Scrum framework, for managing the six-month development lifecycle. This approach, characterized by iterative and incremental delivery, is preferred over the traditional Waterfall model due to its flexibility and adaptability (GeeksforGeeks, 2023).

For a time-constrained academic project, Agile provides significant advantages: it prioritizes the delivery of working software as the primary measure of progress, allows for continuous feedback, and facilitates rapid adaptation to evolving requirements or unforeseen technical challenges that commonly arise in development (GeeksforGeeks, 2023). The project will utilize the standard five phases of the Scrum life cycle: Initiation, Planning & Estimates, Implementation (Sprints), Review & Retrospective, and Release (Workamajig, 2023). This iterative structure ensures that milestones are clearly defined and that the project remains on schedule for the final submission deadline (Dallas, 2024).

2.3 Review of Key Technologies

2.3.1 Frontend: Flutter for Cross-Platform Mobility

Flutter, Google's UI toolkit for building natively compiled applications for mobile, web, and desktop from a single codebase, has been strategically selected for the frontend.

The primary justification for this choice is the ability to deliver high-quality applications simultaneously for both Android and iOS devices using a single codebase (Technology, 2021).This maximizes market accessibility in Kenya, where the user base operates on diverse mobile platforms, while significantly reducing the development complexity and resource cost. Furthermore, Flutter's stateful hot reload feature dramatically shortens the development cycle, allowing the me–the developer to iterate on features and fix bugs at a higher velocity. In the context of a short, high-stakes academic project, this efficiency is non-negotiable, ensuring more time can be dedicated to rigorous testing, technical integration, and comprehensive documentation required for academic success.

2.3.2 Backend: Fast API for Asynchronous Performance and Scalability

Fast API is an async-first Python web framework designed from the ground up to support asynchronous programming using Python's asyncio library (Kanithkar, 2023). It is known for its high performance, often ranking among the fastest Python web frameworks available.

The choice of FastAPI is a critical architectural decision based on future scalability requirements. Since Phase II involves integrating high-volume Internet of Things (IoT) data, which consists of massive I/O-bound tasks (network requests, data ingestion), an asynchronous framework is fundamentally necessary. FastAPI's ability to handle non-blocking requests means that while it waits for an I/O operation (like writing sensor data to the database), it can process other incoming user or sensor requests, maintaining high concurrency and system elasticity (Industries, 2023). This proactive architectural selection ensures that the API layer built in Phase I will not become a bottleneck when scaling to handle high-frequency data streams from external devices. An additional benefit is the automatic generation of interactive API documentation (based on OpenAPI/Swagger), which streamlines the development process and simplifies future integration work (Team, 2023).

2.3.3 Database: PostgreSQL for Hybrid Data Processing (HTAP)

PostgreSQL is a powerful, mature, and extensible relational database management system that provides robust transactional integrity (OLTP). Its flexibility makes it ideal for supporting the hybrid data demands of ranch management.

The project requires simultaneously managing high-frequency transactional data (e.g., recording a specific animal's daily weight or vaccination—OLTP) and supporting large-scale analytical workloads (e.g., calculating quarterly Feed Conversion Ratios across the entire herd—OLAP). PostgreSQL can be configured for Hybrid Transactional/Analytical Processing (HTAP) to meet this challenge. This is achieved through advanced features such as Table Partitioning, which segments large historical tables (like feed logs) based on time or batch ID to optimize analytical query performance, and Materialized Views, which precompute complex, slow-running aggregation queries, providing rapid access to batch-level performance metrics (Hasib, 2023). By employing these strategies, the system ensures both granular data accuracy and aggregated reporting performance without necessitating a separate, complex data warehouse setup.

Chapter Three: Methodology and System Analysis

3.0 Introduction

This chapter details the specific research techniques and the **System Development Life Cycle (SDLC)** model adopted to manage the **Smart Ranch Management System (SRMS)** project. It outlines the structured processes for data collection, project execution, and quality control. The methodology is designed to balance the rigorous academic requirements for documentation with the need for flexibility, speed, and continuous validation, ensuring the final system is both technically sound and operationally relevant to Kenyan smallholder ranching

3.1 Project Methodology

The development of the SRMS will be governed by the **Agile-Scrum Model**. This methodology is selected for its **iterative** and **incremental** nature, which is highly suited for software projects where user requirements may evolve during the development process. Scrum facilitates **rapid prototyping**, continuous feedback loops, and flexible adaptation to ensure the final product—the SRMS—is robust, highly functional, and precisely meets the complex needs of modern ranch management. Although Agile prioritizes working code, the academic requirement for detailed documentation necessitates a strong emphasis on documentation and internal code comments at every stage, fulfilling university marking criteria (Lemayian, 2017).

3.2 Scrum Application and Phasing

The entire development life cycle will be organized into a series of **Sprints**, each lasting **two weeks**. The key Scrum phases utilized will be:

- **Initiation:** Defining the overall project vision and compiling the **Product Backlog** (the master list of required features).
- **Sprint Planning:** At the start of each two-week cycle, features are pulled from the Product Backlog to create the **Sprint Backlog**.

- **Implementation (Sprints):** The actual development and testing of features in time-boxed cycles.
- **Sprint Review:** A formal session at the end of each sprint to demonstrate completed, working software to the supervisor/stakeholder proxy for feedback.
- **Retrospective:** An internal meeting to continuously improve the development process, focusing on technical debt and efficiency.

This iterative process ensures that the project remains within the academic timeframe while prioritizing the delivery of tested, functional software.

3.3 Data Collection and Validation Techniques

Requirements for the SRMS were gathered and validated using a mixed-method approach, combining primary and secondary research techniques:

Table 1:Data Collection and Validation Techniques

Data Collection Method	Purpose	Application to SRMS
Primary Data: Semi-Structured Interviews	To understand direct pain points, existing manual processes, and required workflows.	Targeted interviews with local smallholder ranch managers and veterinary practitioners to define key functional requirements (e.g., specific health record formats, expense categories).
Secondary Data: Desk Research	To establish industry best practices and identify architectural precedents.	Review of academic literature on Precision Livestock Farming and analysis of commercial LMIS systems (as detailed in Chapter 2) to validate feature sets and identify scalability gaps.
Observation	To capture constraints and user habits in the target environment.	Observing existing manual record-keeping habits (e.g., use of logbooks or simple spreadsheets) to inform the Usability (NFR) design goals for the mobile application.

3.4 Resources and Budget

This section outlines the essential resources required for the successful development and deployment of the Smart Ranch Management System (SRMS) and estimates the associated costs.

The SRMS development utilizes a stack of open-source and cloud-native technologies, minimizing capital expenditure and ensuring cost-effectiveness.

3.4.1 Required Resources

The project requires two primary categories of resources: Software (Technical Stack) and Hardware/Infrastructure.

3.2 Requirements Analysis

The requirements are defined based on the need to solve the core record-keeping deficiencies and provide a platform for future scalability.

Functional Requirements (FRs)

1. **FR1: Inventory Management:** The system must allow for the creation, editing, and retrieval of detailed records for individual animals (ID, breed, age, sex, acquisition cost) and the organization of these individuals into defined Batches or Herds.
2. **FR2: Expense Tracking:** The application must enable users to input categorized financial expenses (e.g., feed, medication, labor) and link them accurately to either a specific Batch, an Individual Animal, or the general Farm Account.
3. **FR3: Feed Consumption Tracking:** The system must allow users to log the type, quantity, and cost of feed consumed by designated Batches over defined time periods to facilitate the calculation of Feed Conversion Ratios (FCR).
4. **FR4: Health Monitoring:** The application must support the chronological recording of health events for individual animals, including symptom observations, diagnosis, treatment

administered, and final outcome, providing a clear health audit trail.

5. **FR5: Reporting:** The system must generate analytical summaries, including financial performance reports (net expenses per batch or per category) and herd performance metrics (FCR, mortality rates).

Non-Functional Requirements (NFRs)

1. **Usability (NFR1):** The mobile application interface must be simple, intuitive, and designed to accommodate users with potentially moderate digital literacy levels, ensuring high adoption rates among smallholder farmers.
2. **Performance (NFR2):** The Fast API must demonstrate high responsiveness and efficient handling of concurrent user requests, leveraging its asynchronous capabilities to minimize latency.
3. **Security (NFR3):** Standard security practices, including secure user authentication, authorization, and data encryption, must be implemented to protect sensitive farm data.
4. **Scalability (NFR4):** The underlying architecture must possess the elasticity and modularity to integrate additional high-volume services (e.g., IoT data streams and ML endpoints) without requiring a complete redesign.

3.3 Technology Stack Justification Synthesis

The selected technology stack represents a balanced approach, prioritizing rapid development (Flutter), high performance (Fast API), and robust data management (PostgreSQL) while maintaining direct relevance to the operational environment in Kenya.

Component	Technology	Technical Advantage	Contextual Relevance (Kenya)
Frontend	Flutter	Cross-platform capability from a single codebase, delivering native performance and reducing complexity.	Crucial for maximizing reach across diverse mobile devices utilized by Kenyan farmers; speeds up development time for a 6-month project.
Backend API	Fast API	Async-first architecture, enabling high concurrency and superior I/O handling.	Provides the necessary low-latency, resilient foundation for handling future high-frequency IoT sensor data ingestion (Phase II).
Database	PostgreSQL	Robust transactional integrity (OLTP) and advanced features for Hybrid Transactional/Analytical Processing (HTAP).	Supports reliable granular tracking (Individual animal records) and complex, rapid analytical reporting (Batch profitability analysis) simultaneously.

Table 2:Technical Stack Justification and Contextual Relevance

Chapter Four: System Design

4.0 Introduction

The system design phase translates the established requirements into a detailed technical blueprint for implementation. This phase is critically important as it defines the system's structure, behavior, and interactions, ensuring that the final product is robust, maintainable, and scalable. A well-considered design mitigates development risks, prevents costly architectural changes during coding, and serves as a precise guide for developers. This chapter details the architectural patterns, process flows, and data models that form the foundation of the Smart Ranch Management System (SRMS).

4.1 System Architecture Overview

The Smart Ranch Management System employs a classical Three-Tier Architecture, designed for modularity and scalability.

1. **Presentation Tier (Flutter Mobile Application):** This tier handles all user interaction, data input forms, and visualization of reports. It communicates exclusively with the Application Tier via RESTful HTTP requests.
2. **Application Tier (FastAPI Backend API):** This is the middleware layer, responsible for implementing all business logic, data validation, user authentication, and serving data endpoints. It utilizes an asynchronous Object-Relational Mapper (ORM) to communicate efficiently with the database, maximizing the non-blocking nature of the FastAPI framework.
3. **Data Tier (PostgreSQL Database):** This tier provides persistent storage and is structured to support both rapid transactional writes and complex analytical reads, leveraging its HTAP capabilities.

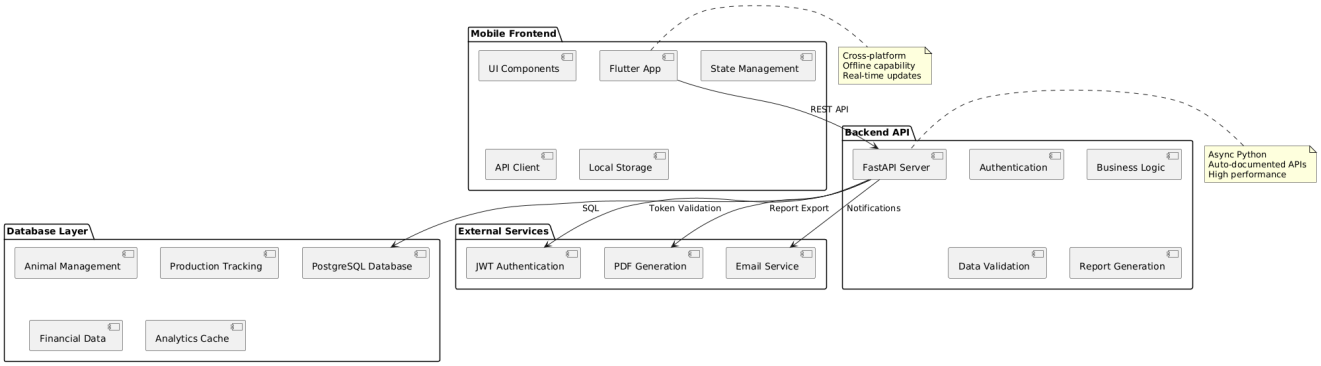

Figure 1:System architecture components diagram

4.2 Database Design: The Hybrid Data Model (Individual vs. Pen)

The database schema, implemented in PostgreSQL, is designed specifically to handle the co-existence of batch-level aggregation and individual animal granularity.

4.2.1 Relational Structure

The core structure utilizes a relational design centered on two main entities: PEN and ANIMAL.The PEN table holds aggregated details (e.g., Pen ID, primary livestock type, aggregated purchase cost). The ANIMAL table contains unique identifiers (Animal_ID), detailed individual metadata (e.g., DOB, specific breed traits), and a foreign key linking it directly to its corresponding PEN_ID. Here is an ER Diagram to show this in pictorial form.

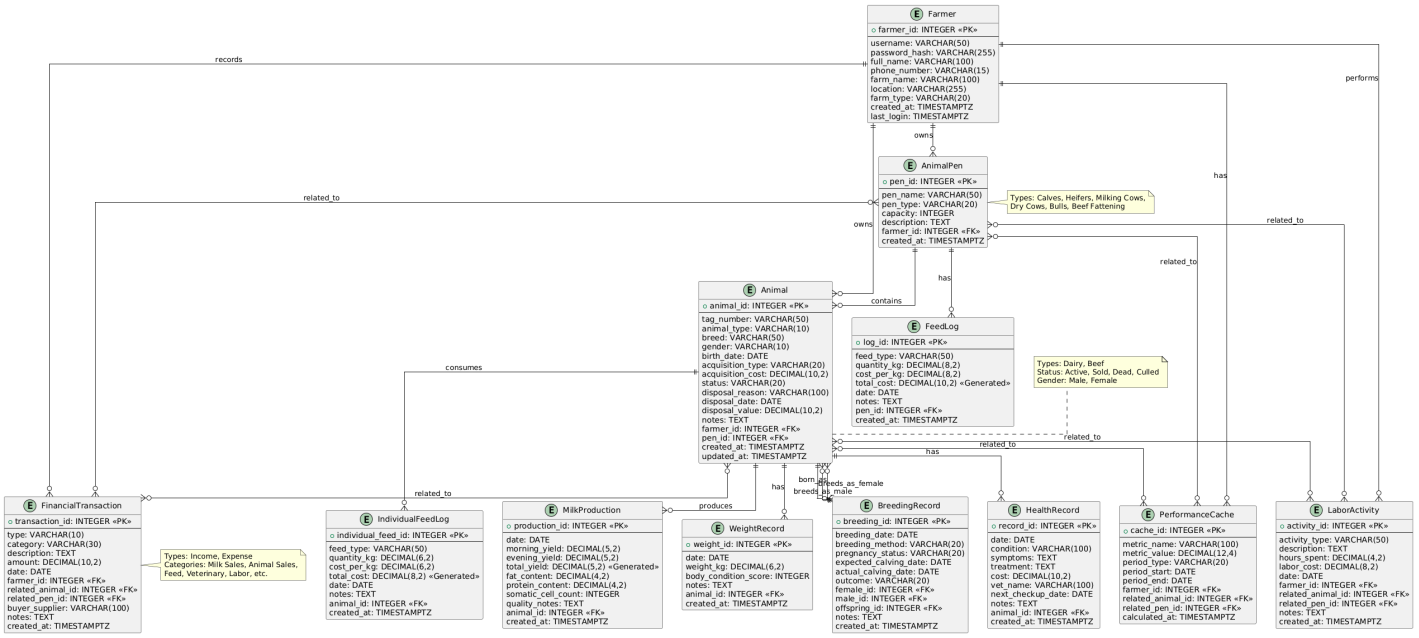

Figure 2:ER Diagram

4.2.2 Supporting Transactional and Analytical Workloads

1. **Individual Tracking (OLTP):** High-frequency or event-driven data, such as records in the HEALTH_EVENTS table (vaccinations, treatments) or detailed, daily FEED_LOGS, are linked explicitly to the unique ANIMAL_ID. This ensures that every piece of data maintains transactional integrity and can be traced to a specific animal identity, providing a clear audit trail essential for health monitoring. Here are different Er diagrams to demonstrate these properties well in a diagram.

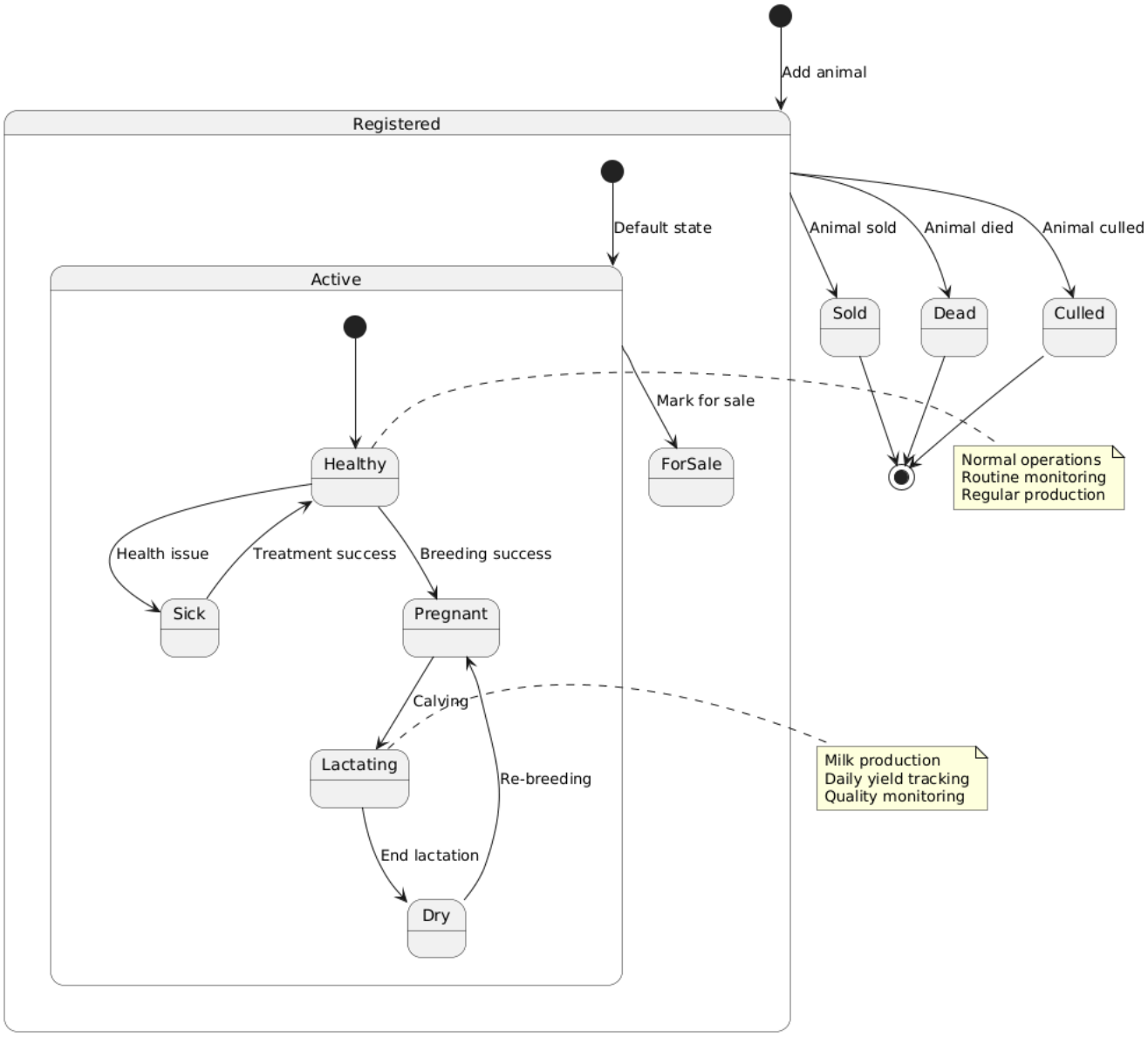

Figure 3:Animal life expectancy state diagram

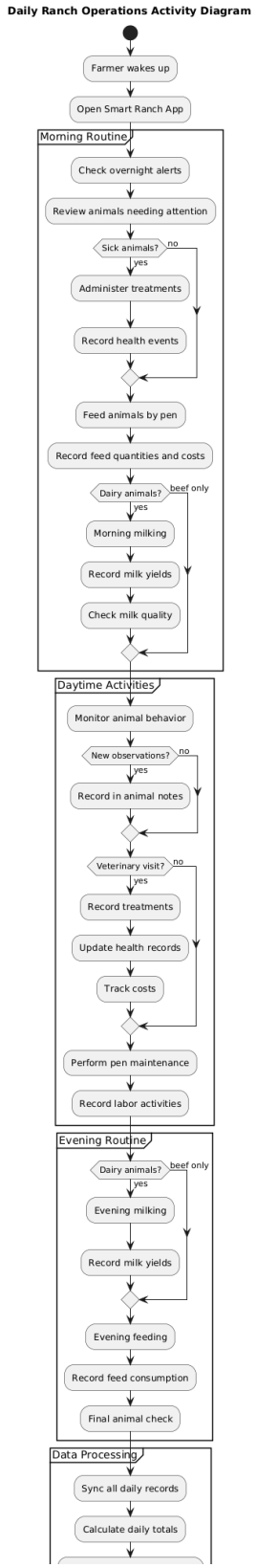

Figure 4:Daily Ranch operation activity diagram

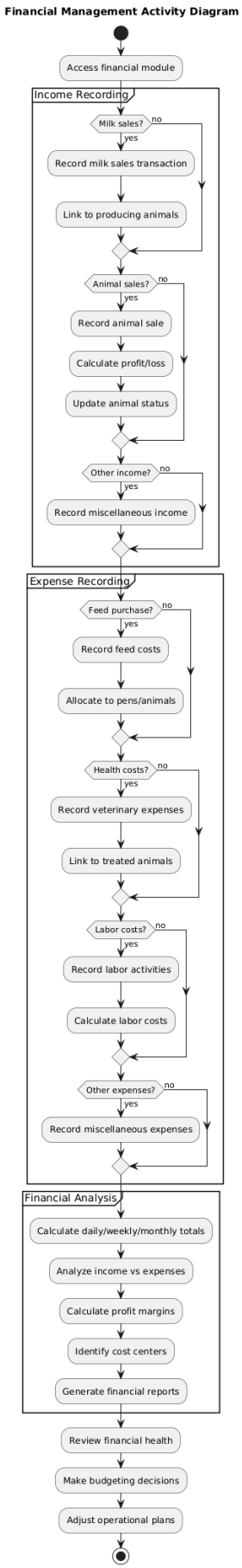

Figure 5:Financial management activity diagram

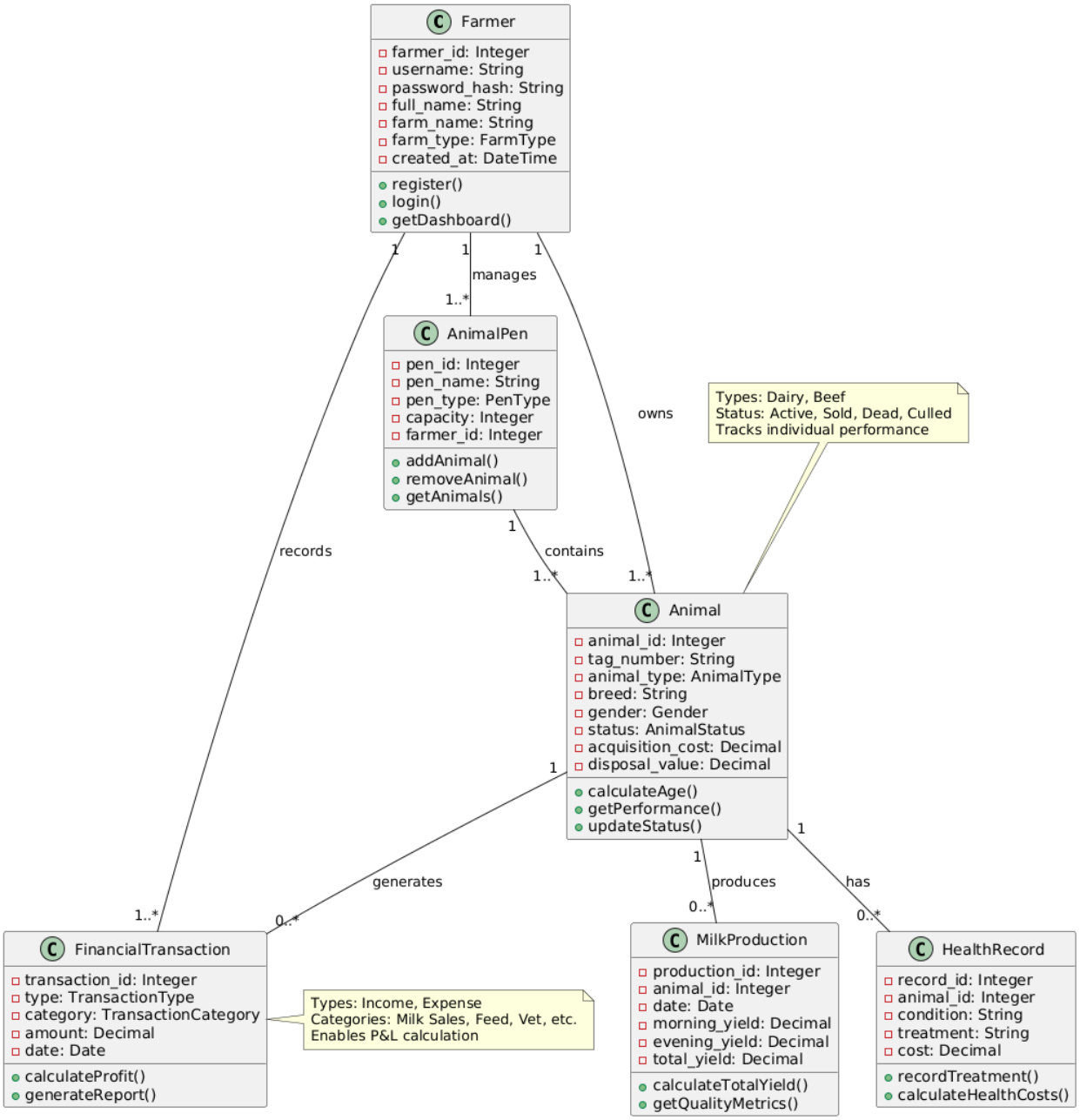

Figure 6:Class diagram

2. **Batch Tracking (OLAP via HTAP):** To provide fast, aggregated reporting, PostgreSQL's

HTAP features are utilized.

- **Materialized Views:** Complex analytical metrics (e.g., average Feed Conversion Ratio per pen, total veterinary expenses per month) are calculated using SQL aggregations and stored in Materialized Views. These views are refreshed on a schedule, offering near-instant query responses for pen performance reports without needing to re-calculate extensive underlying data every time the report is generated.
- **Partitioning:** High-volume, historical tables, such as the FEED_LOGS or generalized EXPENSES, will be partitioned by time (e.g., quarterly) or by Batch ID. This technique limits the database engine's scan area when executing long-term analytical queries, drastically improving performance and efficiency for historical analysis.

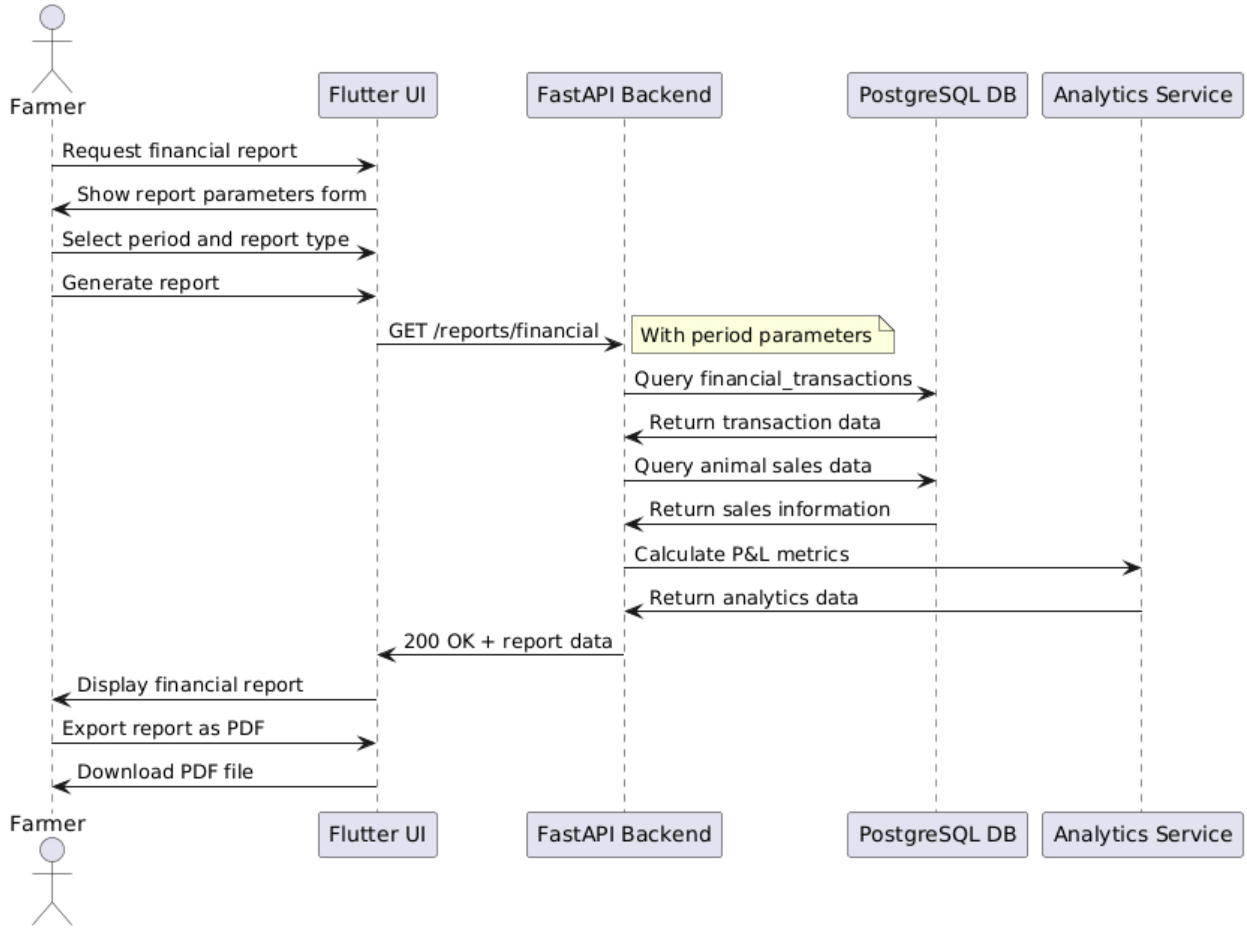

Figure 7:General financial report sequence diagram

4.3 Module Design Details

1. **Livestock Module:** Manages the creation and grouping of animals. Data input is structured

to ensure mandatory fields are completed, guaranteeing data quality necessary for later analysis.

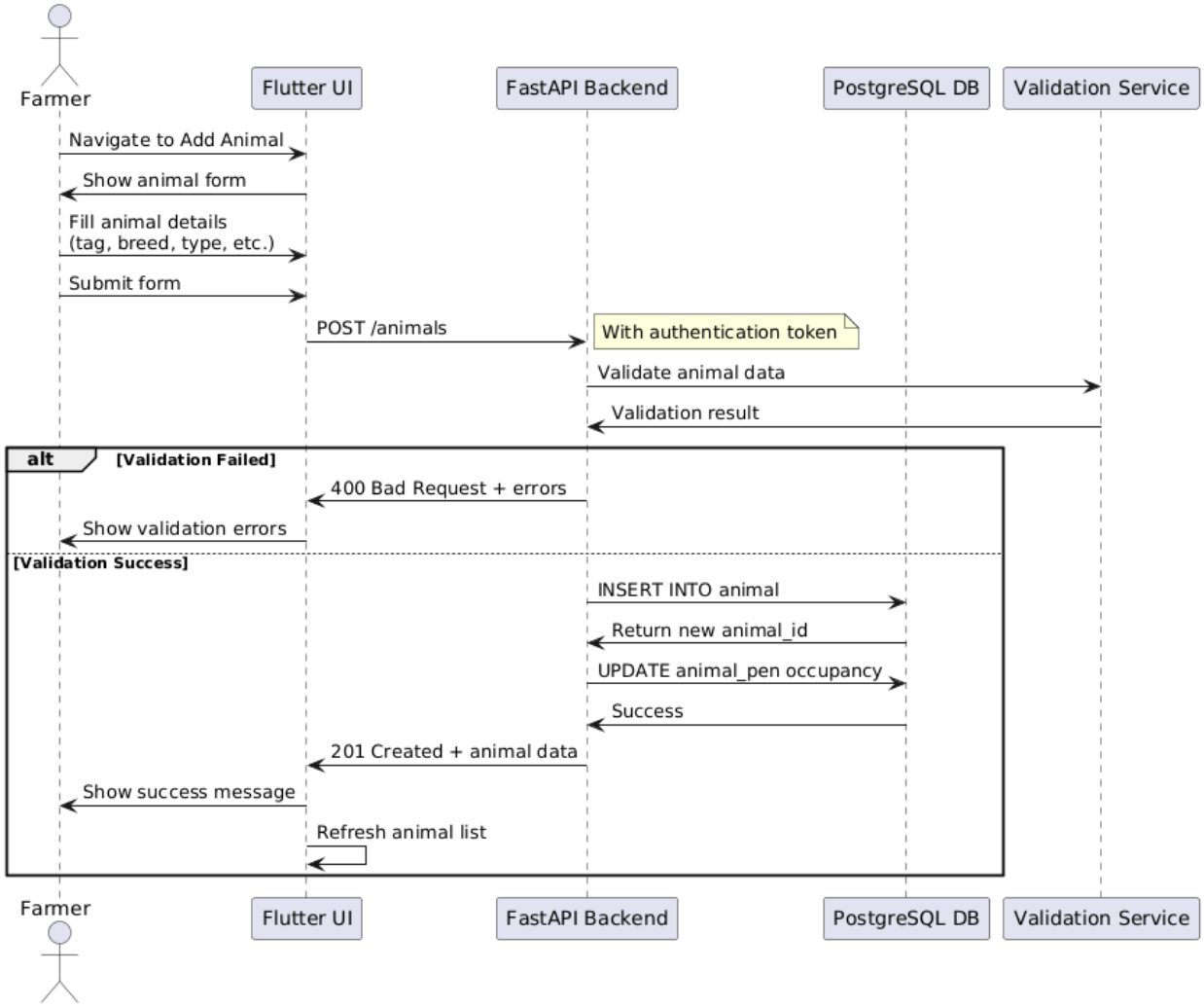

Figure 8:Add new animal sequence diagram

2. **Expense Module:** Provides forms for itemized cost entry. Entries are systematically categorized and linked via foreign keys to the appropriate PEN_ID or general farm account, providing the basis for net profitability analysis.
3. **Health Module:** This module enforces standardized data entry. Instead of free text, it utilizes

predefined lists for common symptoms, diagnoses, and treatments. This structured input simplifies the user experience for farmers and, critically, improves the consistency and quality of the data, making the health logs directly applicable for future integration with machine learning diagnostic models.

4. **Feed Module:** Tracks the consumption of various feed types by batches over time, generating the raw data necessary for calculating Feed Conversion Efficiency metrics.

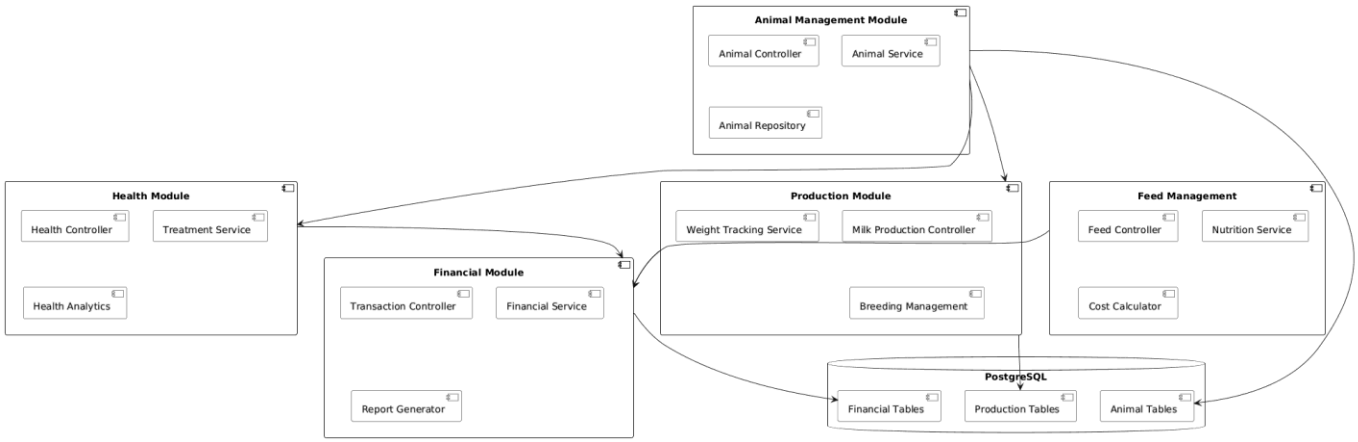

Figure 9:Module component diagram

4.4 Scalability and Future Readiness Design

The architectural choices made in Phase I are deliberately positioned to support significant expansion in Phase II, particularly the planned integration of IoT and Machine Learning.

4.4.1 Architectural Separation for ML Integration

FastAPI's structure naturally facilitates high scalability through microservices. Future Machine Learning models (such as predictive disease risk scoring or optimal resource allocation algorithms) can be deployed as separate, resource-intensive service containers. These services would consume clean data prepared by the PostgreSQL database, exposing their predictive results via internal APIs

which the main FastAPI application tier can then call. This separation ensures that the core application remains lean and operational, while the heavy computational demands of ML processing are handled independently, maintaining system elasticity (Industries, 2023).

4.4.2 IoT Data Ingestion Strategy

The adoption of FastAPI's asynchronous nature provides the fundamental performance capability required for handling high-volume I/O-bound tasks (Kanithkar, 2023). In Phase II, a dedicated data ingestion microservice, potentially utilizing an MQTT broker (a protocol standard for IoT), would receive raw telemetry (e.g., animal location, vital signs) from sensors. The FastAPI backend is designed to integrate seamlessly with this dedicated ingestion service layer, allowing the system to scale its data processing capacity dynamically without needing to modify the user-facing business logic layer (Isaac, 2021). This guarantees that the current design does not create a performance bottleneck for future sensor integration.

The underlying Python/FastAPI and PostgreSQL stack is highly compatible with modern cloud infrastructure, allowing the system to leverage cloud-native features such as load balancing and containerization (e.g., Docker or Kubernetes) for handling dynamic workloads efficiently.[15]

Chapter Five: Expected Outcomes, Project Schedule, and Conclusion

5.0 Expected Outcomes and Deliverables

Upon successful completion of the project, the following deliverables will be produced:

5.1 Software Deliverables (Working System)

1. **Smart Ranch Mobile Application:** A fully functional, tested cross-platform mobile application developed using Flutter, covering the Inventory, Health, Feed, and Finance modules.
2. **RESTful API Backend:** A robust, asynchronous API developed using FastAPI, providing secure and high-performance data interaction.
3. **Database System:** A fully implemented PostgreSQL database featuring the optimized hybrid data model, including partitioned tables and Materialized Views for HTAP capability.

5.2 Academic and Documentation Deliverables

1. **Formal Project Proposal:** The current document, submitted for supervisory approval.
2. **Comprehensive Technical Documentation:** Including detailed system design documents, Entity-Relationship Diagrams (ERDs), data flow charts, and API specifications.
3. **Source Code:** A documented source code repository with comprehensive inline comments and version control, satisfying academic criteria for technical mastery.

4. **Final Project Report:** A detailed, bound report summarizing the research, methodology, implementation, and evaluation findings.
5. **Final Presentation and Demonstration:** A working demonstration of the system's functional completeness and usability for the university examination board.

5.3 Project Schedule

The project schedule is structured into five major phases, adhering to the iterative principles of the Agile/Scrum methodology, ensuring key academic milestones are met promptly.[25]

Project Schedule: Six-Month Agile Implementation Plan

Month	**Phase (Scrum)**	**Key Activities**	**Deliverables / Milestones**
Month 1 (Weeks 1-4)	Initiation & Planning	Detailed requirements elicitation, formal design analysis (ERD, Architecture diagrams), setup of version control and development environment; Supervisor consultation.	**Formal Proposal Submission**; Detailed Functional Specification Document; Initial Product Backlog.
Month 2 (Weeks 5-	Sprint 1: Core	Environment setup	Working Core Data

8)	Foundation	(FastAPI, PostgreSQL); implementation of the database schema (hybrid model); API development for core CRUD operations (Animal, Batch, User Auth).[31]	Model; Proof-of-Concept Backend API (CRUD functions tested).
Month 3 (Weeks 9-12)	Sprint 2: Frontend & Financials	Flutter UI scaffolding and design implementation; integration of user authentication; development and linking of the Expense Tracking module (data input and display).	Integrated Alpha Prototype (Inventory and Finance modules complete); Mid-Sprint Review.
Month 4 (Weeks 13-16)	Sprint 3: Core Features	Development and integration of the Health Monitoring module; implementation of the Feed Consumption Tracking module;	Integrated Beta Prototype with 80% feature completeness; Mid-Project Review presentation.

		deployment and testing of complex HTAP queries (Materialized Views).	
Month 5 (Weeks 17-20)	Sprint 4: Testing & Polish	Comprehensive unit and integration testing; User Acceptance Testing (UAT) using simulated ranch data; performance tuning of async API; final UI/UX polishing and robust error handling.	Tested System (Release Candidate); Comprehensive Test Report.
Month 6 (Weeks 21-24)	Release & Documentation	System hardening and final packaging; compilation of detailed system documentation, user manuals, and the Final Report; preparation of the demonstration environment.	**Submission of Final Working System**; **Submission of Final Bound Project Report**.

5.4 Conclusion and Future Work (Phase II Roadmap)

5.4.0 Conclusion

This project proposes a strategic and timely intervention into the structural challenges facing Kenyan smallholder livestock management. By addressing the critical problem of poor record-keeping through a dedicated, mobile-first application, the system is poised to empower farmers toward data-driven profitability. The foundational choices of Flutter, FastAPI, and PostgreSQL are not merely arbitrary technology selections but calculated architectural decisions. They establish a high-performance, asynchronous backend capable of handling both necessary transactional integrity and complex analytical workloads (HTAP), while guaranteeing that the system is ready to seamlessly adapt to the future demands of IoT and Machine Learning, essential steps for the long-term digitalization of agriculture in the region.

5.4.1 Future Work (Phase II Roadmap)

The successful completion of Phase I establishes a robust platform for advanced development. The subsequent roadmap for Phase II would include:

1. **IoT Integration:** Implementing a dedicated MQTT protocol handler within the FastAPI service layer to reliably ingest high-frequency, low-latency data streams from external IoT devices (e.g., animal collars for GPS tracking and vital signs monitoring).[15]
2. **Machine Learning Integration:** Developing and deploying dedicated ML models (e.g., deep learning algorithms for anomaly detection) as microservices, using the structured health and feed data collected in Phase I to provide predictive analytics, such as forecasting disease risk or optimizing breeding schedules.
3. **Advanced Analytics and Geo-Spatial Tracking:** Extending PostgreSQL HTAP capabilities to incorporate PostGIS for handling spatio-temporal data, allowing for the analysis of animal movement patterns and grazing optimization.

Bibliography

Anastasia Mumbi, J. B. (2024). Characterization of small-scale farmers and assessment of their access to crop production information in selected counties of Kenya. *Scientific Research*. Retrieved from https://www.scirp.org/journal/paperinformation?paperid=133599

Bernard Ijesunor Akhigbe, K. M. (2021). IoT technologies for livestock management: A review of present status, opportunities, and future trends. *MDPI, 5*(1).

Bipesh Subedi, G. S. (2021). *Smart agriculture: Components, processes, challenges, and future perspectives.* ResearchGate.

Dallas, U. o. (2024). *UTDesign Students & Advisors*. Retrieved from Project schedules: https://utdesignstudents.utdallas.edu/resources/project-schedule/

Farmonaut. (2023). *Smart farming in Kenya: Boosting small scale agriculture*.

Farmonaut. (2023). *Top ranch & farm management software tools*. Retrieved from Farmonaut: https://farmonaut.com/blogs/ranch-management-software-7-powerful-tools-for-efficiency

GeeksforGeeks. (2023). *Agile software development - Software engineering*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/software-engineering/software-engineering-agile-software-development/

Hasib, S. W. (2023). PostgreSQL hybrid transactional/analytical processing using partitioning and materialized views. *Medium*. Retrieved from https://medium.com/@wasiualhasib/postgresql-hybrid-transactional-analytical-processing-using-25292f106239

Industries, P. (2023). *IoT scalability: What it means, common challenges, and how to scale effectively*. Retrieved from Particle: https://www.particle.io/iot-guides-and-resources/iot-scalability/

Isaac, J. O. (2021). *IOT - LIVESTOCK MONITORING AND MANAGEMENT SYSTEM.* RearchGate. Retrieved from https://www.researchgate.net/publication/352379751_IOT_-_LIVESTOCK_MONITORING_AND_MANAGEMENT_SYSTEM

Kanithkar, B. (2023). FastAPI vs Django: Async capabilities. *Medium*. Retrieved from https://medium.com/@kanithkar_baskaran/fastapi-vs-django-async-capabilities-bdf2a358c0a8

Kiaka, A. N. (2024). *Digital technology in Kenyan agriculture: Challenges and opportunities.* Institute for Poverty, Land and Agrarian Studies (PLAAS). Retrieved from https://plaas.org.za/wp-content/uploads/2024/05/Working-Paper-67-Digital-Technology-in-Kenyan-Agriculture-Kiaka.pdf

Lemayian, D. (2017). Advice to Kenyan computer science students on final year projects. David Lemayian Blog. Retrieved from https://davidlemayian.com/blog/2017/05/27/advice-to-kenyan-cs-students/

Marie McCampbell(PhD), U. I. (2023). Community conversation: Digital innovations for livestock development. Livestock Data Initiative.

Mehendale, P. (2024). Scalable architecture for machine learning applications. *Journal of Scientific and Engineering Research*(8), 111-117.

Mhlanga, D. (2024). *Digital revolution in African agriculture.* ResearchGate. Retrieved from https://www.researchgate.net/publication/378222763_Digital_Revolution_in_African_Agriculture

Njarui, D. M. (2016). A comparative analysis of livestock farming in smallholder mixed crop-livestock systems in Kenya: 1. Livestock inventory and management. *Livestock Research for Rural Development, 28*(4). Retrieved from

https://lrrd.cipav.org.co/lrrd28/4/njar28066.html
Team, J. P. (2023). Django vs. FastAPI: Which is the best Python web framework? *JetBrains*. Retrieved from https://blog.jetbrains.com/pycharm/2023/12/django-vs-fastapi-which-is-the-best-python-web-framework/
Technology, B. (2021). *Why use Flutter? Cross-platform app development made easy*. (B. Technology, Producer) Retrieved from https://www.bacancytechnology.com/blog/why-use-flutter-app-development
University, C. (n.d.). *Faculty of Science & Technology*. Retrieved from Chuka University: https://www.chuka.ac.ke/faculty-of-science-engineering-technology/
University, U. (2024). *Best short-term courses after BSc computer science in 2025*. Retrieved from UPES: https://www.upes.ac.in/blog/computer-science/short-term-courses-after-bsc-computer-science
Wambui, S. &. (2020). *Effect of record keeping practices on dairy farming business performance in Kiambu County.* ResearchGate. Retrieved from https://www.researchgate.net/publication/391436539_EFFECT_OF_RECORD_KEEPING_PRACTICES_ON_DAIRY_FARMING_BUSINESS_PERFORMANCE_IN_KIAMBU_COUNTY
WHO. (2017). Urban livestock keeping in the city of Nairobi: Diversity of production systems, supply chains, and their disease management and risks. *BMC Public Health, 17*.
Workamajig. (2023). *What are the phases of Scrum?* Retrieved from Workamajig: https://www.workamajig.com/blog/scrum-methodology-guide/scrum-phases