

Ministry of Higher Education, Research and Innovation

**THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE**

**Specialty:** Computer Science

|  |
| --- |
| **Topic: Design of a mobile application for breeders** |

Presented and publicly defended:

**Zida Assomption Désiré Wendlasida**

**Academic year: 2023 – 2024**

**Supervised by**: Dr Pegdwendé N. Sawadogo

**Under the supervision by:** Dr Pegdwendé N. Sawadogo

I dedicate this project to my parents and my brother for their unwavering support and constant love. Their encouragement and belief in me have been essential throughout this journey. Thank you for being my source of strength and inspiration.

**Acknowledgements**

First and foremost, I wish to express my profound gratitude to God for granting me the health and strength necessary to complete these three years of work. Without His grace, this journey would not have been possible. I also thank my parents for their unwavering support, constant encouragement, and unconditional love throughout this adventure. You have been my greatest source of inspiration and motivation. My thanks also go to my thesis supervisor, Mr. Sawadogo, for his guidance, valuable advice, and availability throughout this project. Your expertise and kindness have been invaluable assets in the completion of this work. I am grateful to the BIT school for the quality education I received there, and to the department head for his commitment and leadership. Thank you for providing us with an environment conducive to learning and personal growth. I would also like to thank my classmates, with whom I have shared so many moments of work, doubt, and success. Together, we have overcome the challenges of these years of study, and I look back on our journey with pride. Finally, I thank all my loved ones who contributed in one way or another to this project. Your support, understanding, and encouragement have been of great help throughout this process.

**Summary**

AnimalCare is an innovative mobile application designed for comprehensive management of pets and livestock. Created for breeders, pet owners, and animal enthusiasts, the app offers a range of essential features to ensure the well-being and care of animals. AnimalCare allows users to track the health status of their animals by recording information such as species, age, vaccination dates, and care history. Users can easily add new animals and update their information at any time. The app also includes a sales section where breeders can showcase their animals and related products for sale through a dedicated platform, featuring detailed listings with descriptions, prices, and photos. AnimalCare also facilitates connections between breeders, customers, and animal lovers through a chat feature that enables real-time discussions for sales, service exchanges, or breeding advice. Users can schedule and track their animals' care, organize vaccination sessions, and receive reminders for upcoming care tasks. With an intuitive interface and modern design, the app offers easy navigation and a pleasant user experience, enhanced by animations and visual effects. AnimalCare aims to transform animal management by simplifying daily tasks and centralizing crucial information in a single user-friendly application.

**TABLE OF CONTENTS**

Acknowledgements i

Summary 3

Part I: Theoretical Study 8

Chapter 1: General Information on Livestock Management 9

I.Study Context 9

II.Problem Statement 9

III. Objectives 9

1. General Objective

2. Specific Objectives 10

IV. State of the Art 10

1. The Concept of Health and Well-being of Livestock 10

2. Well-being of Livestock 11

3. Importance of Management for Quality and Productivity 11

4. Review of Existing Studies on Integrated Livestock Platforms 12

5. Improvements in Management and Productivity 12

V. Enhancing Animal Health and Well-being 13

1. Disease Prevention 13

2. Efficiency, Time Savings, and Access to Advice for Breeders 14

3. Analysis of Existing Applications 14

VI. Conclusion 15

Chapter 2: Methodologies and Conceptual Approaches 16

I. Introduction to UML 16

1. Definition 16

2. Object-Oriented Design 17

II. UML Modeling Language 17

1. Class Diagram 17

2. Sequence Diagram 19

3. Activity Diagram 19

III. Application of UML Diagrams in Development 20

1.Process and Development Method 20

IV. Conceptual Methodological Approach 22

V. Conclusion 24

Part II: Design and Implementation 25

Chapter 1: Design, Modeling, and Conceptual Approach 26

I. Design and Modeling 26

1. Use Case Diagram 26

2. Textual Description of the Animal Management Use Case 27

3. Sequence Diagram 27

4. Class Diagram 28

II. Conceptual Approach 30

III. Conclusion 32

Chapter 2: Implementation and Results 33

I. Implementation 33

1. Work Environment 33

a. Hardware Environments 33

b. Software Environment 34

2. Characteristics of Environments34

II. Results 36

Conclusion 40

bibliography41

**List of Acronyms and Abbreviations**

UML: Unified Modeling Language

OO: Object-Oriented Design

RAD: Rapid Application Development

JIT: Just-In-Time (Compilation)

AOT: Ahead-Of-Time (Compilation)

SCRUM: Agile Project Management Method

VS Code: Visual Studio Code

Firebase: Mobile and Web Application Development Platform

NoSQL: Non-relational SQL (Database Management System)

SQL: Structured Query Language

**List of Tables**

Table 1: Comparison of Development Methods 38

Table 2: Scrum vs RAD 38

Table 3: Use Cases vs User Stories 38

Table 4: Textual Description of the Animal Management Use Case 38

Table 5: Flutter vs React Native 38

Table 6: Firestore vs MySQL 38

Table 7: Characteristics of the Methods Used 38

**List of Figures**

Figure 1: AgriWebb Logo 15

Figure 2: Moocall Logo 15

Figure 3: UML Logo 17

Figure 4: Class Diagram Illustration 19

Figure 5 : Sequence Diagram Illustration 20

Figure 6 : Activity Diagram Illustration 20

Figure 7 : Application Use Case Illustration 27

Figure 8: Sequence Diagram for Product Purchase 18

Figure 9: AnimalCare Project Class Diagram 38

Figure 10: Loading Screen at Application Entry 38

Figure 11: Home Screen (Catalog) 39

Figure 12: Application Login Screen 39

Figure 13: Application Register Screen 39

Figure 14: Vaccine Management 39

Figure 15: Adding Vaccines 39

Figure 16: Adding Publication 39

**General Introduction**

The field of animal husbandry plays a crucial role in the economy and food security of many regions around the world. However, effectively managing animals, their health, reproduction, and products remains a significant challenge for many farmers, especially those operating in resource-limited environments. With the advancement of digital technologies, it is increasingly evident that innovative solutions can be developed to address these challenges. It is in this context that AnimalCare is positioned, a mobile application dedicated to the comprehensive management of livestock.

AnimalCare is designed to provide farmers with a comprehensive and intuitive platform to monitor the health status of their animals, plan vaccination sessions, manage animal-derived products, and facilitate the sale of these animals or products. The application also offers features to optimize interactions between farmers and clients, thus contributing to improved efficiency and profitability of operations.

This document presents the process of designing, developing, and deploying the AnimalCare application. It covers the various modules of the application, the technological choices made, the challenges encountered, and the solutions provided throughout development. This project is part of a broader vision of modernizing the livestock sector by leveraging mobile technologies to enhance daily animal management and connectivity between sector stakeholders.

Additionally, AnimalCare integrates with cloud computing and real-time database technologies, ensuring that user data is accessible from anywhere securely. This connectivity also allows farmers to collaborate with other sector stakeholders, access online advisory services, and stay informed about the latest trends and innovations in livestock management. By combining advanced features with a simple and user-friendly interface, AnimalCare aims to transform the way farmers manage their daily operations.

**Part I: Theoretical Study**

**Chapter 1: General Information on the Livestock Sector**

* 1. **Study Context**

The livestock market is currently experiencing rapid growth, characterized by increased demand for diverse services. This demand encompasses animal health and hygiene care, as well as a variety of products and professional advice tailored to the needs of farmers. This phenomenon is driven by a growing awareness of the importance of animal welfare and a strong consumer demand for quality and traceability of animal products.

In this context, farmers face major challenges in meeting these requirements while effectively managing their daily operations. They need solutions that not only facilitate the management of animal health and welfare but also support the commercialization of their products and access to specialized advice.

An integrated platform, such as AnimalCare, could play a significant role in addressing these diverse needs more effectively and accessibly. By focusing on health, hygiene, sales services, and advisory support, this application would optimize interactions between farmers, clients, and suppliers, creating a more coherent and functional ecosystem.

* 1. **Problem Statement**

How can an integrated platform like AnimalCare improve the sale of livestock while optimizing the services offered to farmers, enhancing interactions between farmers, clients, and suppliers, and facilitating the commercialization of products?

* 1. **Objectives**

1. **General Objective**

Evaluate the impact of an integrated platform on livestock health and welfare as well as on the effectiveness of interactions and transactions between farmers, clients, and suppliers.

1. **Specific Objectives**
   * Analyze the commercial benefits of such a platform for service and product providers in the livestock sector.
   * Assess the satisfaction of farmers, clients, and suppliers using the AnimalCare platform.
   * Identify the challenges and obstacles to implementing an integrated platform.
   * Propose improvements based on user feedback.

**IV. State of the Art**

Modern livestock farming aims to enhance animal production while ensuring their well-being throughout their lives. Animal health and welfare are essential concepts that directly influence the quantity and productivity of animal products. This section explores the notions of animal health and welfare in detail, focusing on the importance of good management to ensure the quality and productivity of animal products.

1. **Animal Health**

Animal health is crucial for optimal production. Here are some aspects of animal health management:

Disease Prevention and Treatment: Preventing diseases through vaccination, regular veterinary care, and hygiene practices is fundamental. Once diseases are identified, effective and timely treatment is essential to avoid spreading and minimize economic losses.

Adequate Living Conditions: Animal housing conditions should be clean, secure, and comfortable. Good living space reduces stress and the risk of diseases.

Balanced Nutrition: Proper nutrition is crucial for the growth, reproduction, and production of animals.

1. **Animal Welfare**

Animal welfare goes beyond physical health and includes mental and emotional well-being. Key elements include:

* The Five Freedoms: Animal welfare is often evaluated according to the five freedoms, which are:
* Freedom from Hunger and Thirst: Every animal should have access to water and adequate nutrition.
* Freedom from Discomfort: Provide an environment with shelter and clean resting areas.
* Freedom from Pain, Injury, and Disease: Ensure prevention and prompt treatment of animals.
* Freedom from Fear and Distress: Avoid stressful conditions and ensure appropriate care.

1. **Importance of Management for Quality and Productivity**

Proper management of animal health and welfare is essential for maintaining high standards of quality and productivity. Effective management can have a positive impact:

Improvement in Productivity: Healthy and well-treated animals are more productive. For example, a well-maintained laying hen produces more eggs, and the same applies to all animals. Better growth rates lead to higher reproduction rates and improved animal growth.

**Product Quality:** Animal welfare has a direct impact on the quality of derived products. The meat from a stressed animal is not as good as that from a well-cared-for animal.

**Cost Reduction:** A good animal health and welfare program can reduce veterinary costs and losses due to diseases. Fewer sick animals mean lower expenses for medications and treatments, as well as minimized production losses.

**Social and Regulatory Acceptance:** In many countries, animal welfare has become a major societal issue. Consumers are increasingly concerned about the conditions in which animals are raised. Respectful management of animal welfare can enhance the brand image of products and meet the expectations of consumers and regulators. It can also help in entering new markets where animal welfare standards are strict.

**Sustainability:** Animal welfare also contributes to the sustainability of farming. Farming practices that are respectful of animals are often aligned with environmentally friendly practices, thereby reducing the ecological footprint of farming. For example, practices that minimize animal stress can also reduce resource needs and environmental impact.

1. **Review of Existing Studies on Integrated Platforms for Livestock Farming**

The integration of digital platforms in the livestock industry has gained popularity, responding to the growing demand for innovative solutions to improve farm management, animal welfare, and productivity. This review highlights the observed benefits and challenges in this digital transformation, drawing on previous studies and research.

1. **Improvements in Management and Productivity**

**Resource Optimization:** Integrated platforms play a crucial role in optimizing resource management, particularly regarding feed and water. They provide real-time data that allows farmers to make informed decisions. For example, in Australia, the use of connected sensors to monitor the nutritional needs of herds has reduced feed costs by 15% while increasing productivity. This optimization results in more efficient resource use and improved yields.

* 1. **Improvement in Animal Health and Welfare**

The use of integrated platforms, combining biometric sensors and machine learning algorithms, plays a crucial role in disease prevention. These technologies allow continuous monitoring of animal health indicators, identifying risks before symptoms appear. A study conducted in North America revealed that integrating these digital tools reduced calf mortality by 20%. This highlights the importance of advanced technologies for proactive and effective animal health management, minimizing the risks of serious diseases.

* 1. **Reduction of Animal Stress**

Constant monitoring and controlled environments provided by integrated platforms help minimize stress factors for animals. In Japan, a study also showed that the use of these platforms reduced stress hormone levels in cattle by 25%. A less stressful environment promotes overall animal health and well-being, which can also enhance the quality of animal products.

* 1. **Efficiency, Time Savings, and Access to Advice for Farmers**

Platforms also offer training and online advice modules to promote best farming practices. In India, a study revealed that access to these resources significantly improved farmers' skills and knowledge, enabling them to adopt more cost-effective practices. Integrated platforms automate many daily tasks such as feeding and cleaning, freeing up time for farmers to focus on higher-value activities. In New Zealand, a study also indicated that farmers using integrated platforms reduced their manual labor by 40%, with automation contributing to more efficient farm management and reducing the physical workload on farmers.

* 1. **Analysis of Existing Applications**

Several applications have been developed to address the needs of the livestock industry in terms of management and efficiency. Here is an overview of some existing applications:

* + - **AgriWebb:** This application is designed for farm management, providing tools for animal tracking, field management, and recording veterinary treatments. It also facilitates regulatory compliance. Link: <https://www.agriwebb.com>



*Figure 1 : logo de Agriwebb*

• **Moocall:** A sensor-based monitoring system that alerts farmers to the onset of labor in cows. This helps minimize losses due to unsupervised births. Link: <https://www.moocall.com>



*Figure2: logo de Moocall*

* 1. **Conclusion**

Compared to existing applications, our application stands out due to its comprehensive integration of essential features for livestock management, including:

Optimization of Contacts Between Farmers and Clients: By facilitating interactions, our application allows for better connections for the sale of animals and animal products. • Personalized Approach to Animal Management: Unlike some generalist applications, our solution offers in-depth customization to meet the specific needs of farmers. • Intuitive and Modern User Interface: Our application emphasizes a smooth and modern user experience, which is often a weak point in existing applications. • Training and Real-Time Monitoring Features: Our application integrates training modules directly within the app, offering direct access to continuous education for farmers.

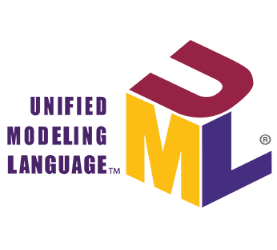
In summary, AnimalCare integrates industry best practices while offering unique features that meet the needs of modern farmers. It represents a significant advancement in using technology to transform the livestock industry.

**Chapter 2: Methodologies and Conceptual Approaches**

**I. Introduction to UML**

1. **Definition**

Unified Modeling Language (UML) is an essential tool for modeling complex software systems. It provides a series of graphical notations that help designers and developers represent different aspects of a system in an understandable way. Using UML, teams can create visual representations that facilitate communication, documentation, and understanding of systems, which is crucial for project success. UML includes various types of diagrams that focus on the structure, behavior, and interactions of the system. These diagrams help capture requirements, design architectures, plan interactions, and define workflows. The main diagrams include class diagrams, sequence diagrams, and activity diagrams, each serving a specific role in the modeling process.



*Figure 3 : logo of UML*

1. **Object-Oriented Design**

Object-Oriented Design (OO) is an approach that models’ software systems in terms of objects that interact with each other.

* **Entity Modeling**: We used object-oriented concepts to model the main entities of the system, such as breeders, animals, and products. Each main entity in the system is represented by a class that encompasses the associated data and behaviors.
* **SOLID Principles**: SOLID principles are a set of design best practices aimed at making code more understandable, flexible, and maintainable. They include concepts such as single responsibility and dependency management.

.

* 1. **UML Modeling Language**

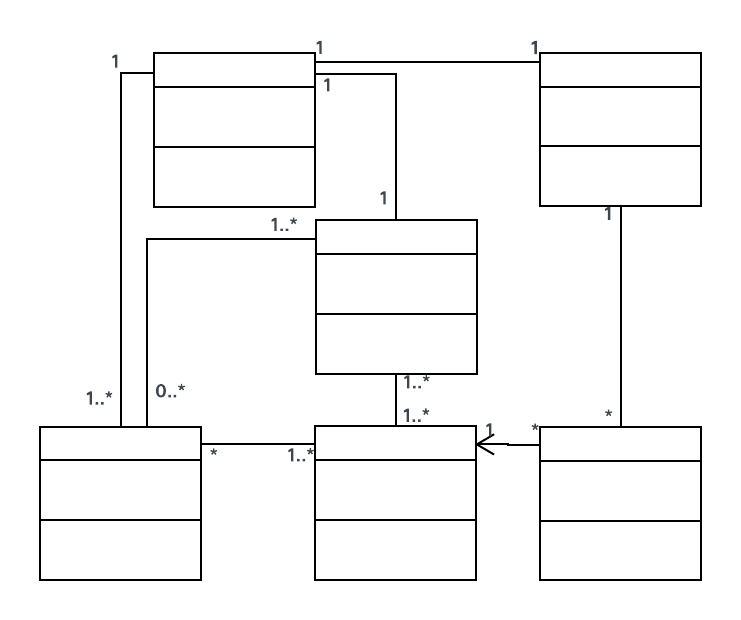
UML (Unified Modeling Language) provides a variety of diagrams that facilitate the visualization and specialization of different aspects of the system.

1. **Class Diagram**

A key element of UML, the class diagram shows the static structure of a system. It illustrates system classes, their attributes, methods, and interactions, aiding in the design of the database and system implementation.

**Components of a Class Diagram**:

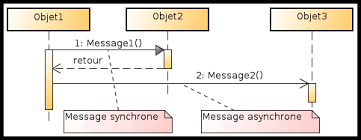
* + **Class**: Represents a system entity with its properties and behavior, depicted by a rectangle.
    - * **Sections** :
        + Class name: The top section of the rectangle that indicates the class name.
        + Attributes: The middle section that lists the properties of the class in the format name: type.
        + Methods: The lower section that lists operations or functions the class can perform, usually in the format name(parameters): returnType.
  + **Association**: Describes the relationship between two classes, represented by a line connecting them.
  + **Dependency**: Shows that one class depends on or uses another class, depicted by a dashed line with an arrow.



*Figure 4: illustration of class diagram*

1. **Sequence Diagram**

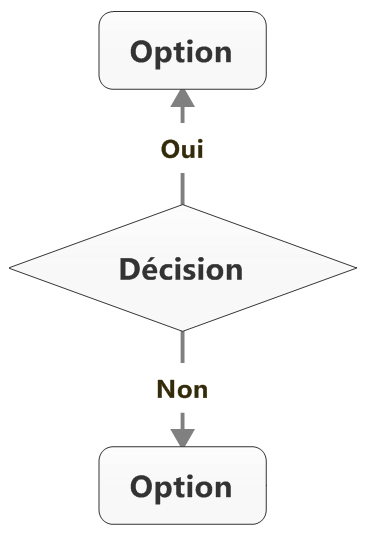
Depicts interactions between objects over time, focusing on the order of messages passed between system components.



*Figure 5: Illustration of a Sequence Diagram*

1. **Activity Diagram**

Captures workflows and business processes, showing sequential steps, conditional choices, and parallelism.



*Figure 6: illustration of an activity diagram*

* 1. **Application of UML Diagrams in Development**

UML diagrams are powerful tools that play a dominant role in software development by facilitating the specification, visualization, and documentation of systems.

* **Specification and Visualization**: UML diagrams allow for the clear specification of requirements and visualization of the expected behavior of the system. They provide a graphical representation that helps with communication and understanding between developers and stakeholders.
* **Documentation and Communication**: By providing detailed and standardized documentation, UML improves communication between development team members and other stakeholders, ensuring a shared understanding of the system's functionality and architecture.

By integrating UML into our methodology, we have created a well-structured, flexible software architecture that is ready to evolve with future needs while ensuring clear and effective communication within the team.

* 1. **Process and Development Method**

The development of our livestock management application required the adoption of modern methodologies to ensure flexibility and efficiency. We chose methods that promote adaptability to the changing needs of users while ensuring rapid delivery of functional solutions.

* **Agile Method**

The agile method is a software development approach that emphasizes human interactions, rapid deliveries, and the ability to adapt to changes. Here is how we applied it:

* + Agile Principles: Agility is based on twelve principles, such as customer satisfaction through rapid and continuous delivery of functional software and accepting change even late in development. These principles guided our process, ensuring that we remained aligned with the expectations of end users.
  + Scrum: Scrum is a specific agile framework that structures work into short and irregular sprints, usually lasting two to four weeks. Each sprint begins with planning where priorities are defined, followed by intensive development, and ends with a review and retrospective to evaluate the completed work and identify possible improvements.

| **Criteria** | **Agile Method** | **Traditional Method** |
| --- | --- | --- |
| Flexibility | High, allows changes even late in the process | Low, difficult to incorporate changes |
| Delivery | Frequent delivery | Single delivery at end of cycle |
| Collaboration | Strong collaboration between teams | Limited collaboration at certain stages |
| Documentation | Lightweight, evolves with the project | Heavy, established at the start of the project |
| Approach | Iterative and incremental | Linear |

*Table 1: Comparison of Development Methods*

| **Criteria** | **Scrum** | **RAD** |
| --- | --- | --- |
| Cycle Duration | Short sprints of 2 to 4 weeks | Rapid phases with prototypes |
| Focus | Delivery of a functional version each sprint | Rapid delivery through prototypes |
| Adaptability | High flexibility | High flexibility, but less structured |
| Documentation | Adaptive, adjusted each sprint | Minimalist, based on prototypes |
| Communication | Continuous strong communication | Strong, but sometimes UI-centered |

*Table 2: Scrum vs RAD*

* 1. **Conceptual Methodological Approach**

A well-established conceptual methodological approach is crucial for structuring and organizing the development of complex software systems. Our approach focuses on requirements analysis and object-oriented design.

* Requirements Analysis

Requirements analysis is the first important step in software development, defining what the system needs to accomplish.

* + Requirements Gathering: We conducted workshops and interviews with stakeholders to gather detailed requirements. These interactions helped us understand the specific needs of breeders, clients, and service providers.
  + Use Cases: Use cases are narrative descriptions of interactions between users and the system. They were used to document how the system should behave in various scenarios, providing a solid foundation for design and development.
* Requirements Analysis: Use Cases vs User Stories

| **Criteria** | **Use Cases** | **User Stories** |
| --- | --- | --- |
| Format | Detailed narrative with diagrams | Simple phrase: "As a… I want… so that…" |
| Level of Detail | Very detailed | Less detailed, focused on user needs |
| Documentation | Heavy, necessary for complex systems | Lightweight, evolves with needs |
| Objective | Capture all possible scenarios | Capture user needs simply |
| Usage | Complex projects with many interactions | Agile projects, user-centered agile |

*Table 3: Use Cases vs User Stories*

* 1. **Conclusion**

In this chapter dedicated to methodology and the conceptual approach, we explored essential tools for designing and modeling complex systems. Class and activity diagrams play a fundamental role in structuring and visualizing processes and system entities.

Class diagrams help define the static structure of the system by illustrating classes, their attributes, methods, and relationships. This representation is crucial for database design and organizing different domain entities, facilitating a clear understanding of the structure and interactions of classes.

Simultaneously, activity diagrams provide a dynamic view of business processes, showing workflows, conditional choices, and parallel tasks. They are particularly useful for modeling complex processes, allowing for better visualization of sequential steps and interactions in business processes.

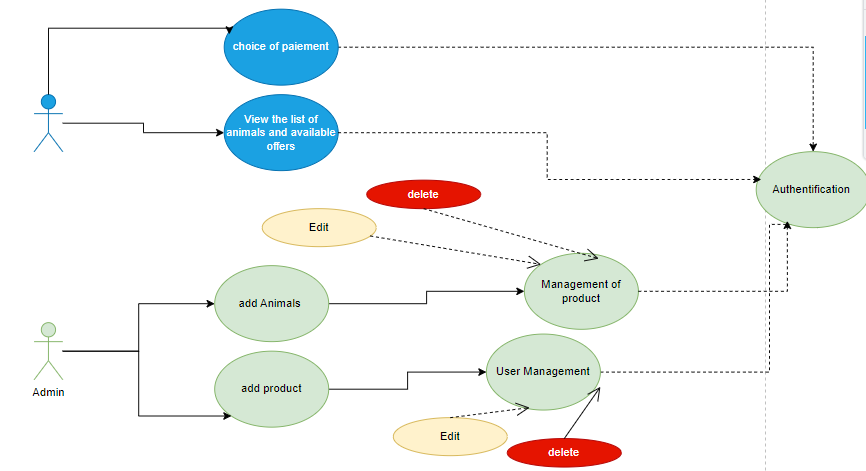
By integrating these elements into the design phase, we can better understand system requirements, optimize processes, and ensure a coherent and effective implementation. The combination of class and activity diagrams ensures a systematic and comprehensive approach, providing a solid foundation for developing solutions tailored to the specific needs of the project.

**Part II: Design and Implementation**

**Chapter 1: Design, Modeling, and Conceptual Approach**

* + 1. **Design and Modeling**

1. **Use Case Diagram**

****

*Figure 7: Illustration of the use case of the AnimalCare application.*

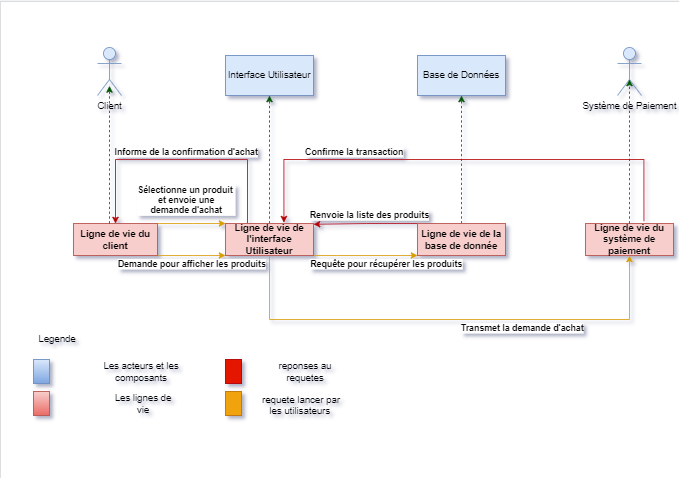
Before registration, the user can only browse the product catalog available on the platform. After registering, they can become a client of our mobile application. Once registered, they can proceed to purchase a product online through the application.

1. **Textual Description of the Animal Management Use Case**

| **Title** | **Textual Description** |
| --- | --- |
| Use Case Name | Animal Management |
| Actors | User, Management System, Database |
| Objective | Allow the user to manage the information of their animals |
| Preconditions | Animal information is updated and stored in the database |
| Main Flow | 1. The user accesses the animal management section.  2. They select an animal from the list.  3. The user modifies the animal's information.  4. The system saves the changes. |
| Alternative Flow | 1. The user is not logged in, and the system redirects them to the login page.  2. If the selected animal does not exist, the system displays an error message. |

*Table 4: Textual description of the animal management use case.*

1. **Sequence Diagram**



*Figure 8: Sequence diagram for purchasing a product on AnimalCare.*

1. **Class Diagram**

The class diagram includes the following entities:

* User Class: Attributes: Name, First Name, email, password, address. Methods: logIn(), createAccount(), updateInformation().
* Animal Class: Attributes: Name, age, species, dateOfBirth, photo, careHistory. Methods: scheduleVisit(), addVaccination().
* Veterinarian Class: Attributes: Name, contact, specialty. Methods: scheduleVisit(), addVaccination().
* Vaccination Class: Attributes: Date, vaccineType, veterinarian. Method: recordVaccination().
* Product Class: Attributes: Name, description, price, photos. Methods: addProduct(), modifyProduct().
* Sale Class: Attributes: Animal, product, price, saleDate. Method: recordSale().
* PremiumAccount Class: Attributes: startDate, expirationDate, status. Method: renewSubscription().

Relationships between the classes:

* A user can own multiple animals (1

).

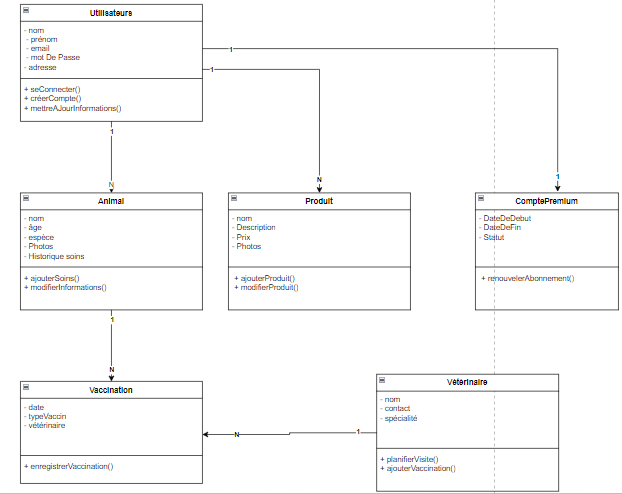
* An animal can receive multiple vaccinations, each administered by a veterinarian (1

).

* A user can sell multiple products (1

).

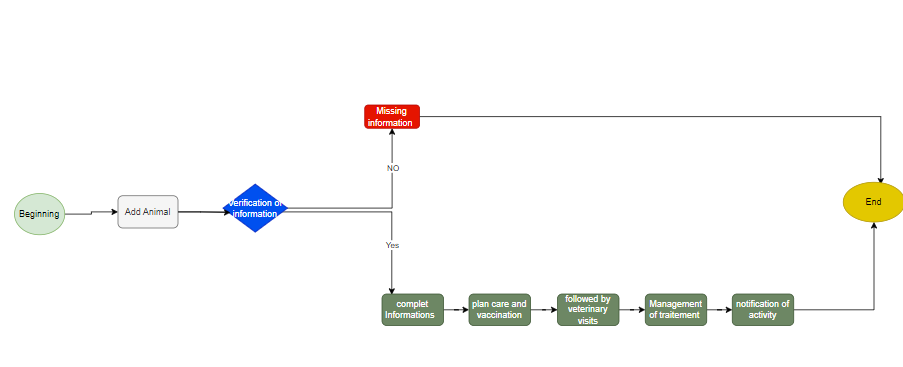
* A user can have a premium account (1

). 

***Figure 9: Class diagram of the project.***

1. **Activity Diagram**

Captures workflows and business processes, showing sequential steps, conditional choices, and parallelism.

****

*Figure 10: Activity diagram of animal Management*

* + 1. **Conceptual Approach**

1. **Comparison of Development Methods**

| **Criterion** | **Flutter** | **React Native** |
| --- | --- | --- |
| **Language** | **Dart** | **JavaScript** |
| **UI & UX** | **Near-native performance with JIT and AOT compilation** | **Good performance but sometimes inferior to native** |
| **Hot Reload** | **Yes, allows instant changes** | **Yes, allows instant changes** |
| **Ecosystem** | **Younger, growing fast** | **Mature, established community and ecosystem** |
| **Plugins & Libraries** | **Growing, some plugins require native code** | **Mainly for iOS and Android, limited web support** |
| **Learning Curve** | **Dart may require additional learning** | **JavaScript is widely known, easier for web developers** |
| **Platform Support** | **Native support for iOS, Android, web, and desktop** | **Good integration with native features through wrappers and modules** |

***Table 5: Flutter vs React Native.***

1. **Comparison Between Firestore and MySQL**

| **Criterion** | **Firestore** | **MySQL** |
| --- | --- | --- |
| **Real-Time** | **Native support for real-time updates** | **No native support, can be done with external solutions** |
| **Transactions** | **Atomic transactions, but with limitations on complex operations** | **Robust ACID transactions with full support for complex operations** |
| **Schema Flexibility** | **Flexible schema, ideal for unstructured or evolving data** | **Fixed schema, requires migrations to modify data structure** |

***Table 6: Firestore vs MySQL.***

For our application development, we chose Flutter for the development framework and Firestore for the database.

* + 1. **Conclusion**

This first chapter established the essential foundations for the project by focusing on design, modeling, and the conceptual approach. We first defined the goals and specific needs of the project, leading to the development of an appropriate conceptual architecture. Data and process modeling were rigorously addressed, ensuring a solid and coherent structure for further development. The adopted conceptual approach clarified the different components of the system, while ensuring consistency between user needs and the features offered by the application. Tools and methods used, such as UML diagrams and data models, helped structure and formalize the proposed solution, minimizing risks of inconsistencies or misunderstandings during development phases. In summary, this chapter laid the theoretical and methodological groundwork essential for the project's continuation, ensuring a smooth and coherent progression towards achieving the set goals.

**Chapter 2: Implementation and Results**

* 1. **Implementation**

1. **Work Environment**

**Hardware Environment:**

****

* **Processor: Intel(R) Celeron (R) N4000 CPU @ 1.10GHz, 1.10 GHz**
* **RAM: 8GB**
* **Hard Drive: 230GB**

 RAM: 12go

Hard Drive:256go

Android Version: 13

1. **Software Environment:**

* **Flutter**

Flutter is an open-source framework developed by Google for creating mobile, web, and desktop applications from a single codebase using the Dart language. It offers rich and performant user interfaces, with Hot Reload functionality that allows developers to instantly see code changes without restarting the application.

Access link: <https://flutter.dev>

* **Draw.io**

An online tool for creating diagrams and mockups. It’s particularly useful for designing flowcharts, organizational charts, and application mockups. Draw.io integrates easily with cloud storage services like Google Drive and Dropbox.

Access link: <https://app.diagrams.net/>

* **Visual Studio Code**

A lightweight and customizable open-source code editor developed by Microsoft. It supports a wide variety of programming languages through extensions and offers seamless integration with version control systems like Git.

Access link: <https://code.visualstudio.com/>

* **Firestore**

A cloud-based NoSQL database developed by Google, designed for real-time data storage and synchronization. Firestore integrates easily with other Firebase services and offers a flexible data model, but costs can rise with intensive usage.

Access link: <https://firebase.google.com/docs/firestore>

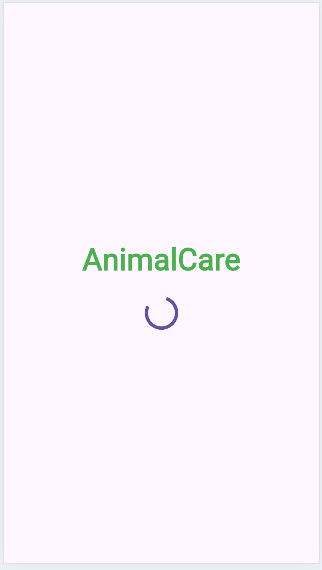
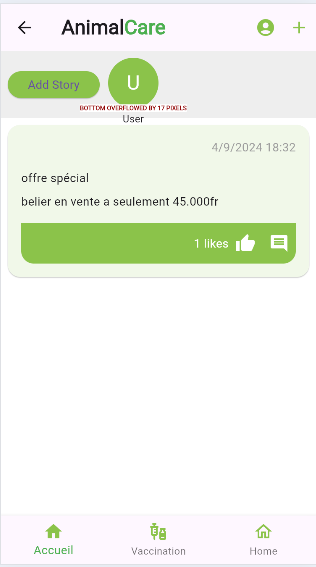
1. **Characteristics of Used Methods**

**Technology/Tool Comparison Table**

| **Tool/Technology** | **Type** | **Language** | **Description** | **Key Features** |
| --- | --- | --- | --- | --- |
| Flutter | UI Framework | Dart | Open-source framework for developing mobile, web, and desktop applications. | Cross-platform development, Hot Reload, Rich and customizable widgets |
| Draw.io | Diagram Tool | N/A | Online tool for creating diagrams, mockups, and flowcharts. | Real-time collaboration, Integration with cloud storage, Free and accessible |
| Visual Studio Code | Code Editor | N/A | Open-source code editor with support for various programming languages and development tools. | Extensive extensions, Git integration, Fast performance, Customization |
| Firestore | NoSQL Cloud Database | N/A | Cloud database for real-time data storage and synchronization. | High scalability, Integration with Firebase, Flexible data model |

*Table 7: Characteristics of Used Methods*

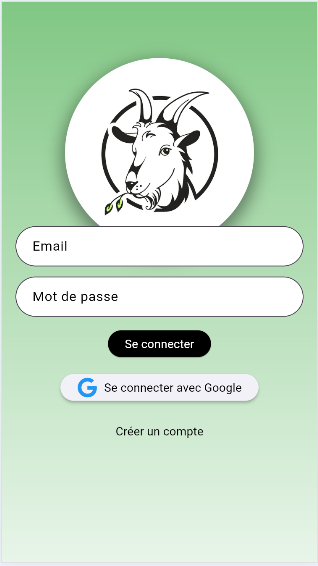
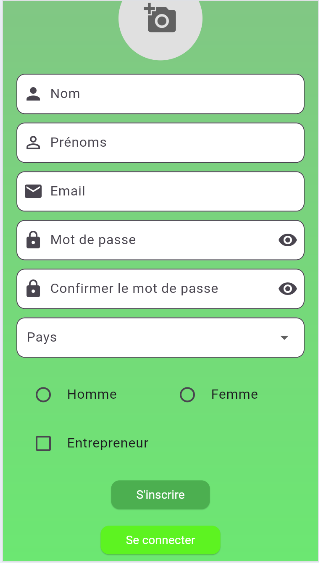
**II. Results**

****

*Figure 10: Splash screen at app launch Figure 11: Home screen (catalog)*

The splash screen appears for 5 seconds

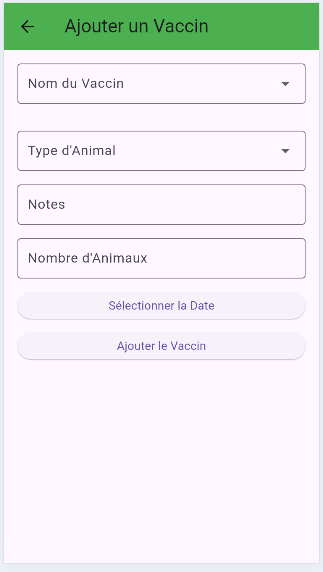
when the application starts.

 ****

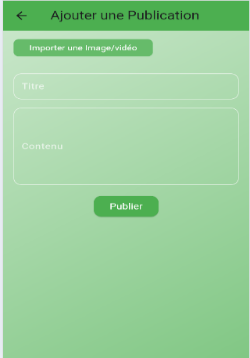
*Figure 12: Login screen Figure 13: Registration screen*

This interface allows users to log This page allows new users to register.

in or create an account

** **

*Figure 14: Vaccine management Figure 15: Adding a vaccine*

****

*Figure 16: Adding a post*

**Conclusion**

The "AnimalCare" project aims to address the modern challenges of managing domestic and farm animals. By integrating care management, vaccination scheduling, and a sales platform, this mobile application offers a comprehensive and innovative solution for breeders, veterinarians, and animal owners. The results demonstrate that "AnimalCare" not only facilitates daily animal management but also improves communication and transactions within the sector. With the potential for additional features and market expansion, "AnimalCare" is poised to become an essential tool in animal management.

**Bibliography**

Google Firebase - "Firebase Documentation." <https://firebase.google.com/docs>

Flutter - "Flutter Documentation." Google Developers. <https://flutter.dev/docs>

Draw.io - "Draw.io Online Diagram Software." <https://www.diagrams.net/>

Visual Studio Code - "Visual Studio Code Documentation." Microsoft. <https://code.visualstudio.com/docs>

Firestore - "Cloud Firestore Documentation." <https://firebase.google.com/docs/firestore>

Meat & Livestock Australia (MLA) <https://www.mla.com.au/research-and-development/Animal-Health-welfare/Digital-technology>