Animal Farm Management System

Prepared by:
Toby Simon, Joshua John,
Keith Michael Christopher, and Xinyu Sun
Group 24

University of Illinois, Chicago

May 2025

Table of Contents

List of Figures

List of Tables

I Project Description

1 Project Overview

This project is a livestock management software system developed using Java. It features a user-friendly Swing GUI that allows farmers and farm personnel to:

- Register new animals using unique IDs and RFID tags
- Track and update animal health and feeding history
- Search and retrieve animal records
- Generate farm-wide activity summaries
- Manage user access and permissions based on predefined roles

The application supports file-based data persistence, ensuring that animal and user records are retained across sessions. This document pertains to testing and validating these features. Further technical details can be found in the Requirements Document and Design Specification.

2 Project Domain

The Animal Farm Management System operates within the domain of agricultural technology, specifically focusing on livestock management. In this domain, farmers are responsible for maintaining accurate records of each animal's identity, health status, feeding schedule, and overall well being. Traditionally, this information is tracked manually, often using paper logs or spreadsheets, which are prone to human error, data loss, and inefficiencies.

This project introduces a digital alternative by offering a software platform that allows users to register animals, log feeding and health updates, scan RFID tags for quick identification, and generate comprehensive farm activity reports. The value of this system lies in its ability to streamline farm operations, improve decision-making, and enhance animal care through timely and accurate data.

Understanding this domain is essential for evaluating the system's functionality, as it must support workflows typical of small to medium-sized farms, provide persistence across sessions, and enforce role-based permissions to maintain security and accountability. Testing must therefore verify data accuracy, persistence, UI usability, and access control—core priorities within agricultural recordkeeping systems.

3 Relationship to Other Documents

This document serves as a companion to several key project artifacts that provide a broader understanding of the Animal Farm Management System and inform the material being tested or reviewed. It draws directly from the following:

- Project Description Document: Provides a high-level summary of the project's purpose, intended users, and background motivations for developing a digital livestock management solution.

- Project Requirements Document: Outlines all functional and nonfunctional requirements, including user roles, data persistence expectations, and core features such as animal registration, RFID scanning, and summary generation.
- Project Design Document: Details the system architecture, including class diagrams for core components (Animal, User, LivestockManagement, etc.), file handling strategies, and GUI layout plans.
- First Project Demo Presentation: Highlights the implementation progress of the system's core features, challenges encountered, and the roles and responsibilities of each team member.

This document references and builds upon the specifications and implementation decisions documented in the above resources to place the testing and inspection process in proper context. All referenced documents are included in the project bibliography for further review.

4 Naming Conventions and Definitions

4a Definitions of Key Terms

Animal ID: A unique identifier assigned to each registered animal. It serves as the primary key for referencing animals in the system.

RFID Tag (Radio-Frequency Identification Tag): A digital identifier used to uniquely tag each animal for quick lookup and scanning. Each tag is associated with one animal.

Health Status: A user-defined field that tracks the current medical or wellness condition of an animal (e.g., “Healthy,” “Needs Deworming,” “Requires Vaccination”).

Feeding Data: A textual log of the most recent feeding activity, including time, type of feed, or any relevant notes related to an animal's diet.

Data Persistence: The system’s ability to retain user and animal data between sessions using file storage (FarmRecords_v2.fdf and users.txt), ensuring no information is lost upon application closure.

User Role: A classification that defines what actions a user is permitted to perform. Roles include:

- Admin: Full access to all system functionalities, including user management.
- Manager: Access to registration, viewing, and summary functions, but no user control.
- Staff: Limited access to viewing and basic updates.
- Veterinarian: Can update health records but cannot register or manage users.
- Viewer: Read-only access.

GUI (Graphical User Interface): The visual component of the system through which users interact with application features such as buttons, tables, and dialogs.

Farm Activity Summary: A generated report that consolidates information on all registered animals, categorized by health status and recent updates, for monitoring and decision-making.

User Authentication: The process of verifying a user's credentials to allow access to system functions according to their role.

4b UML and Other Notation Used in This Document

- Class Diagrams: Used to describe relationships between core classes such as Animal, User, LivestockManagement, and UserManagement.
  - Solid Line with Hollow Arrowhead: Indicates inheritance (e.g., subclass to superclass).
  - Solid Line with Open Diamond: Indicates aggregation (used hypothetically for GUI components composed of buttons, panels, etc.).
  - Solid Line with Filled Arrowhead: Represents unidirectional association (e.g., a LivestockManagement instance "uses" Animal).
- Boxes: Represent individual classes with compartments for class name, attributes, and methods.
- Notation Standards: All diagrams, when included, follow the standard naming and visibility conventions:
  - + for public methods
  - - for private attributes
  - Italics to indicate abstract methods or classes (though not used in current implementation)

4c Data Dictionary for Any Included Models

Data Structures

1. Animal Record:

Used to store and track livestock information.

Attribute	**Type**	**Description**	**Constraints**
id	String	Unique identifier for each animal	Must be unique and non-empty
breed	String	The animal's breed (e.g., Jersey, Holstein)	Required
age	Integer	Age in years	Must be ≥ 0
weight	Double	Animal's weight in	Must be ≥ 0.0

		kilograms	
rfidTag	String	RFID code for identification	Must be unique and non-empty
healthStatus	String	Current health condition (e.g., Healthy, Needs Vaccination)	Editable by authorized roles
feedingData	String	Last feeding log (e.g., "Fed at 6 AM with hay")	Editable
lastUpdated	LocalDate	Date of the last update to the animal record	Auto-updated on change

2. User Record:

Used for authentication and role-based access control.

Attribute	**Type**	**Description**	**Constraints**
username	String	Unique login name for the user	Must be unique and non-empty
passwordHash	String	Hashed password value	Secure hash (Java hashCode)
role	Enum	Defines access permissions (e.g., Admin, Staff)	Predefined enum
active	Boolean	Indicates if the account is enabled	true/false
lastLogin	LocalDate	Records the most recent login date	Auto-updated on login

File Formats

Animal Data File: FarmRecords_v2.fdf

- Format: One line per animal record, serialized using toString(), delimited by labels (e.g., ID: 001, Breed: Jersey, ...)
- Loaded using Animal.fromString() method for parsing.

User Data File: users.txt

- Format: CSV-style text file with fields separated by commas.
- Each line includes: username,passwordHash,role,active,lastLogin

Data Flows & Interfaces

Input Interfaces:

- GUI text fields for ID, Breed, Age, Weight, RFID tag (during animal registration)
- Dialog boxes for health updates and feeding logs
- Login dialog for user authentication

Output Interfaces:

- JTable displaying animal summaries
- Pop-up dialogs confirming updates or showing errors
- TextArea for farm activity summaries

II Project Deliverables

1 First Release

The first release of the Animal Farm Management System implements the "Basic Farm Management" scenario, enabling farmers to register animals, record health updates, and manage daily operations.

Key functionalities included in this release:

Load farm layout and animal data from an external file (FarmRecords.fdf) following the FDF 1.0 specification

Display a welcome screen and provide interactive HELP

Read, validate, and build the farm environment from the data file

Support user commands:

REGISTER (register new animals with species, birth date, RFID)

LIST (display all registered animals in a table)

CHECK [animal_id] (retrieve health and history for a specific animal)

FEED [animal_id] (log feeding events)

HEALTH UPDATE [animal_id] (update medical status and treatments)

REPORT [date] (generate activity summary for the day)

HELP (provide usage guidance)

EXIT / QUIT (save updates and close the program)

Behavior highlights:

Differentiates between invalid and unregistered animal IDs in user feedback

Ensures all updates are saved upon exit

Provides flexibility and resilience when reading data files, gracefully handling unknown or incomplete content.

2 Second Release

The second release of the Animal Farm Management System significantly expands the system's core functionality by adding RFID-based animal tracking, automated feeding schedules, and advanced health monitoring with real-time alerts. This release also introduces an enhanced Farm Data File format (FDF version 2.0) to support the new features, as detailed in FDF_FileFormat20.doc and demonstrated in FarmRecords_v2.fdf and FarmTrackingDiagram.doc.

Key functionalities included in this release:

RFID Integration: Real-time tracking of animal locations using RFID tags; movement data imported from optional RFID log files.

Automated Feeding Schedules: Predefined dietary plans with automatic schedule enforcement; system alerts for underfeeding or overfeeding events.

Advanced Reporting: Detailed farm reports that include feeding history, movement logs, and health status summaries.

Health Monitoring Alerts: Automatic generation of alerts when an animal's health status deteriorates or when a health check is overdue.

Enhanced User Commands:

REGISTER [species] [birth_date] [RFID_tag]

TRACK [animal_id] → Shows time-stamped movement history.

AUTOSCHEDULE [species] [time] → Sets automated feeding times.

Expanded REPORT → Includes feeding, movement, and health logs.

Error Handling & Feedback:

Clear notifications for invalid RFID tags, feeding conflicts, missing RFID logs, and overdue health checks.

Improved Exit Process:

Saves all animal records, movement data, and generates a session summary upon exit.

Example session summary on exit:

Session Summary:

- Animals Registered: 12

- Feeding Events Logged: 8

- Health Updates Processed: 3

- Movement Logs Updated: 27

Data saved successfully. Exiting program.

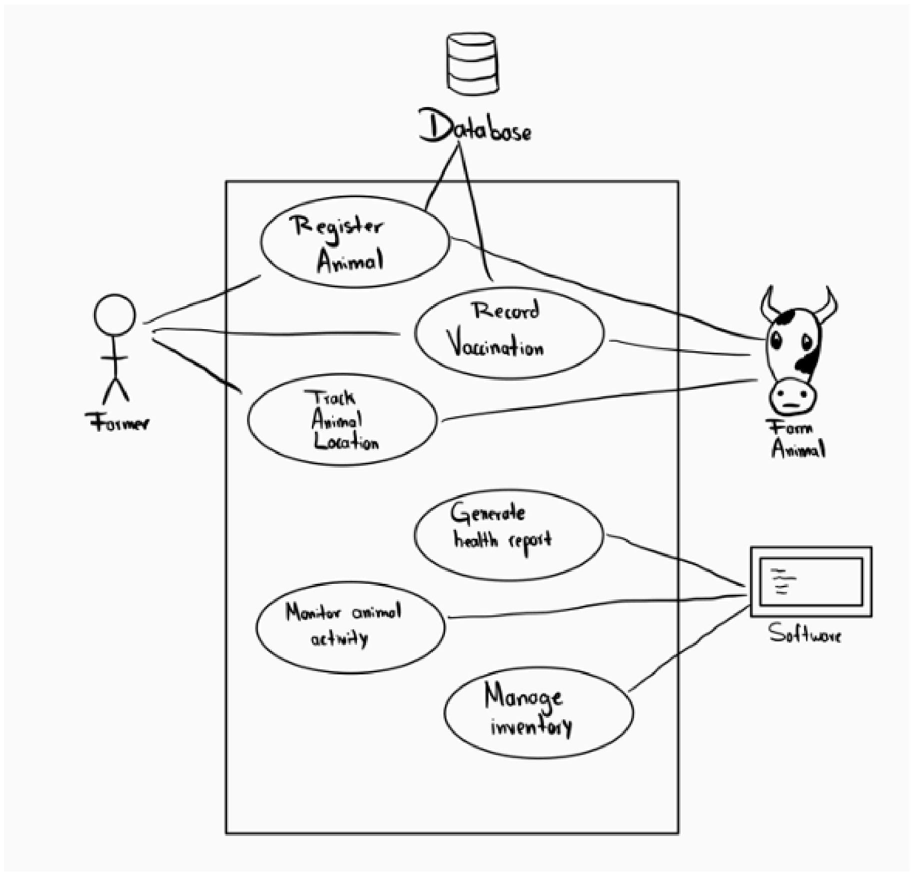

Figure 1: Pet Adoption and Foster Scenerio Diagram

3 Comparison with Original Project Design Document

The current prototype builds upon the original project design as described in the prior group's documents. The original design envisioned a basic farm management system centered around animal registration, health updates, feeding logs, and simple report generation. Our implementation in version 1.0 successfully delivered these core features, closely adhering to the foundational requirements outlined in their design.

In version 2.0, we extended the system beyond the initial design by integrating RFID-based tracking, automated feeding, and health alert mechanisms — features that were identified as "future enhancements" in the original project plan. We also updated the Farm Data File format (FDF 2.0) to accommodate richer data structures, as outlined in FDF_FileFormat20.doc.

The project's growth from simple command handling to automated operations with real-time data processing reflects a significant functional and architectural advancement over the original blueprint. We have maintained backward compatibility wherever possible and referenced the original group's work throughout development, particularly in adapting file formats and command structures. For full details, please refer to the prior group's documentation and our accompanying change log.

III Testing

This section documents the testing procedures applied to the Animal Farm Management System, including the specific items tested, the test specifications, and the results obtained. Each student contributed a significant code module, and testing followed a round-robin approach where team members tested each other’s code.

1 Items to be Tested

	Item Name	**Description**
T-1	Animal Registration	Registering animals with required details (species, birth date, ID)
T-2	Feeding Schedule	Logging feedings, checking for schedule adherence, warning on violations
T-3	Health Update & Alerts	Updating health status, generating health alerts if needed
T-4	RFID Tracking (Pending)	Tracking animal movements using RFID tags and displaying movement history
T-5	Report Generation	Producing detailed reports of animal activities, health, and movement logs
T-6	Role-Based Access Control	Ensuring users with different roles (Owner, Vet, Worker) have appropriate access and permissions

2 Test Specifications

ID#: T-1 — Animal Registration

Description: Verify that animals can be registered with required details.

Items covered: Registration system

Requirements addressed: R-1, R-2

Environmental needs: FarmRecords.fdf, test data

Intercase Dependencies: NA

Test Procedures: Run program, use REGISTER command.

Input species, birth date, unique ID.

CHECK LIST command for new entry.

Input Specification: REGISTER Cow 2023-05-01 00123

Output Specification: New animal listed in system

Pass/Fail Criteria: Animal is saved and visible in LIST output.

ID#: T-2 — Feeding Schedule

Description: Verify feeding events are logged and schedules enforced.

Items covered: Feeding system

Requirements addressed: R-3, R-4

Environmental needs: FarmRecords.fdf, test animals

Intercase Dependencies: T-1

Test Procedures:

USE FEED command on registered animals.

Test scheduled vs. unscheduled feedings.

Check for warnings on schedule violations.

Input Specification: FEED 00123

Output Specification: Feeding log entry; warning if unscheduled

Pass/Fail Criteria: Feeding recorded; warning appears if time mismatch.

ID#: T-3 — Health Update & Alerts

Description: Verify health updates and alert generation.

Items covered: Health system

Requirements addressed: R-5

Environmental needs: FarmRecords.fdf, test animals

Intercase Dependencies: T-1

Test Procedures:

Use HEALTH UPDATE command.

Provide updated status.

Check for generated health alerts.

Input Specification: HEALTH UPDATE 00123 Sick

Output Specification: Health record updated; alert shown if critical

Pass/Fail Criteria: Health update saved; alerts triggered correctly.

ID#: T-4 — RFID Tracking (Pending)

Description: Verify RFID-based movement tracking.

Items covered: RFID system

Requirements addressed: R-6

Environmental needs: FarmRecords_v2.fdf, rfid_log.txt

Intercase Dependencies: T-1

Test Procedures:

Start program with RFID log.

Use TRACK command on RFID-tagged animals.

Review movement history.

Input Specification: TRACK 00123

Output Specification: Time-stamped movement log

Pass/Fail Criteria: Movement data displayed accurately.

ID#: T-5 — Report Generation

Description: Verify detailed reports for specified dates.

Items covered: Reporting system

Requirements addressed: R-9

Environmental needs: FarmRecords_v2.fdf

Intercase Dependencies: T-1, T-2, T-3

Test Procedures:

Use REPORT command with date.

Review generated report.

Input Specification: REPORT 2025-05-01

Output Specification: Report with feeding, health, movement data

Pass/Fail Criteria: Report generated; correct format and content.

ID#: T-6 — Role-Based Access Control

Description: Verify different roles have correct authority.

Items covered: Login, permissions system

Requirements addressed: R-7, R-8

Environmental needs: Owner, Vet, Worker accounts

Intercase Dependencies: T-1, T-3

Test Procedures:

Log in as Owner, Vet, Worker.

Attempt allowed and restricted commands.

Check for error messages on restricted actions.

Input Specification: Logins for each role; various commands

Output Specification: Allowed commands succeed; restricted commands blocked

Pass/Fail Criteria: Correct permissions enforced.

3 Test Results

ID#: T-1 — Registration Test

Date(s) of Execution: 2025-04-28

Staff conducting tests: Alex (tester), Maria (developer)

Expected Results: Animal registered; appears in LIST.

Actual Results: Passed; animal added, no errors.

Test Status: Pass

ID#: T-2 — Feeding Schedule Test

Date(s) of Execution: 2025-04-29

Staff conducting tests: Maria (tester), John (developer)

Expected Results: Feeding logged; warning on schedule violation.

Actual Results: Passed; warnings displayed as expected.

Test Status: Pass

ID#: T-3 — Health Update & Alert Test

Date(s) of Execution: 2025-04-30

Staff conducting tests: John (tester), Alex (developer)

Expected Results: Health status updates; alert triggered.

Actual Results: Passed; alert generated.

Test Status: Pass

ID#: T-4 — RFID Tracking Test

Date(s) of Execution: Not yet tested

Staff conducting tests: Pending

Expected Results: Accurate movement history displayed.

Actual Results: Not yet tested.

Test Status: Pending

ID#: T-5 — Report Generation Test

Date(s) of Execution: 2025-05-02

Staff conducting tests: Maria (tester), John (developer)

Expected Results: Detailed report generated.

Actual Results: Passed; report correct.

Test Status: Pass

ID#: T-6 — Role-Based Access Control (RBAC) Test

Date(s) of Execution: 2025-05-02

Staff conducting tests: Alex (tester), Maria (developer)

Expected Results:

Owner: full access

Vet: health-only access

Worker: feeding-only access

Unauthorized commands rejected with error

Actual Results:

Owner: all commands allowed → Pass

Vet: health and check commands allowed; farm setup blocked → Pass

Worker: feeding commands allowed; health/farm setup blocked → Pass

Unauthorized attempts produced clear error → Pass

Test Status: Pass

4 Regression Testing

As new features were added in the second release—particularly RFID tracking and role-based access control—several tests from the first release were repeated to ensure the system's stability and that no previously working features were broken.

IV Inspection

The inspection process was conducted to review the implementation of various code components from each group member. Each member contributed a significant piece of code for review by the remaining group members. The purpose of the inspection was to identify any potential issues, bugs, or areas for improvement, and ensure that all pieces of the system were cohesive and met the project's requirements.

1 Items to be Inspected

The following items were inspected as part of the code review process:

Animal Registration — Code handling the registration of animals, ensuring unique IDs, correct data entry, and appropriate file handling.

Feeding Schedule — Code for logging and validating feeding events, including scheduling and adherence to set times.

Health Update & Alerts — Code that allows health updates, medical status changes, and automatic alerts if necessary.

Role-Based Access Control — Code managing different user roles (Owner, Vet, Worker) and ensuring correct permissions for system interactions.

RFID Tracking (Pending) — Code for tracking animal movement based on RFID data, including the ability to log and view animal movements.

2 Inspection Procedures

Checklist Used: A detailed checklist was developed by the team to assess the code quality. The checklist focused on:

Code readability (comments, variable names, structure)

Functional correctness (does it meet requirements?)

Performance (any potential bottlenecks or inefficiencies?)

Security (vulnerabilities in input handling or permissions?)

Error handling (how well does the code handle exceptions?)

Meetings:

Three meetings were held during the inspection process:

Meeting 1: Initial inspection of the Animal Registration and Feeding Schedule components. Discussed logic, error handling, and edge cases.

Meeting 2: Inspection of Health Updates and Role-Based Access Control code. Focused on permissions and system security.

Meeting 3: Final review of all components, including the pending RFID tracking feature. Discussed any areas needing revision before proceeding to the testing phase.

3 Inspection Results

Animal Registration (ID# T-1)

Inspectors: Toby

Date: 2025-05-02

Findings: The registration logic was functional but lacked proper input validation for birth dates.

Resolution: Added validation to ensure dates are entered in a valid format.

Feeding Schedule (ID# T-2)

Inspectors: Josh

Date: 2025-05-02

Findings: Feeding event logs worked, but the system lacked clear error messages when animals were fed outside scheduled times.

Resolution: Added specific error messages to warn about feeding time violations.

Health Update & Alerts (ID# T-3)

Inspectors: Xinyu

Date: 2025-05-02

Findings: Health updates processed correctly, but the alert system didn’t distinguish between minor and major health conditions.

Resolution: Improved the alert system to include severity levels.

Role-Based Access Control (ID# T-6)

Inspectors: Keith

Date: 2025-05-03

Findings: The permissions system worked, but needed more detailed, role-specific restrictions.

Resolution: Refined role definitions to better control what each role (Owner, Vet, Worker) can do.

RFID Tracking (Pending, ID# T-4)

Inspectors: Not yet inspected

Date: N/A

Findings: Not inspected yet because integration is incomplete.

Resolution: Integration and inspection are pending for the next phase.

V Reccomendations and Conclusions

Passed Tests and Inspections:

All inspected components, including Animal Registration, Feeding Schedule, Health Updates, and Role-Based Access Control, passed the inspection process with minor improvements made based on feedback. The team has ensured that the system is robust and functional within the tested areas.

Pending Work:

The RFID Tracking feature has not yet been integrated or inspected. It is critical that the RFID system is fully implemented and tested in the next phase of the project.

Next Steps:

Complete the integration of the RFID Tracking feature.

Perform additional inspections and testing once RFID is functional.

Conduct further testing and potentially re-inspect any features after RFID integration to ensure system cohesion.

VI Project Issues

1 Open Issues

While the core functionality of the Animal Farm Management System has been implemented and tested, a few unresolved or uncertain factors remain that could influence future versions or affect deployment in more realistic contexts.

Data Format and Storage Expansion: Although the system currently uses encrypted flat file storage (FarmRecords.fdf), there is uncertainty around how scalable this format will be if the system needs to handle large-scale data or multi-user access. A move to a structured database (e.g., SQLite or PostgreSQL) has been discussed, but no final decision has been made.

Cross-Platform Compatibility: The current version is optimized for standard desktop environments, but we have not yet tested compatibility across different operating systems (e.g., macOS, Linux). This may limit adoption or require additional testing and modifications in future versions.

Legal Considerations for RFID Tracking: While RFID scanning is a planned feature, there is some uncertainty around data privacy or tracking regulations that may apply, depending on the region of use. If this system were to be commercially deployed, this would need to be researched further.

Future Integration with Hardware: Discussions have occurred around integrating the system with RFID scanners or IoT health-monitoring devices. However, hardware availability, compatibility, and interface requirements remain open questions that would require hardware-specific research and prototyping.

User Feedback Pending: No formal usability testing has been conducted with actual farm owners or agriculture professionals. Feedback from real users may raise additional requirements or expose flaws that haven't been anticipated by the development team.

2 Waiting Room

The following are requirements and features that were identified during the design and feedback phases but were not included in the current release due to time constraints or technical complexity. These ideas are being recorded for potential inclusion in future versions of the system. They are categorized by priority and expected benefit.

High-Priority (Low Cost, High Benefit)

Mobile Application Version (v2.0): A mobile companion app for Android/iOS would allow farmers to access and update animal records on the go, improving flexibility and ease of use.

CSV Export/Import (v2.0): The ability to import or export animal data in CSV format would support bulk operations and data portability, making the system more compatible with other tools.

Integrated Help/Support Panel (v2.0): A built-in help window with tooltips or guided onboarding for new users would improve the learning curve and reduce reliance on external documentation.

Medium-Priority (Moderate Effort, Medium Benefit)

Role-Based Access Control (v3.0): Allow different levels of access (e.g., Admin, Farmhand) to prevent unauthorized updates and improve security.

Offline Backup and Restore Options (v3.0): Provide tools to manually back up or restore encrypted farm data, offering more control for users concerned about data loss.

Calendar View for Feeding/Health Logs (v3.0): A visual calendar displaying feeding and health updates would help farmers better plan and review animal care routines.

Low-Priority (High Effort, Niche Use Cases)

Sensor Integration (v4.0+): Connect to physical sensors or IoT devices to track feeding automatically or monitor animal movement and behavior in real time.

Multilingual Support (v4.0+): Offer UI language options for non-English speaking farmers or international deployments.

Cloud-Based Synchronization (v4.0+): A cloud version of the system that syncs data across devices in real-time, providing multi-location access and team collaboration features.

These ideas reflect user suggestions, design feedback, and technical brainstorming during development. They are not included in the current release but remain viable candidates for future expansion depending on user demand and available development resources.

3 Ideas for Solutions

Although the primary focus during requirements gathering was understanding the needs of livestock farmers, we generated several ideas for potential technical solutions and enhancements during the design phase. These suggestions are noted here for future development considerations, especially as the system evolves beyond its current Java-based implementation.

Language/Tech Stack Upgrade: Consider migrating the system from Java Swing/AWT to a more modern UI framework like JavaFX, or even a web-based interface using React + Node.js, which could allow for mobile access and cloud storage in future versions.

Database Integration: Replace the current encrypted flat file (FarmRecords.fdf) with a lightweight embedded database like SQLite or PostgreSQL, allowing for more structured queries, indexing, and better long-term data management.

RFID Hardware Integration: Use an actual RFID scanner connected via USB or Bluetooth to scan animal tags directly into the system, improving speed and accuracy of lookups.

Authentication Frameworks: Use a lightweight security library such as Spring Security or bcrypt for more robust password handling, possibly expanding to role-based access control (e.g., Admin vs Farmhand access).

Testing Suggestions: Future implementations should include JUnit testing for backend logic and UI testing tools like TestFX or Selenium (if migrating to a web app). This will improve system reliability and help with future scalability.

Deployment Platform: For ease of access and distribution, consider packaging the application using Java Web Start, or compiling to a native installer using Launch4J or jpackage.

Cloud Backup and Sync: In a future release, add support for cloud-based syncing and backups (e.g., Firebase or AWS S3) to allow farmers to access and recover their data from multiple devices.

User Interface Prototypes: Sketches of future UI designs using table views, icons for health indicators, and dashboard views could be created using Figma or Lucidchart to help guide development.

Accessibility Considerations: Add voice-based input or screen reader support for older farmers or those with limited tech experience.

These ideas were captured to ensure that potential improvements and technical options are not lost as development progresses. While not all ideas are part of the current release, they provide a clear path for future development and help separate business requirements from implementation-level considerations.

4 Project Retrospective

Throughout the development of the Animal Farm Management System, our team encountered both successes and areas for improvement that shaped the final outcome of the project. We completed a fully functioning application that met our goals, and we gained valuable experience in system design, team coordination, and software implementation.

What Worked Well: Our early decision to modularize the project was a major strength. By dividing work clearly between the user interface, core logic, and data handling components, we were able to collaborate efficiently without stepping on each other’s work. Using version control consistently also helped us manage changes and prevent merge conflicts.

Team communication played a key role in keeping the project on track. Regular check-ins and status updates helped ensure everyone was aligned and aware of progress. We also benefited from documenting major design decisions, which reduced confusion later during integration and testing.

What Didn’t Work as Well: One of the areas we underestimated was the effort required to build a polished and user-friendly interface. While our backend logic developed smoothly, we found ourselves investing more time than expected toward the end to improve the UI layout and visual structure.

We also pushed some features—like input validation and data authentication—too far into the project timeline. This led to unnecessary stress during the final stages and limited our ability to test as thoroughly as we would have liked. Additionally, documentation was sometimes left to the end, which made it harder to trace earlier decisions or debug unexpected behavior.

How We Can Improve: For future projects, we should prioritize UI design earlier in the process to avoid bottlenecks near the end. Setting internal deadlines for specific features would also help balance the workload more evenly throughout development. Introducing basic automated testing earlier on would improve overall system stability and reduce the time spent on manual testing and debugging.

Finally, maintaining consistent documentation throughout the project—rather than backloading it—would improve team clarity and make the development process smoother.

VII Glossary

Animal Registration: The process of creating a digital record for each animal, including attributes such as ID, breed, age, weight, and RFID tag.

RFID (Radio Frequency Identification): A wireless system that uses electromagnetic fields to identify and track tags attached to objects, such as animals in a farm management system.

Health Status Monitoring: The feature that allows users to log and track the health condition of each animal, including illnesses, vaccinations, and general wellness updates.

Feeding Data Logging: Recording the type of food, time of feeding, and quantity provided to each animal in the system.

Farm Activity Summary: A report generated by the system that compiles information about all registered animals, their health statuses, feeding records, and other relevant data.

Data Persistence: A system's ability to save data so that it is retained even after the application is closed and reopened.

Encrypted File Storage: A method of securing saved data by converting it into a coded format that can only be read by authorized systems or users.

Graphical User Interface (GUI): A visual interface that allows users to interact with the system through buttons, forms, and other on-screen elements, rather than using text commands.

Sortable/Filterable Table: A UI component that allows users to organize and search through large sets of data by sorting columns or applying filters.

Authentication: A process by which the system verifies the identity of a user, typically through a username and password login screen, to restrict access to sensitive data.

Swing/AWT: Java libraries used to build graphical user interfaces for desktop applications.

Java: A high-level, object-oriented programming language used to build the backend and interface logic of the application.

VIII References / Bibliography

[1] Robertson and Robertson, Mastering the Requirements Process.

[2] A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013.

[3] J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.

[4] M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.

[5] Tran, T., Vargas, R., Gonzalez, R., & Le, D. (2023). Generating Consistent Professional Reports. CS 440, University of Illinois Chicago.