UE22CS352B - Object Oriented Analysis & Design

Mini Project Report

Dairy Farm Management System

Submitted by:

Name :	***Kiran J Rajpurohit***	***PES2UG22CS265***
	Madhura H B	***PES2UG22CS291***
	Manya Singh	***PES2UG22CS306***
	Midhushi Mahajan	***PES2UG22CS310***

Semester 6 Section E

Faculty Name: Dr. Kamatchi Priya L

January - May 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

Problem Statement:

Managing a dairy farm involves a wide array of daily operations like monitoring cow health, recording milk production, managing feed, and tracking breeding cycles. Traditionally, these tasks are handled manually, leading to errors, inefficiencies, and difficulty in scaling operations.

This project aims to solve these issues by developing a Cow Management System, a software solution that digitizes and streamlines the process of managing dairy farm activities. The system provides a centralized platform to store and manage detailed records of cows, track milk yield, monitor health, and manage breeding schedules — all within a user-friendly interface.

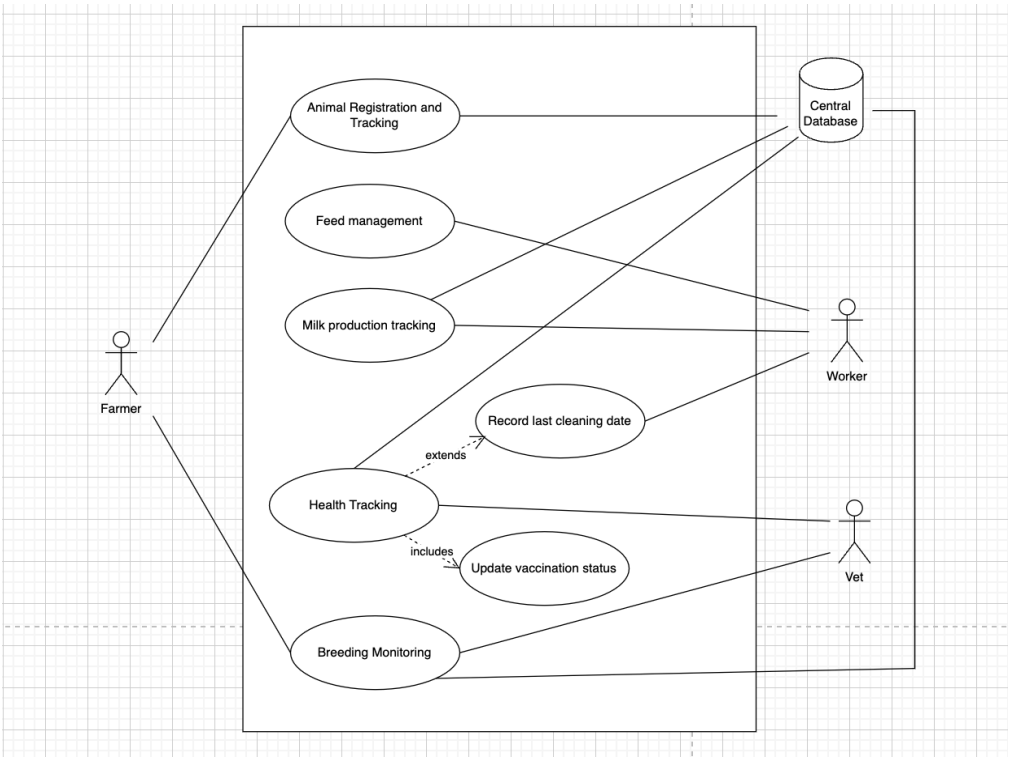
Building a dairy farm management system using the MVC architecture to manage the registration of animals and also track animal health and keeping tabs on the milk production. It can also help track animal feed across the farm. Has access control for 3 different types of users: Farmers, Workers and Veterinarians ensuring security of data and information. This platform consolidates animal registration, health monitoring, tracking milk production, breeding analytics and feed management into a single platform.

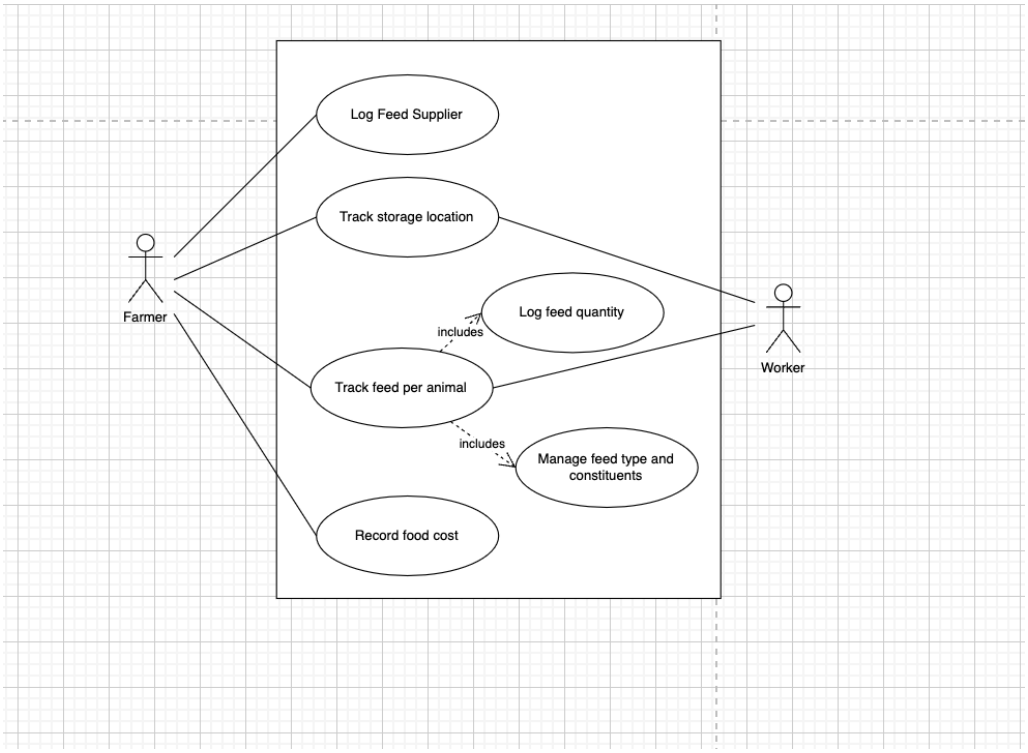
Key Features

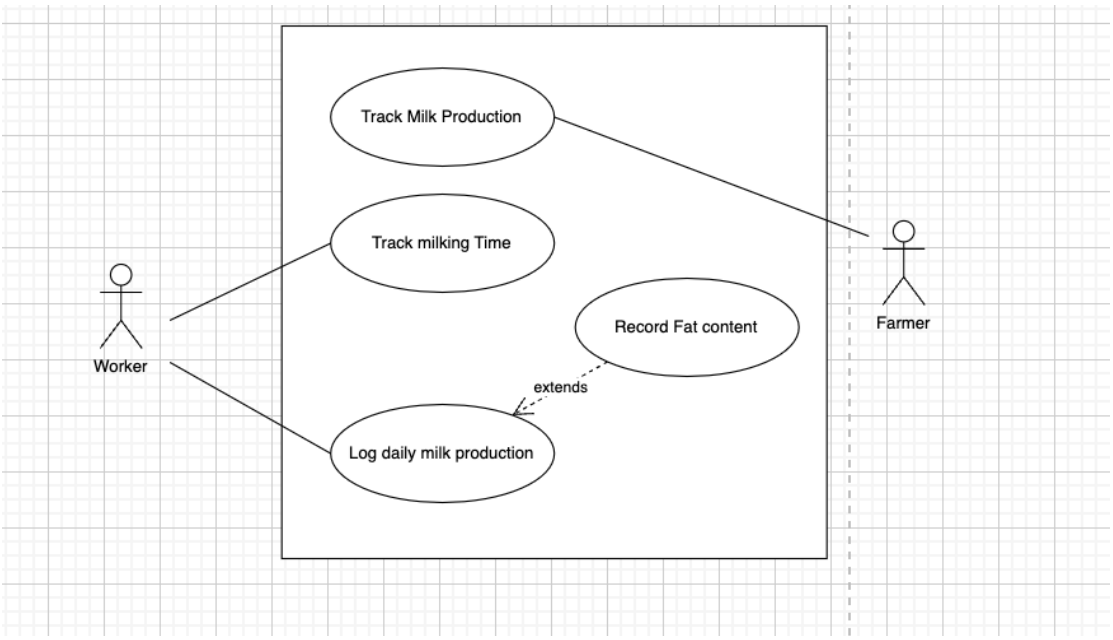
1. **User-Friendly GUI**
    - **Built using Java Swing for ease of navigation and visual interaction.**
2. **Cow Records Management**
    - **Add, update, delete, and view detailed cow profiles including tag number, breed, weight, health status, and birth date.**
3. **Milk Production Tracking**
    - **Record daily or periodic milk yield for individual cows and view trends.**
4. **Health Monitoring**
    - **Track vaccination dates, health checkups, and treatments administered.**
5. **Database Integration**
    - **Stores data persistently using a backend database (likely MySQL or SQLite).**
6. **Search & Filter**
    - **Search cows by ID or filter by breed, age group, or health status.**

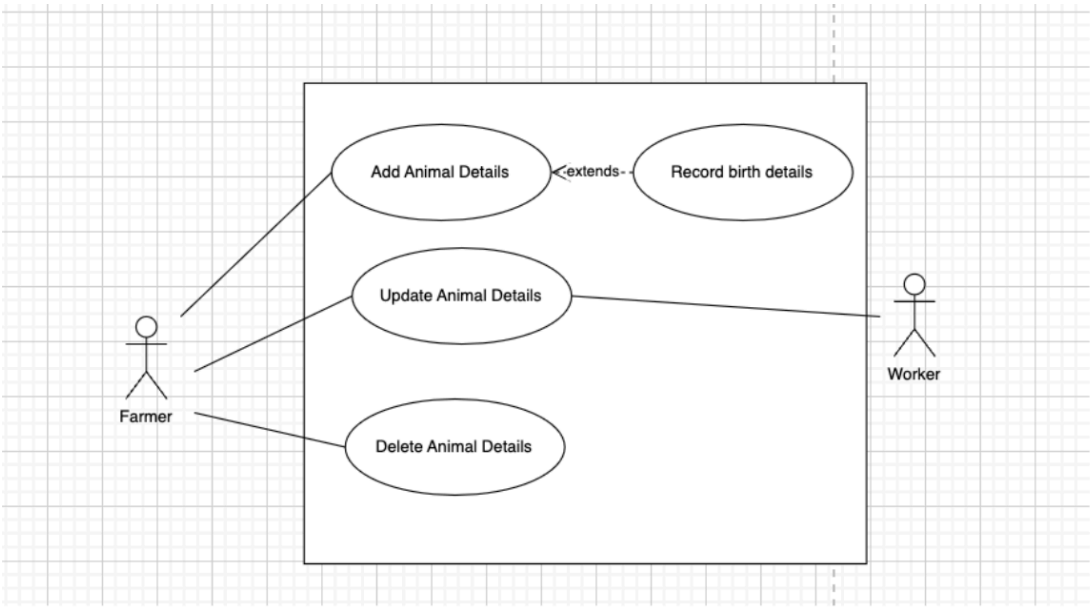
Models:

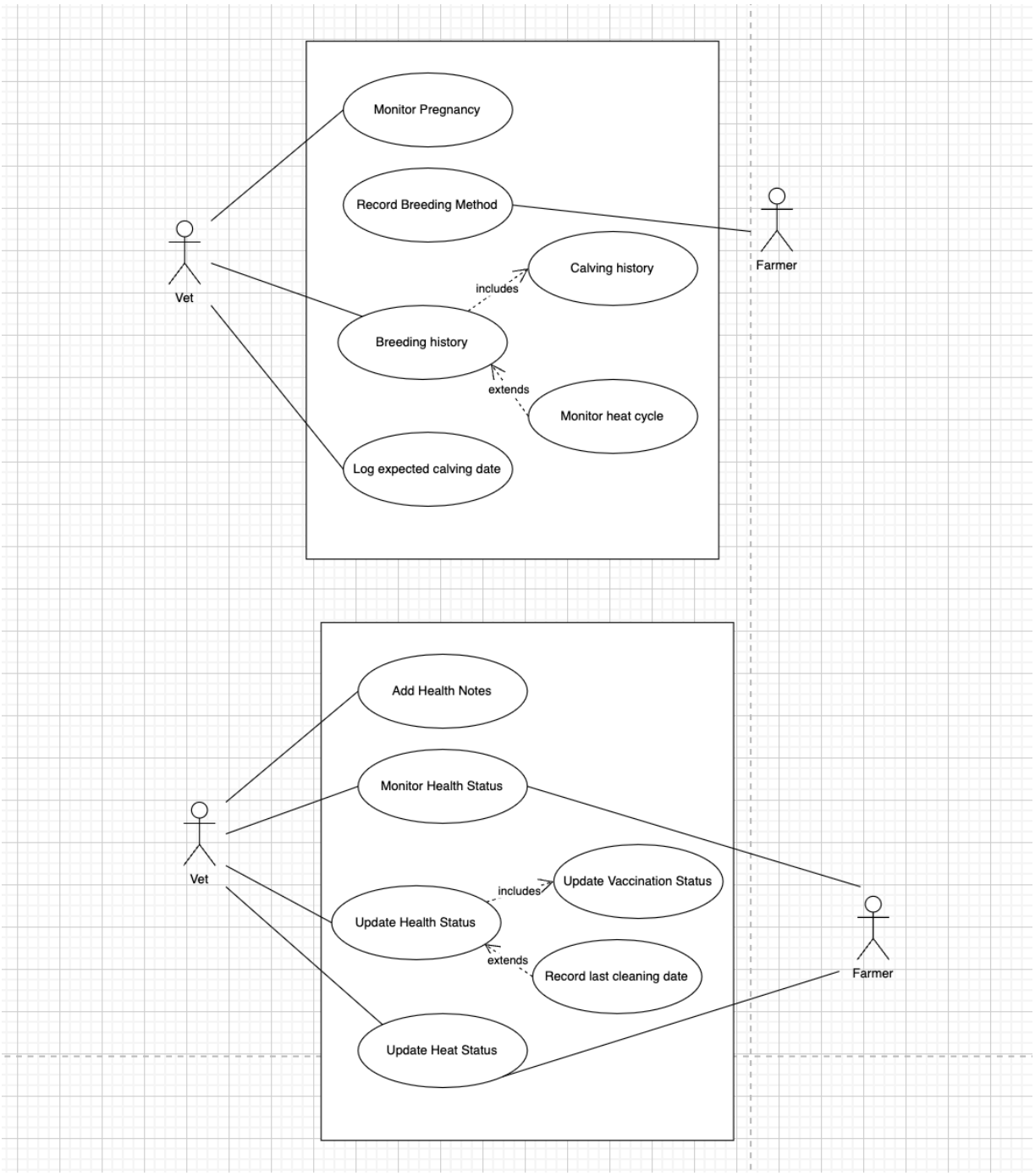
Use Case Diagram:



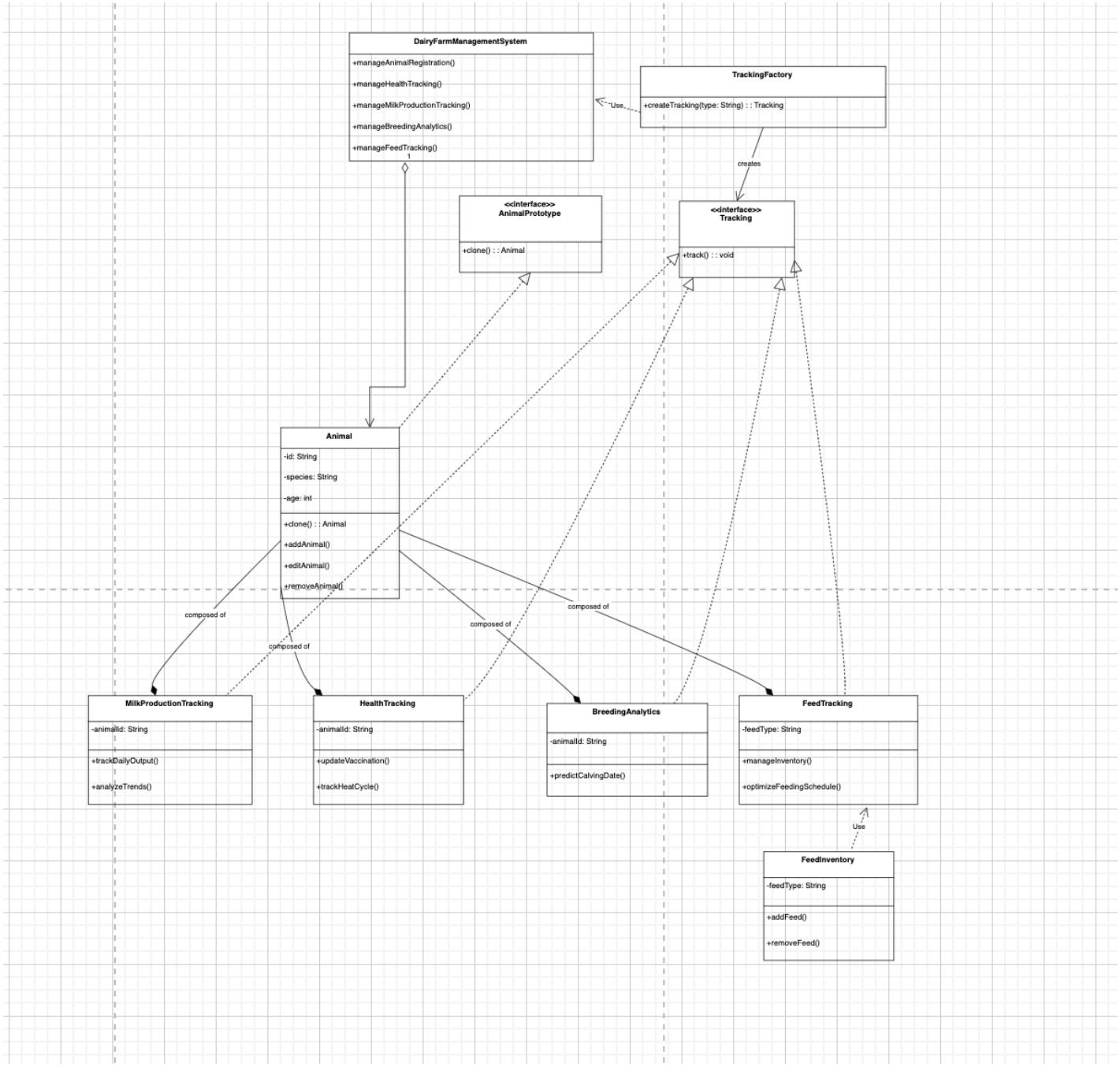



Track Milk Production
Track milking Time
Record Fat content
extends
Log daily milk production
Worker
Farmer

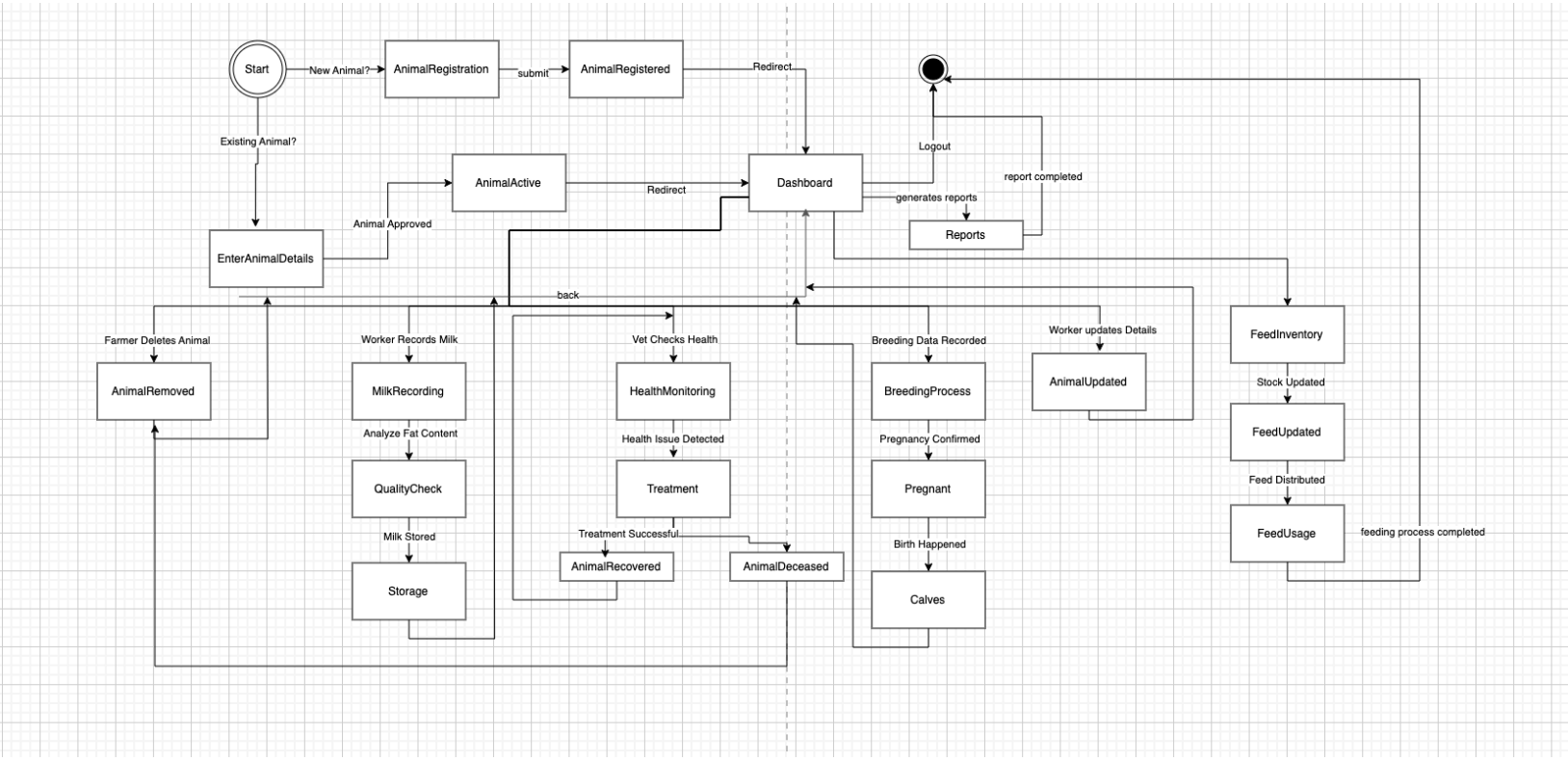

Add Animal Details
extends
Record birth details
Update Animal Details
Delete Animal Details
Farmer
Worker


Monitor Pregnancy
Record Breeding Method
Calving history
includes
Breeding history
extends
Monitor heat cycle
Log expected calving date
Vet
Farmer
Add Health Notes
Monitor Health Status
Update Vaccination Status
includes
Update Health Status
extends
Record last cleaning date
Update Heat Status
Vet
Farmer

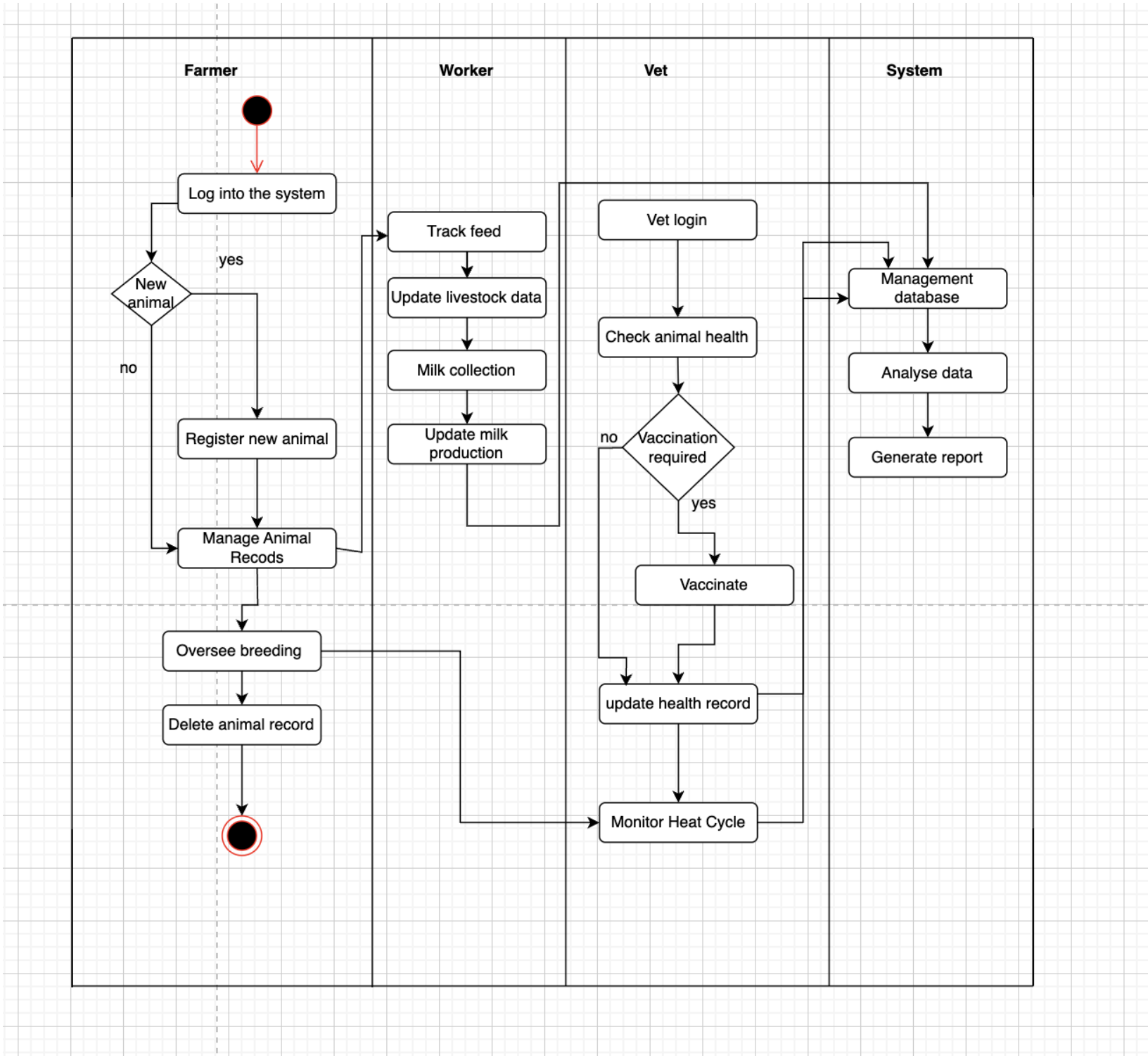
Class Diagram:



State Diagram:



Activity Diagrams:



Architecture Patterns

Model – View – Controller Pattern (MVC)

1. Model (Business Logic and Data Layer)

Models represent your application data and contain logic for storing/retrieving it.

Reference:

- src/main/java/com/example/dairyfarm/model/BreedingAnalytics.java
- src/main/java/com/example/dairyfarm/model/Animal.java
- src/main/java/com/example/dairyfarm/model/MilkRecord.java

2. The Controller handles user input, processes requests, invokes model logic, and determines the appropriate view.

Reference:

- src/main/java/com/example/dairyfarm/controller/UserController.java
- src/main/java/com/example/dairyfarm/controller/FeedController.java
- src/main/java/com/example/dairyfarm/controller/BreedController.java

3. View – Represents User Interface

The View is built using JSP (JavaServer Pages). It is responsible for rendering the UI and displaying data passed from the controller.

Reference:

- src/main/java/com/example/dairyfarm/service/AnimalService.java
- src/main/java/com/example/dairyfarm/service/FeedService.java
- src/main/java/com/example/dairyfarm/service/MilkService.java

Design Principles

1. Separation of Concerns: By dividing the application into Model, View, and Controller, each part handles a specific aspect of the application, enhancing clarity and maintainability.
2. DRY (Don't Repeat Yourself): Common functionalities are abstracted to avoid code duplication, making the codebase more efficient and easier to manage.
3. KISS (Keep It Simple, Stupid): The code avoids unnecessary complexity, focusing on straightforward solutions that are easy to understand and maintain.
4. Single Responsibility Principle: Each class or module has a single, well-defined responsibility, reducing the risk of errors and making the system more robust.
5. GRASP principles

Design Patterns

1. **Factory Pattern**: If your application creates objects without specifying the exact class, especially for different types of cows or users, this pattern is in use. It provides a way to encapsulate object creation, promoting flexibility and scalability.

   Cow.java

```java
package com.example.dairyfarm.service;

import com.example.dairyfarm.model.*;

public class TrackingFactory {

    public static Tracking getTracking(String type) {
        switch (type.toLowerCase()) {
            case "milk":
                return new MilkProductionTracking();
            case "feed":
                return new FeedTracking();
            case "health":
                return new HealthTracking();
            default:
                throw new IllegalArgumentException("Unknown tracking type: " + type);
        }
    }
}
```

2. **Singleton Pattern**: If there's a class ensuring only one instance exists (e.g., a database connection manager), this pattern is applied. It controls object creation, limiting the number of instances to one, ensuring consistent access to resources.

   DBConnection.java

```java
public class DBConnection {
    private static DBConnection instance;
    private Connection connection;

    private DBConnection() {
        connection = DriverManager.getConnection();
    }

    public static synchronized DBConnection getInstance() {
        if (instance == null) {
            instance = new DBConnection();
        }
        return instance;
    }

    public Connection getConnection() {
        return connection;
    }
}
```

3. **Observer Pattern**: If your application updates the UI automatically in response to data changes (e.g., real-time updates of cow health status), this pattern is utilized. It defines a one-to-many dependency between objects, so when one object changes state, all its dependents are notified.

   VetVisit.java

```java
public interface VetVisit {
    void update(String message);
}

public class CowHealthMonitor implements Observer {
    public void update(String message) {
        System.out.println("Health Alert: " + message);
    }
}

public class Cow {
    private List<Observer> observers = new ArrayList<>();

    public void addObserver(Observer observer) {
        observers.add(observer);
    }

    public void notifyObservers(String message) {
        for (Observer observer : observers) {
            observer.update(message);
        }
    }

    public void checkHealth() {
        // Health check logic
        notifyObservers("Cow health is critical!");
    }
}
```

4. **Strategy Pattern**: If your application selects algorithms at runtime (e.g., different feeding strategies for cows), this pattern is evident. It enables selecting an algorithm's behavior at runtime, promoting flexibility and reusability.

   FeedTracking.java

```java
public interface FeedTracking {
    void feed(Cow cow);
}

public class HighProteinFeed implements FeedingStrategy {
    public void feed(Cow cow) {
        System.out.println("Feeding " + cow.getName() + " with high-protein feed.");
    }
}

public class CowFeeder {
    private FeedingStrategy strategy;

    public void setStrategy(FeedingStrategy strategy) {
        this.strategy = strategy;
    }

    public void executeFeeding(Cow cow) {
        strategy.feed(cow);
    }
}
```

Github link to the Codebase:

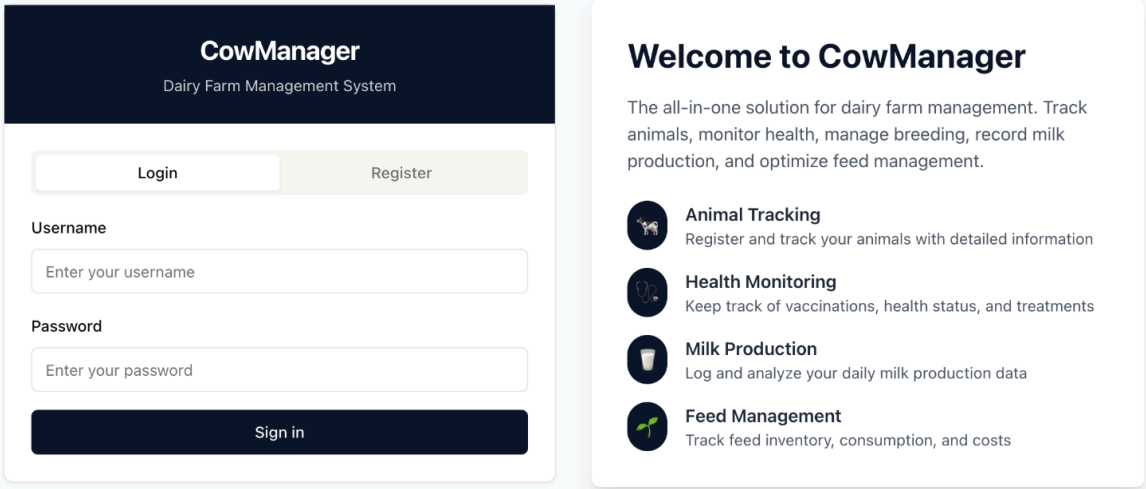
https://github.com/not-ad-chaos/Cow-Management-System

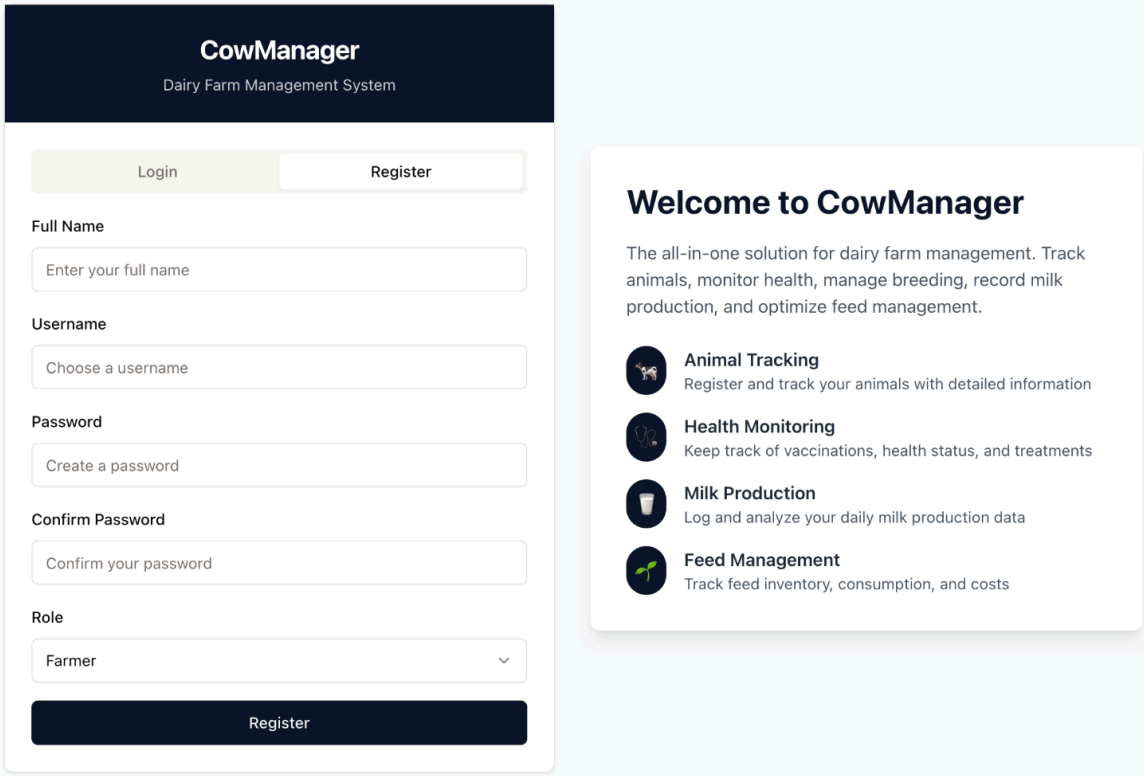
Screenshots:

UI:

```
node_modules/  public/  src/  jsconfig.json  next.config.mjs  package-lock.json  package.json  postcss.config.mjs  README.md
Cow-Management-System/frontend U main ζ npm run dev

> cow-view@0.1.0 dev
> next dev

   ▲ Next.js 15.2.4
   - Local:        http://localhost:3000
   - Network:      http://192.168.68.103:3000

 ✓ Starting...
 ✓ Ready in 1415ms
```

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.2.0)

2025-04-22T15:54:19.381+05:30  INFO 29805 --- [           main] c.e.dairyfarm.DairyFarmApplication       : Starting DairyFarmApplication using
Java 23.0.2 with PID 29805 (/Users/kiranrajpurohit/git-repos/not-ad-chaos/Cow-Management-System/dairy-farm-management/target/classes started by
 kiranrajpurohit in /Users/kiranrajpurohit/git-repos/not-ad-chaos/Cow-Management-System/dairy-farm-management)
2025-04-22T15:54:19.382+05:30  INFO 29805 --- [           main] c.e.dairyfarm.DairyFarmApplication       : No active profile set, falling back
to 1 default profile: "default"
2025-04-22T15:54:19.793+05:30  INFO 29805 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA reposi
tories in DEFAULT mode.
2025-04-22T15:54:19.820+05:30  INFO 29805 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scan
ning in 23 ms. Found 7 JPA repository interfaces.
2025-04-22T15:54:20.186+05:30  INFO 29805 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (h
ttp)
2025-04-22T15:54:20.191+05:30  INFO 29805 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-04-22T15:54:20.191+05:30  INFO 29805 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tom
cat/10.1.16]
2025-04-22T15:54:20.217+05:30  INFO 29805 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebAppl
icationContext
2025-04-22T15:54:20.218+05:30  INFO 29805 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initiali
zation completed in 814 ms
2025-04-22T15:54:20.387+05:30  INFO 29805 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUni
tInfo [name: default]
2025-04-22T15:54:20.417+05:30  INFO 29805 --- [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core versio
n 6.3.1.Final
2025-04-22T15:54:20.434+05:30  INFO 29805 --- [           main] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cache disabl
ed
2025-04-22T15:54:20.556+05:30  INFO 29805 --- [           main] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JP
A class transformer
```


CowManager
Dairy Farm Management System
Login
Register
Username
Enter your username
Password
Enter your password
Sign in
Welcome to CowManager
The all-in-one solution for dairy farm management. Track animals, monitor health, manage breeding, record milk production, and optimize feed management.
Animal Tracking
Register and track your animals with detailed information
Health Monitoring
Keep track of vaccinations, health status, and treatments
Milk Production
Log and analyze your daily milk production data
Feed Management
Track feed inventory, consumption, and costs


CowManager
Dairy Farm Management System
Login
Register
Full Name
Enter your full name
Username
Choose a username
Password
Create a password
Confirm Password
Confirm your password
Role
Farmer
Register
Welcome to CowManager
The all-in-one solution for dairy farm management. Track animals, monitor health, manage breeding, record milk production, and optimize feed management.
Animal Tracking
Register and track your animals with detailed information
Health Monitoring
Keep track of vaccinations, health status, and treatments
Milk Production
Log and analyze your daily milk production data
Feed Management
Track feed inventory, consumption, and costs


localhost:3000

Milk Production Management

Total Milk Production
7.00 liters

Total Records
3

Average Production
2.33 liters

Add New Milk Record

Animal ID *
Enter Animal ID

Date *
dd-mm-yyyy

Quantity (liters) *
Enter milk quantity in liters

Add Record

Filter Records

Animal ID
Filter by Animal ID

Date
dd-mm-yyyy

Reset Filters

Milk Production Records

Animal ID	Date	Quantity (liters)	Actions
A001	2025-04-21	2.00	Edit Delete
A001	2025-04-22	3.00	Edit Delete
A001	2025-04-11	2.00	Edit Delete

Cow Breeding Management

Breeding Analytics | Breeds

Add New Breeding Record

Animal ID
Enter Animal ID

Breed Name
Enter Breed Name

Breeding Date
dd-mm-yyyy

Outcome
Enter outcome

Notes
Additional notes

☐ Successful

Add Record

Breeding Analytics Records

Animal ID	Breed Name	Breeding Date	Outcome	Notes	Successful	Actions
A001	Holstein Friesian	4/28/2025	nil		No	Edit Delete

Animal Registration

Register New Animal

Animal ID

Name

dd/mm/yyyy

Breed

Select Gender

Notes

Register Cancel

Registered Animals

ID	Name	DOB	Breed	Gender	Actions
5	A Cow	12/04/2025	Holstien	Male	Edit Delete
9	Cowette	23/04/2025	Jersey	Female	Edit Delete

Vet Visit Management

Add New Vet Visit

Animal ID

Enter Animal ID

Vet's Name

Enter Vet's Name

Date of Visit

dd/mm/yyyy

Diagnosis

Enter diagnosis or notes

Treatment

Prescribed treatment

Add Record

Vet Visit Records

Animal ID	Vet's Name	Date of Visit	Diagnosis	Treatment
1	Smith	2025-04-09	Viral Diarrhea	Pepto Bismol
3	Ramu	2025-04-11	Healthy	NA

Feed Management

Feed Inventory

Feed Inventory

Add New Feed

ID	Feed Type	Total Stock (Kg)
1	Hay	30
2	Grass	50

Refresh Data

Feed Checklist

Feed Consumption Tracking

Date:

22/04/2025

- [ ] Hay – 30 Kg available
- [ ] Grass – 50 Kg available

Submit Feed Consumption

```
mysql> use dairy_farm_db;
Database changed
mysql> select * from breed;
+----+------------------+----------------+----------------+--------+------------+
| id | breed_name       | characteristics | milk_production | origin | temperament |
+----+------------------+----------------+----------------+--------+------------+
|  5 | Holstein Friesian | friendly       | 5              | Dutch  | -          |
+----+------------------+----------------+----------------+--------+------------+
1 row in set (0.01 sec)

mysql> select * from breeding_analytics;
+----+----------+------------------+--------------------------+-------+---------+----------------------+
| id | animal_id | breed_name       | breeding_date            | notes | outcome | successful           |
+----+----------+------------------+--------------------------+-------+---------+----------------------+
|  2 | A001     | Holstein Friesian | 2025-04-28 05:30:00.000000 |       | nil     | 0x00                 |
+----+----------+------------------+--------------------------+-------+---------+----------------------+
1 row in set (0.00 sec)
```

```
mysql> select * from animals;
+----+-----------+----------+------------+--------+---------+----------------+
| id | animal_id | breed    | dob        | gender | name    | notes          |
+----+-----------+----------+------------+--------+---------+----------------+
|  2 | 5         | Holstien | 2025-04-12 | Male   | A Cow   | Healthy animal |
|  3 | 9         | Jersey   | 2025-04-23 | Female | Cowette | Healthy        |
+----+-----------+----------+------------+--------+---------+----------------+
2 rows in set (0.00 sec)

mysql> select * from vet_visit;
+----+-----------+---------------+--------------+----------+------------+
| id | animal_id | diagnosis     | treatment    | vet_name | visit_date |
+----+-----------+---------------+--------------+----------+------------+
|  1 | 1         | Viral Diarrhea | Pepto Bismol | Smith    | 2025-04-09 |
|  2 | 3         | Healthy       | NA           | Ramu     | 2025-04-11 |
+----+-----------+---------------+--------------+----------+------------+
2 rows in set (0.00 sec)

mysql> select * from feed_inventory;
+----+-----------+-------------+
| id | feed_type | total_stock |
+----+-----------+-------------+
|  1 | Hay       |          30 |
|  2 | Grass     |          50 |
+----+-----------+-------------+
2 rows in set (0.00 sec)
```

```
mysql> select * from milk_record;
+----+------------+----------+-----------+
| id | date       | quantity | animal_id |
+----+------------+----------+-----------+
| 12 | 2025-04-21 |        2 | A001      |
| 14 | 2025-04-22 |        3 | A001      |
| 15 | 2025-04-11 |        2 | A001      |
+----+------------+----------+-----------+
3 rows in set (0.01 sec)

mysql> use dairy_farm_db;
```

Individual contributions of the team members:

Name	Module worked on
Kiran J Rajpurohit **PES2UG22CS265**	**Authentication Module**: Implement login and registration system for 3 user types (Vet, Farmer, Worker) Role-based access control and session management UI design for login/register pages
Manya Singh **PES2UG22CS306**	**Animal Management & Health Module**: CRUD operations for cow details Vet visit tracking, vaccination records, treatment history Integration of Observer pattern for health updates
Madhura H B **PES2UG22CS291**	**Milk Production & Feed Tracking Module**: Record and manage daily milk yield Implement feeding strategies using Strategy Pattern Data analysis for yield trends
Midhushi Mahajan **PES2UG22CS310**	**Breeding Analytics & Database Integration**: Manage breeding cycle records and analytics Setup and manage database schema Implement Singleton pattern for DB connection and data services