Hamdard University
Department of Computing
Final Year Project

Farmware App
(FYP-013/FL24)

Software Design Specifications

Submitted by

Maryam Nadeem(2437-2021)
Muhammad Mujtaba(1985-2021)
Hamza Bin Asif (1961-2021)

Supervisor(s)
Mr. Faheem
Ahmed

Fall 2024

Document Sign off Sheet

1.1.1 Document Information

Project Title	Farmware App
Project Code	FYP-013/FL24
Document Name	Software Design Specifications
Document Version	2.2
Document Identifier	FYP-013/FL24-SDS
Document Status	Final
Author(s)	Maryam Nadeem
Approver(s)	Mr. Faheem Ahmed
Issue Date	17/1/2025

Name	Role	Signature	Date
Maryam Nadeem	Team Lead		16/1/25
Muhammad Mujtaba	Team Member 2		16/1/25
Hamza Bin Asif	Team Member 3		16/1/25
Mr. Faheem Ahmed	Supervisor		
	Project Coordinator		

Revision History

Date	Version	Description	Author
15/12/2024	1.0	Initial	Maryam Nadeem
10/1/2025	1.1	Architectural Implementations	Maryam Nadeem
15/1/2025	1.2	Formatting Diagrams	Maryam Nadeem
02/07/2025	2.2	Feature Enhancement	Maryam Nadeem

Definition of Terms, Acronyms, and Abbreviations

Term	Description
AI	**Artificial Intelligence**
ML	Machine Learning
NLP	Natural Language Processing
DBMS	Database Management System

2 Table of Contents

3 Introduction

3.1 Purpose of Document

The purpose of this Software Design Specification (SDS) document is to define the architecture, components, data design, and interaction mechanisms of the Farmware application. This document provides a comprehensive blueprint for developers, testers, and other stakeholders to ensure the system is built efficiently and aligns with the functional and non-functional requirements outlined in the Software Requirements Specification (SRS). Additionally, it serves as a guide for the development team to implement and maintain the system effectively.

3.2 Intended Audience

Development Team
Project Supervisors
Testers
Farmers (End-Users)

3.3 Document Convention

- **Headings**: Arial, Bold, Size 16
- **Subheadings**: Arial, Bold, Size 14
- **Body Text/Descriptions**: Arial, Italic, Size 10
- **Table Fields**: Arial, Regular, Size 12, 10

3.4 Project Overview

Farmware is a mobile application designed to streamline cattle health management for farmers, with a specific focus on Pakistani cow breeds and regional diseases. The application enables users to monitor the daily health of cows, predict potential diseases using machine learning algorithms, and receive tailored recommendations for treatment and prevention. By leveraging data collection and advanced analytics, Farmware helps improve livestock health, productivity, and overall farm management.

3.4.1 Key Functionalities

- **Daily Health Monitoring**: Users can input data about each cow’s health, including symptoms and behaviors, using a user-friendly interface.
- **Disease Prediction**: The system analyzes collected data with machine learning to identify potential health issues early.
- **Recommendations and Alerts**: Provides actionable health advice and real-time alerts for critical health conditions.
- **Health History Tracking**: Users can view historical data to track trends and monitor cow recovery.

3.4.2 Basic Design Approach

The software will be developed using the **Ionic Framework** combined with **Angular**, which provides a modular, reusable structure for the application. This approach breaks the system into components such as data handling, disease prediction, user management, and reporting.

The architecture will follow a **client-server model**, where:

- The **client-side** (mobile app) is built using **Ionic Framework** with **Angular** for cross-platform compatibility, targeting **Android**.
- The **server-side** handles data storage and processing using **Node.js** and **MySQL** for efficient data management.
- **Machine Learning algorithms** will be integrated using **Scikit-learn** or **TensorFlow** on the server to perform data analysis and disease prediction.

3.5 Scope

Farmware will:

- *Track cow health daily.*
- *Predict diseases using ML algorithms.*
- *Offer tailored advice based on symptoms.*
- *Provide recovery monitoring.*

Not in Scope

- *Integration with external farm management systems.*
- *Real-time syncing with external medical databases.*
- *Replacing professional veterinary diagnosis.*

4 Design Considerations

Issue: As Farmware is aimed at farmers who may not be tech-savvy, the user interface must be intuitive, simple, and easy to navigate.

- **Consideration**: The design should prioritize clarity and ease of use, especially for older generations or those not familiar with complex digital tools. It's essential to minimize user input errors by using clear labels, error messages, and easy-to-understand options.
- **Action**: Focus on a minimalistic and responsive design. The use of large buttons, clear navigation, and a well-organized layout will ensure accessibility.

Issue: The large volume of health data, including cow history and symptom tracking, must be stored efficiently.

- **Consideration**: The backend needs a well-structured database to store health data, cow profiles, and disease predictions. **Scalability** is key to ensuring the system can handle large amounts of data as the number of users grows.

- **Action**: Use a **relational database** like **MySQL** or **PostgreSQL** to store structured data. Consider implementing **data backup** and **restore mechanisms** to ensure data integrity and recovery in case of system failures.

Issue: Disease prediction based on symptoms is a complex task that requires a high degree of accuracy and reliability.

- **Consideration**: Implementing machine learning (ML) algorithms for disease prediction requires accurate training data and fine-tuning to avoid false positives or negatives.
- **Action**: Collaborate with veterinarians and domain experts to collect **high-quality training data**. Consider using algorithms like **decision trees** or **neural networks** (e.g., **TensorFlow** or **Scikit-learn**) to analyze historical health data and predict diseases.

Issue: As the app grows, the backend may face performance challenges, especially with large datasets and concurrent users.

- **Consideration**: Ensure that the backend is optimized to handle multiple simultaneous connections and that it can scale with increasing usage.
- **Action**: Implement **caching** for frequently accessed data and use **load balancing** for the server to manage high traffic. Consider optimizing queries and using indexing in the database for faster access to records.

Issue: Farmers may not be familiar with using apps or new technology, making training and support an essential part of the system.

- **Consideration**: Provide clear instructions, **tooltips**, and tutorials within the app. Offering customer support channels (e.g., chat or phone support) is also crucial.
- **Action**: Include a **help section** with frequently asked questions (FAQs), video tutorials, and in-app tips to guide new users.

4.1 Assumptions and Dependencies

4.1.1 Modular Design for Future Extensibility

- **Assumption**: The system will need to accommodate future functionality, such as adding new livestock species, crop management features, or advanced analytics (e.g., predictive maintenance for farming equipment).
  - **Design Impact**: The system should be designed with **modular components**, allowing future features to be added or updated without significant changes to the core architecture.
  - **Solution**: Use a **microservices architecture** for the backend, breaking down the system into independent modules for different functionalities (e.g., cow health tracking, disease prediction, user management, etc.). This ensures that new features can be integrated without disrupting existing features.

4.1.2 Machine Learning Model Management

- **Assumption**: Machine learning models will be used for disease prediction, and they will need to be retrained and updated over time to remain accurate and relevant.
    - **Design Impact**: The backend must allow for easy **model updates and versioning** without requiring downtime or disruption to the application.
    - **Solution**: Implement **model versioning** and **continuous model training** pipelines. Use a **Dockerized environment** or a **model management service** like **TensorFlow Serving** to manage model deployment and updates without downtime.

4.1.3 User Support and Feedback

- **Assumption**: As farmers might not be familiar with technology, there will be a need for user support and feedback channels within the app.
    - **Design Impact**: The system will require built-in support mechanisms to address issues, answer questions, and gather feedback for continuous improvement.
    - **Solution**:
        - Implement **in-app chat** or **FAQ sections** for self-help.
        - Include **contact support** options through email or a ticketing system.

4.2 Risks and Volatile Areas

4.2.1 Changing User Requirements

Farmers may request new features or changes in existing functionalities after the initial release, such as more detailed health reports or the ability to add multiple types of livestock.
Mitigation Strategy:
A feedback mechanism will be integrated to gather user suggestions continuously.

4.2.2 Database Structure Changes

The system stores detailed health records, which may evolve as new data fields are identified (e.g., new symptoms or treatments). Changes to the database structure can disrupt the existing design if not handled correctly.
Design Approach:
The database will follow a **schema migration approach** to accommodate changes in

the structure without affecting the existing data. The design will also use **normalized relational models** with clearly defined relationships to avoid redundancy and ensure scalability.

4.2.3 User Interface and User Flow Modifications

Based on user feedback, there may be a need to modify the user interface or the way users interact with the system. These changes can impact the underlying data flow and system interactions.

Design Approach:

The system's **design layers will be decoupled**, ensuring that changes to the user interface can be made independently of the core system logic. This separation allows for flexible updates to the user flow without affecting back-end operations.

5 System Architecture

The Farmware system follows a **modular architecture**, where functionality is divided into key components, each responsible for specific tasks. This approach ensures flexibility, scalability, and maintainability.

Overview of Architecture

- Presentation Tier
- Business Logic Tier
- Data Tier

5.1 System Level Architecture

User Interface (UI) Layer

The UI layer allows farmers to input animal details, complete daily health symtpom toggles, and view disease predictions. It prioritizes simplicity to accommodate varying levels of user expertise.

Business Logic Layer

This core layer processes user inputs, manages the flow of data between components. It handles symptom toggles response.

Data Management Layer

The data layer stores and manages health records and disease data. It ensures data integrity and supports schema updates to accommodate future changes.

Machine Learning and Prediction Module

This module analyzes health data to predict diseases using trained models. It updates prediction accuracy over time based on new input data.

Component Interactions

- The **UI Layer** collects data from users.
- The **Business Logic Layer** processes the data and passes it to the **Prediction Module**.
- The **Data Management Layer** stores the data and results.

5.1.1 Interfaces to External Systems

Farmware is a **self-contained system** with no current interfaces to external systems or databases. Data is managed internally to ensure reliability and independence from third-party tools. However, the system is designed to support future integrations through well-defined API interfaces if required.

5.1.2 Major Physical Design Issues

- **Device Compatibility:** The system will be compatible with mobile and tablets.

- **Cloud-Based Data Storage:** Data processing and storage will be handled on a cloud server to ensure scalability and reliability.

- **Internet Dependency:** The app requires an active internet connection for data synchronization and disease prediction.

5.1.2 Error Handling:

- **Input Validation:** As farmers toggle symptoms manually, the system will validate that at least one symptom has been selected and ensure the input is saved correctly. The system will also check for data consistency.

- In case of unexpected issues such as network failures or server errors, the app will notify the user with a message like: *"There was an issue with saving your data. Please check your internet connection."* The app should handle such failures gracefully without interrupting the symptom toggling process, allowing the farmer to retry submission without losing previously selected symptoms.

- **Internet Connectivity Issues:** Since connectivity is a key factor, the app will handle errors related to a lack of internet by saving data locally and notifying the user to sync once the connection is restored.

5.2 Software Architecture

The Farmware system follows a layered architecture to ensure scalability, maintainability, and separation of concerns. The architecture consists of three main layers: **User Interface Layer**, **Middle Tier (Business Logic)**, and **Data Access Layer**.

5.2.1.1 User Interface Layer

- The User Interface (UI) Layer provides the front-end experience for farmers. It collects data through symptom toggle and displays disease predictions.
- Interaction: This layer communicates with the Business Logic Layer to pass user inputs and display responses.

5.2.1.2 Middle Tier (Business Logic Layer)

- This is the core of the application where the business rules, symptom toggle logic, and disease prediction algorithms are executed.
- Interaction: It processes the data received from the UI and sends requests to the Data Access Layer for data storage and retrieval. It also interacts with external services (like ML models) for disease prediction.

5.2.1.3 Data Access Layer

- The Data Access Layer manages interactions with the database, ensuring data integrity and handling requests to read and write health records, cow profiles, and predictions.
- Interaction: It receives requests from the Business Logic Layer to fetch, update, or delete data in the database.

6 Design Strategy

The design strategy focuses on creating a robust, scalable, secure, and user-friendly platform through the following key principles:

6.1.1.1 User Interface Paradigms

- **Strategy:** A minimalistic and intuitive design is adopted, focusing on ease of navigation. The user interface uses clear labels, large buttons, and responsive layouts to cater to farmers with varying levels of technological literacy. Guide users through a dynamic, personalized flow of questions.
- **Reasoning:** An accessible design ensures that the app is usable even by farmers with limited technical expertise, improving adoption rates.

6.1.1.2 Data Management (Storage, Distribution, Persistence)

- **Strategy:** Data is stored in a relational database (MySQL), which supports efficient querying and management of structured data like health records and disease predictions. Backup mechanisms ensure data integrity, while future plans include cloud integration for scalability. Data persistence is managed to enable offline data entry, with synchronization when the internet is available.
- **Reasoning:** Relational databases provide robust and scalable solutions for managing large datasets, while cloud integration ensures the system can scale as usage grows.

6.1.1.3 Concurrency and Synchronization

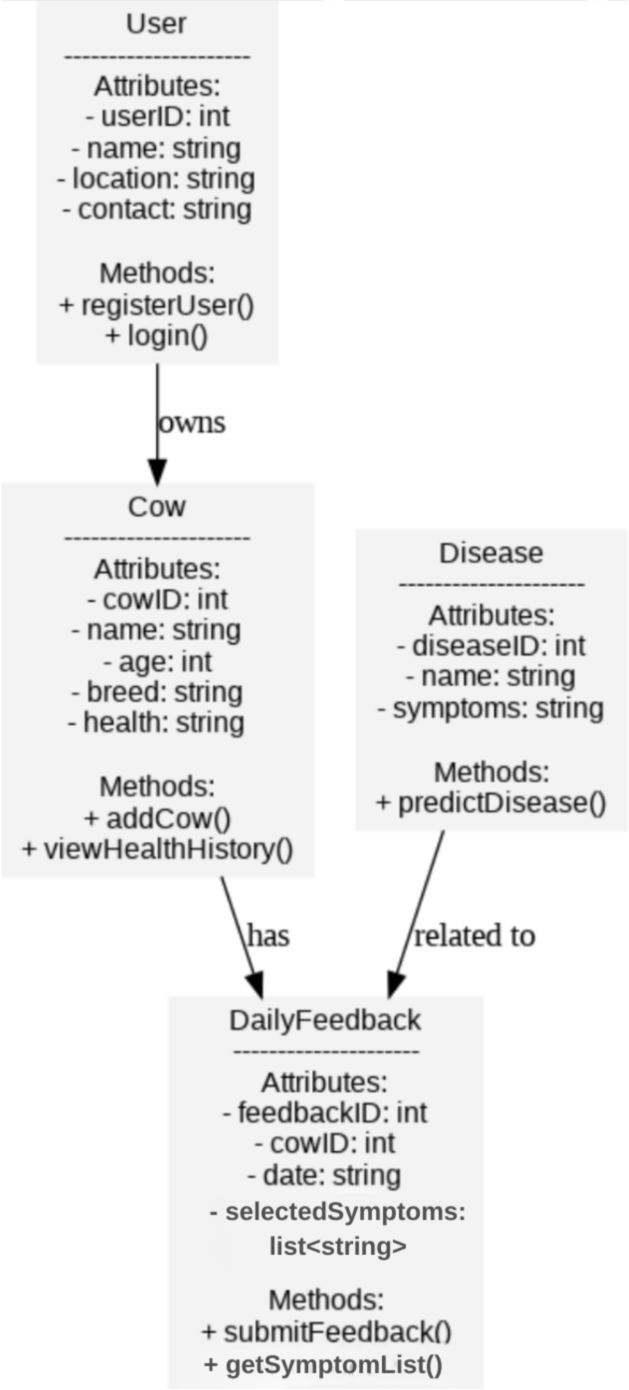
- **Strategy:** The backend is designed to handle concurrent requests efficiently by employing asynchronous processing techniques. This ensures that multiple users can interact with the system simultaneously without performance degradation.
- **Reasoning:** This approach ensures the system remains responsive and reliable, even as the number of users grows. By prioritizing data consistency and integrity, it avoids issues arising from simultaneous data updates or network interruptions.

6.1.1.4 Future System Extension or Enhancement

- **Strategy:** The system is designed to support future extensions by adopting a flexible and scalable architecture. This ensures that new functionalities can be added seamlessly without affecting the core system.
- **Reasoning:** A flexible design approach enhances maintainability and adaptability, allowing the system to evolve in response to changing needs.

7 Detailed System Design

7.1.1 Design Class Diagram



7.1 ER Model Description

- The relationships between the entities are as follows:

- **User – Cow**: One user can own multiple cows (1-to-many).

- **Cow – Daily Feedback**: Each cow can have multiple daily symptom feedback entries (1-to-many).

- **Daily Feedback – Symptoms**: A feedback entry can have multiple symptoms selected, and each symptom can be associated with multiple feedback entries (many-to-many).

- **Diseases – Symptoms**: Many diseases can have many symptoms (many-to-many).

- **Predictions – Diseases**: Each prediction maps a cow to a likely disease based on selected symptoms.

7.2 ER Data Model

Entity	Attributes	Primary Key
Users	UserID, Name, Location, Contact	UserID
Cows	CowID, Name, Age, Breed, Health, UserID	CowID
Symptoms	SymptomID, Name, Category	SymptomID
DailyFeedback	FeedbackID, CowID, Date	FeedbackID
FeedbackSymptoms *(junction table)*	FeedbackID, SymptomID	Composite (FeedbackID, SymptomID)
Predictions	PredictionID, CowID, DiseaseID, Date, Accuracy	PredictionID
Diseases	DiseaseID, Name, Description, Precaution	DiseaseID
DiseaseSymptoms *(junction table)*	DiseaseID, SymptomID	Composite (DiseaseID, SymptomID)

7.2.1 E/R Model Description

7.2.1.1 1. Splash Screen

- **Component:** Logo
- **Description:** Displays the Farmware app logo with a subtle loading animation.
- **Navigation:** Automatically navigates to the Login Page after 2-3 seconds.

7.2.1.2 2. Login Page

- **Components:**
  - Username Input Field
  - Password Input Field
  - Login Button
  - Signup Navigation Button
- **Description:** The page prompts existing users to log in by entering their credentials. If a user is new, they can navigate to the Signup page by pressing the Signup button.
- **Navigation:**
  - Successful login navigates to the Home Page.
  - Signup button navigates to the Signup Page.

7.2.1.3 3. Signup Page

- **Components:**
  - Name Input Field
  - Email Input Field
  - Password Input Field
  - Confirm Password Input Field
  - Register Button
- **Description:** Allows new users to register an account by filling out the necessary details and pressing the Register button.
- **Navigation:**
  - Successful registration navigates to the Home Page.

7.2.1.4 4. Home Page

- **Components:**
  - Header: Displays the app title "Farmware".
  - Cow List: Displays saved cow IDs as clickable entries.
  - Add Button: A floating action button in the bottom right corner to add a new cow.
  - Sidebar Menu: Accessible from a hamburger icon, containing the following navigation options:
    - App Training Page
    - Settings
    - About Us
    - Help
- **Description:** The main dashboard for the user, showing all saved cows. Users can add a new cow using the Add Button or click on an existing cow to view more details.

7.2.1.5 5. Add Cow Page

- **Components:**
  - Input Fields:
    - Cow ID
    - Name
    - Birthday
    - Gender (Dropdown)
    - Breed

- Save Button

- **Description:** A form page to add new cow data. After entering the details and pressing Save, the user is navigated back to the Home Page, where the new cow entry is displayed.

7.2.1.6 6. Cow Details Page

- **Components:**
  - Cow Picture (Placeholder Image if no picture is uploaded)
  - Basic Information Section:
    - Cow ID
    - Name
    - Birthday
    - Gender
    - Breed
  - Navigation Buttons:
    - Symptom Selection
    - Medical Records
- **Description:** Displays the selected cow's details. Users can navigate to either the Symptom Selection page or the Medical Records page using the buttons.

7.2.1.7 7. Symptom Selection Page

- **Components:**
  - Symptom Toggle Buttons
  - Submit Button
- **Description:** The page displays a list of predefined symptoms. Farmers can toggle symptoms on or off based on their observations of the cow. Each toggle represents a symptom being present or not
- **Flow:**
  - The user is shown a categorized list of toggleable symptoms.
  - They select (toggle on) any symptoms they observe.
  - Upon pressing **Submit**, the selected symptoms are recorded in the cow's daily feedback.
  - After submission, the user is navigated to the **Medical Records Page** to review updated health logs and possible disease predictions.

7.2.1.8 8. Medical Records Page

- **Components:**
  - Cow Picture
  - Basic Info Section
  - Medical Report Section

        - Diagnosis
        - Recommended Actions
    - Download Report Button
- **Description:** Displays the health report of the cow based on the symptom selection session. Users can download a copy of the report for their records.

7.2.1.9 9. Sidebar Pages

- **App Training Page:**
    - Video Tutorial Section
    - Text Guide
- **Settings Page:**
    - Account Information
    - Change Password
- **About Us Page:**
    - Brief description of the app's purpose.
    - Contact Information
- **Help Page:**
    - FAQ Section
    - Contact Support Form

7.3 Database Design

7.3.1 Data 1: Users

Name: Users

Alias: User Data

Where-used/how-used:

- Input to user authentication process (Login/Signup)
- Output as user profile data
- Used to link user to farms and cows

Content Description: Stores user information for authentication and identification.

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
UserID	Unique identifier for user	INT	11	NO	AUTO_INCREMENT	PK
Name	Full name of the user	VARCHAR	100	NO	NULL	
Location	Address or region of user	VARCHAR	255	YES	NULL	
Contact	Contact number of user	VARCHAR	15	NO	NULL	

7.3.2 Data 2: Cows

Name: Cows

Alias: Cow Data

Where-used/how-used:

- Input to cow management process
- Output as cow profile data
- Used in Symptom Selection and medical records

Content Description: Stores information about cows owned by users.

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
CowID	Unique identifier for cow	INT	11	NO	AUTO_INCREMENT	PK
Name	Name of the cow	VARCHAR	100	NO	NULL	
Age	Age of the cow in years	INT	3	NO	NULL	
Breed	Breed of the cow	VARCHAR	100	YES	NULL	

Health	Health status description	VARCHAR	255	YES	NULL	

7.3.3 Data 3: Symptoms

Name: Symptoms

Alias: Symptom Data

Where-used/how-used:

- Input to symptom selection process
- Output as symptom report
- Used to predict diseases

Content Description: Stores symptom information related to cows.

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
SymptomID	Unique identifier for symptom	INT	11	NO	AUTO_INCREMENT	PK
Name	Name of the symptom	VARCHAR	100	NO	NULL	
Category	Category of the symptom	VARCHAR	100	YES	NULL	

7.3.4 Data 5: DailyFeedback

Name: DailyFeedback

Alias: Feedback Data

Where-used/how-used:

- Input to feedback process
- Output as feedback report
- Used to track health status

Content Description: Stores daily feedback provided by users.

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
Feedback ID	Unique identifier for feedback	INT	11	NO	AUTO_INCREMENT	PK
CowID	Linked cow ID	INT	11	NO	NULL	FK
Date	Date of feedback	DATE		NO	CURRENT_DATE	
QuestionID	Linked question ID	INT	11	NO	NULL	FK
Answer	User-provided answer	VARCHAR	255	NO	NULL	

7.3.5 Data 6: Predictions

Name: Predictions

Alias: Disease Predictions

Where-used/how-used:

- Input to prediction process
- Output as disease report
- Used to generate medical reports

Content Description: Stores disease predictions based on symptoms feedback.

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
PredictionID	Unique identifier for prediction	INT	11	NO	AUTO_INCREMENT	PK
CowID	Linked cow ID	INT	11	NO	NULL	FK
DiseaseID	Linked disease ID	INT	11	NO	NULL	FK
Date	Date of prediction	DATE		NO	CURRENT_DATE	
Accuracy	Prediction accuracy percentage	FLOAT		YES	NULL	
Actions	Recommended actions	TEXT		YES	NULL	

7.3.6 Data 7: Diseases

Name: Diseases

Alias: Disease Data

Where-used/how-used:

- Input to disease prediction process
- Output as disease report
- Used to link symptoms and provide treatment recommendations

Content Description: Stores disease-related information.

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
DiseaseID	Unique identifier for disease	INT	11	NO	AUTO_INCREMENT	PK
Name	Name of the disease	VARCHAR	100	NO	NULL	
Desc	Disease description	TEXT		YES	NULL	
Symptoms	Related symptoms	TEXT		YES	NULL	
Precaution	Precautionary measures	TEXT		YES	NULL	

7.3.7 Data 8: Actions

Name: Actions

Alias: Action Recommendations

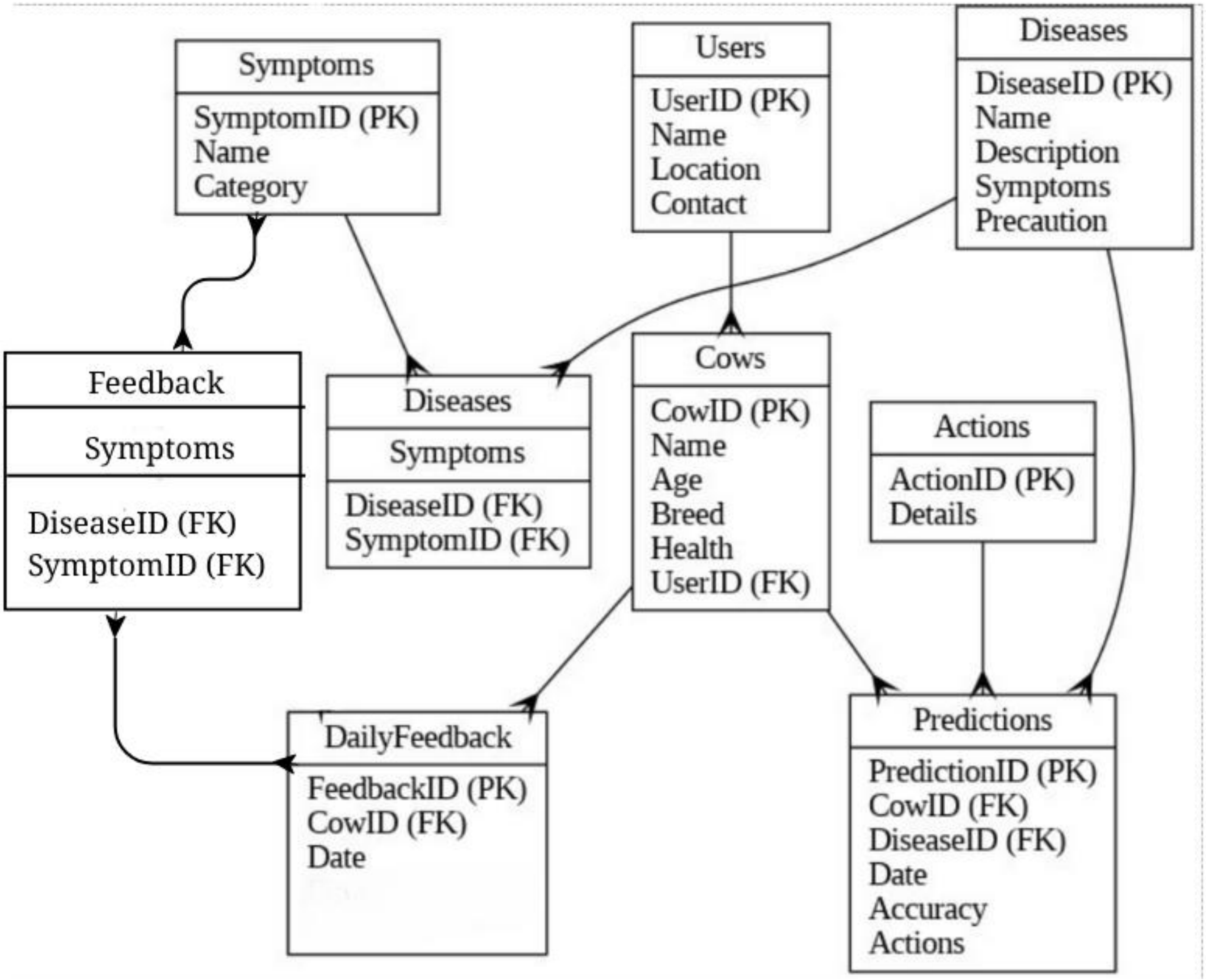
Where-used/how-used:

- Input to prediction process
- Output as recommended actions
- Used in medical reports

Content Description: Stores action recommendations based on predictions.

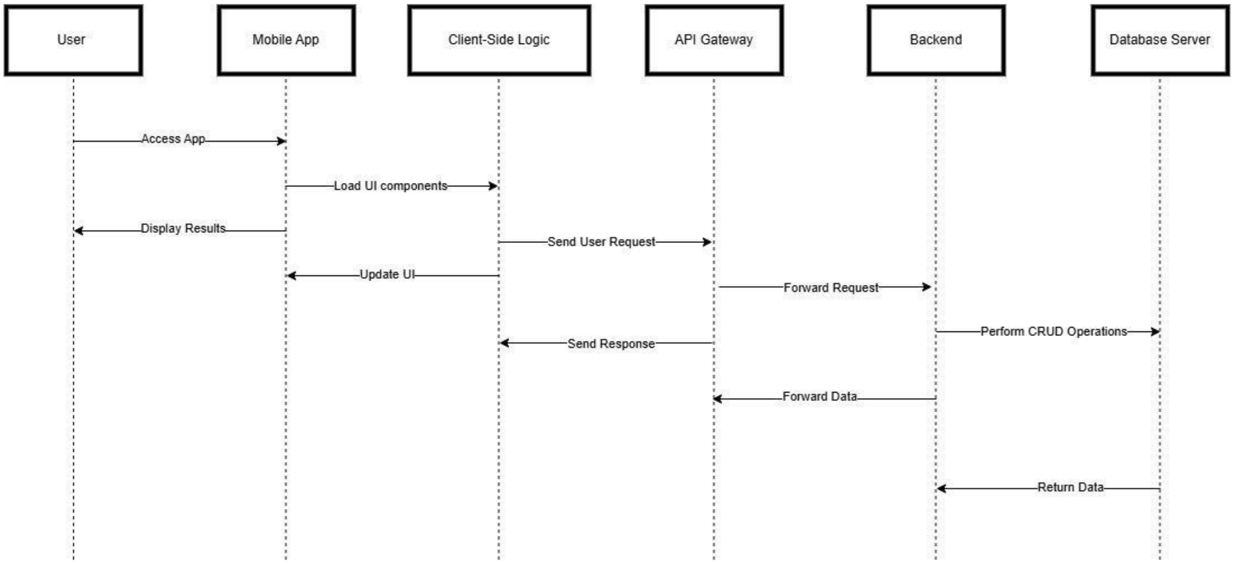
Column Name	Description	Type	Length	Nullable	Default Value	Key Type
ActionID	Unique identifier for action	INT	11	NO	AUTO_INCREMENT	PK
Details	Description of the action	TEXT		NO	NULL	

7.3.8 ER Diagram

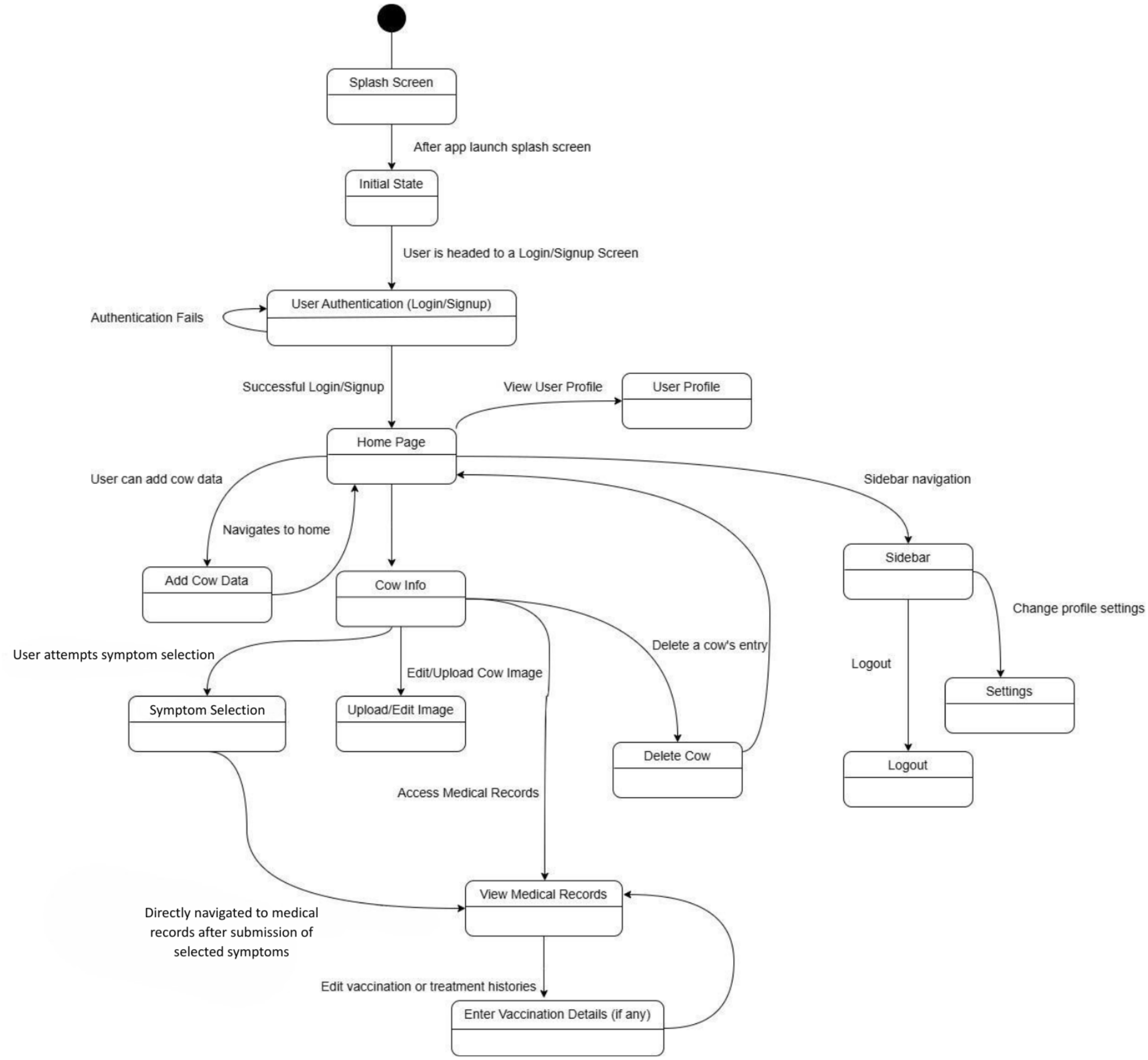


7.4 *Application Design*

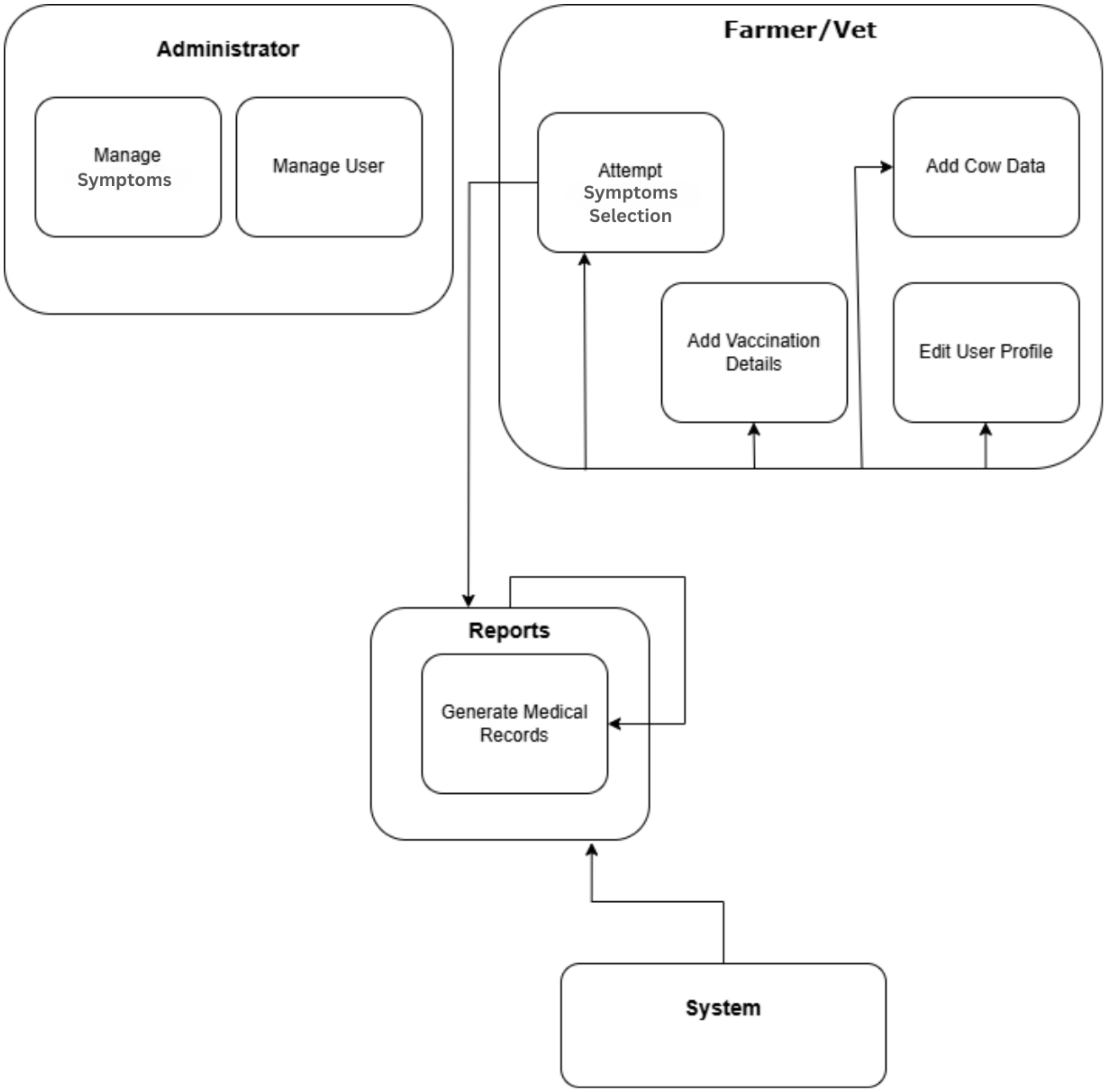
7.4.1 Sequence Diagram



7.4.2 State Diagram

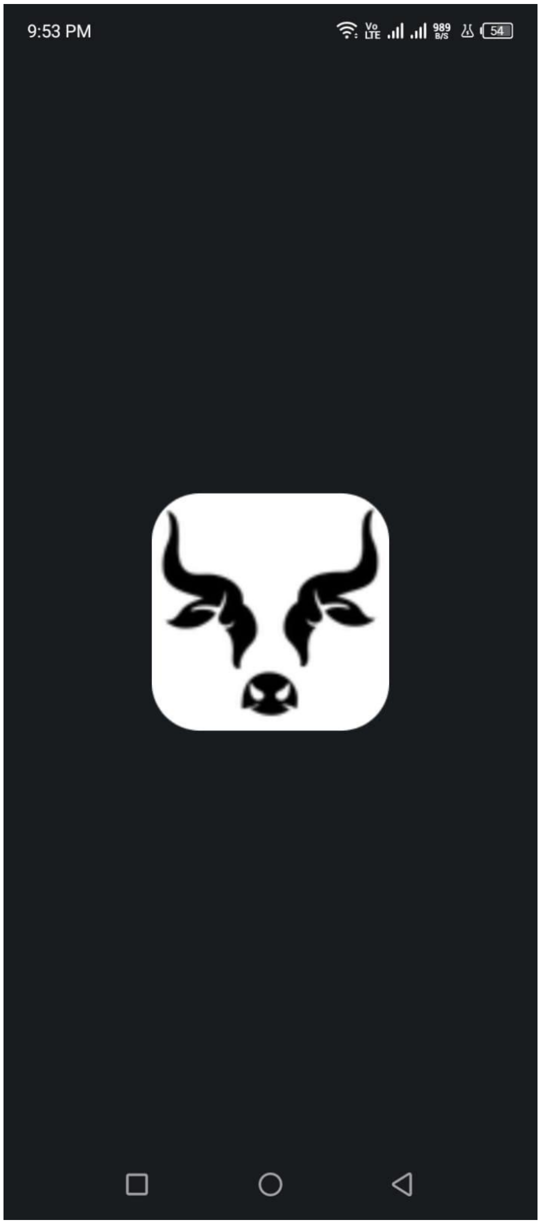


7.4.3 DFD Level 1

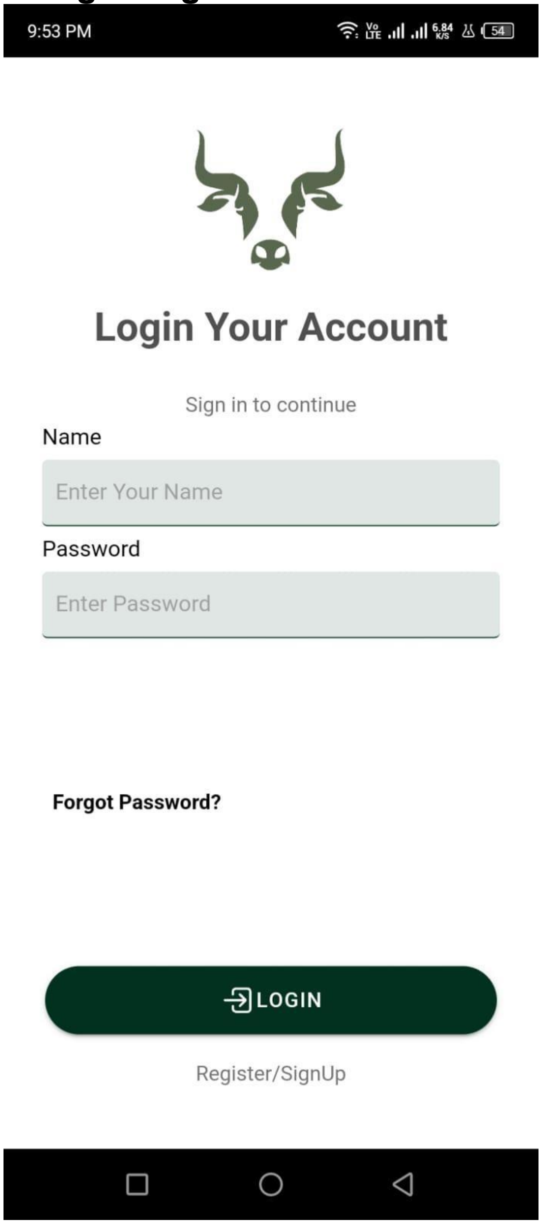


7.5 GUI Design

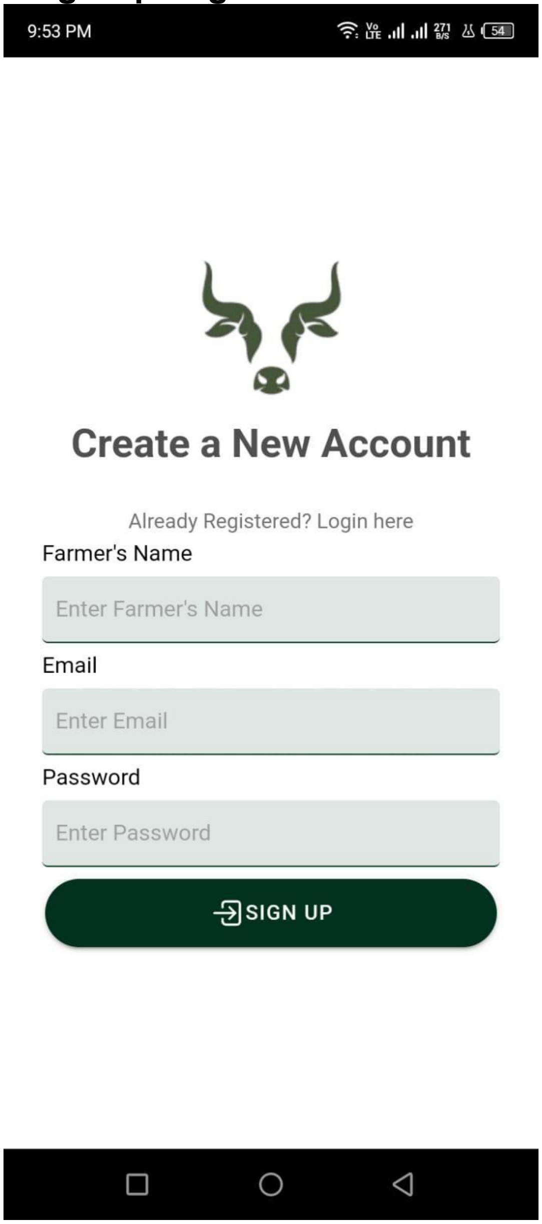
7.5.1 < Splash Screen - Mock Screen 1 >



7.5.2 < Login Page - Mock Screen 2 >



7.5.3 <Sign Up Page - Mock Screen 3 >



Create a New Account

Already Registered? Login here

Farmer's Name

Enter Farmer's Name

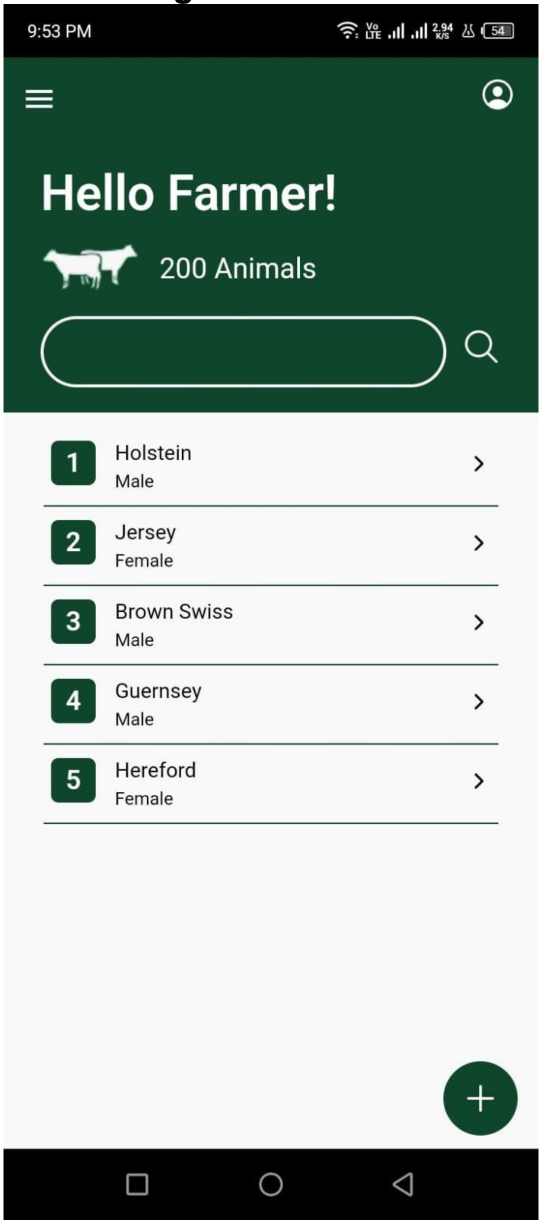
Email

Enter Email

Password

Enter Password

7.5.4 <Home Page - Mock Screen 4 >



Hello Farmer!

200 Animals

1 Holstein
Male

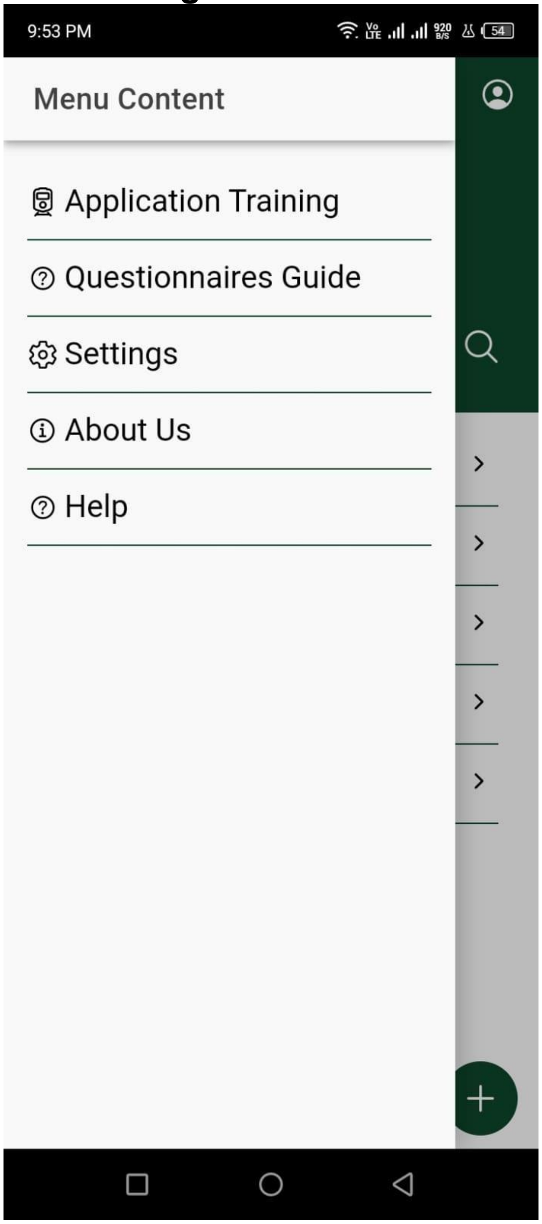
2 Jersey
Female

3 Brown Swiss
Male

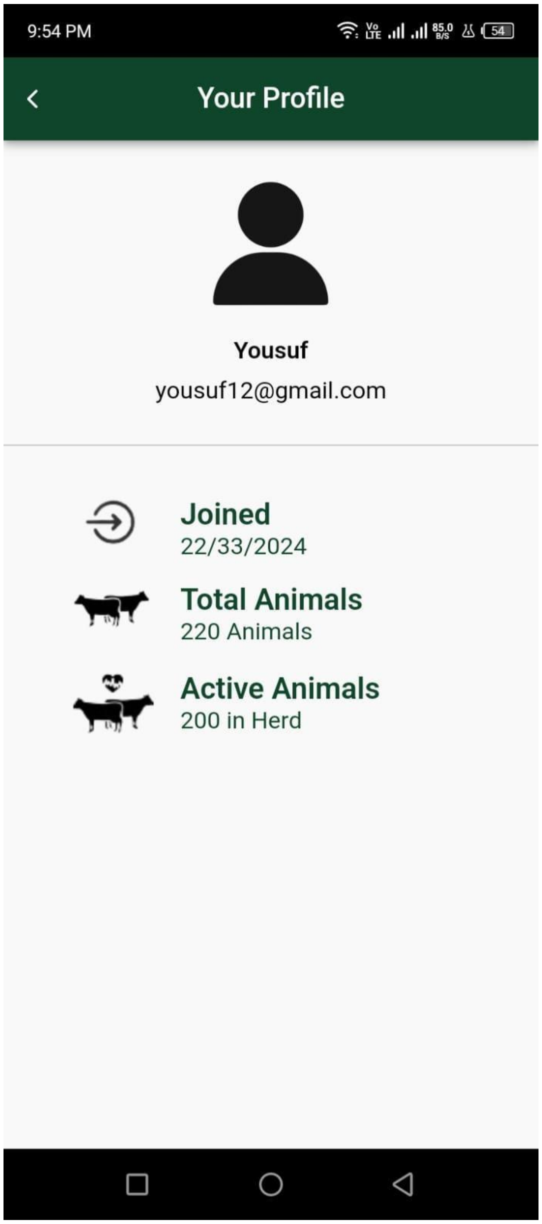
4 Guernsey
Male

5 Hereford
Female

7.5.5 <Side Navigation - Mock Screen 5 >


9:53 PM
Menu Content
Application Training
Questionnaires Guide
Settings
About Us
Help

7.5.6 <Farmer Profile - Mock Screen 6 >



Your Profile

Yousuf

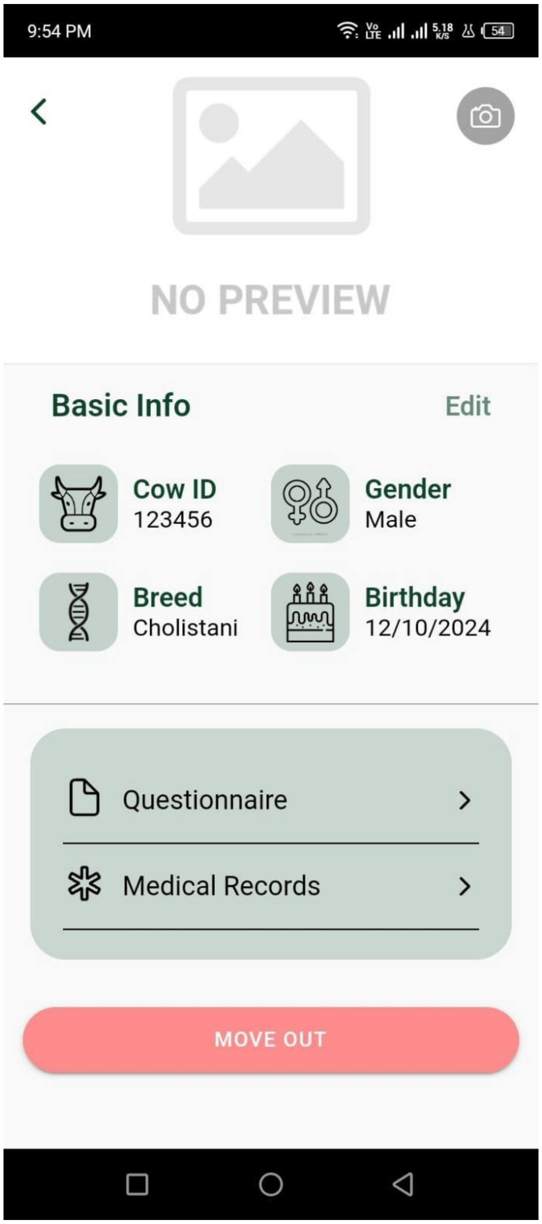
yousuf12@gmail.com

Joined
22/33/2024

Total Animals
220 Animals

Active Animals
200 in Herd

7.5.7 <Cow Profile - Mock Screen 7 >



Basic Info

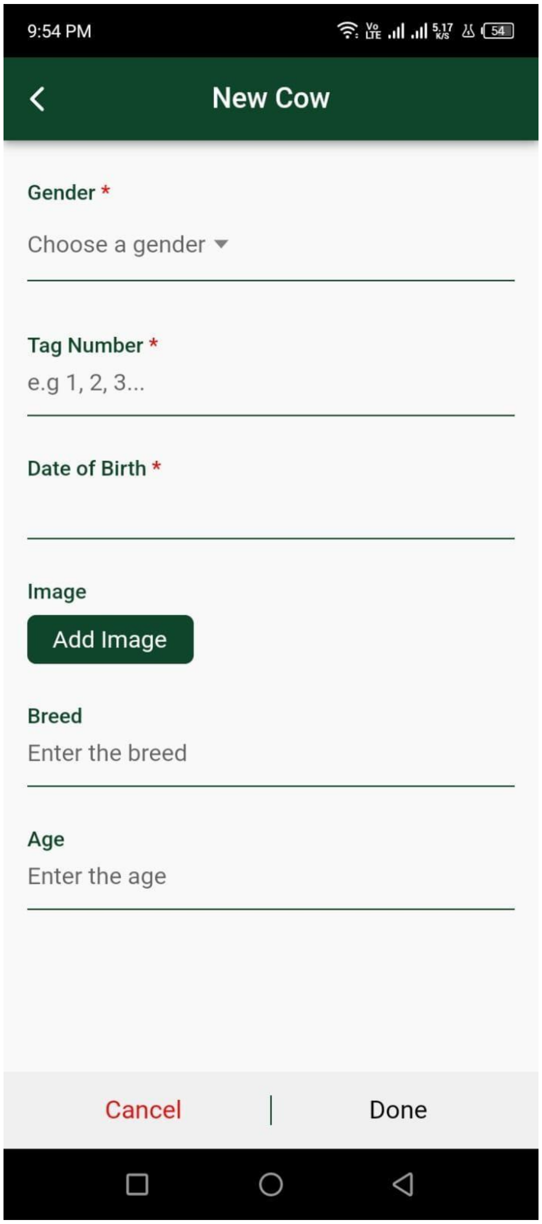
Cow ID
123456

Gender
Male

Breed
Cholistani

Birthday
12/10/2024

7.5.8 <Cow Form - Mock Screen 8 >



New Cow

Gender *

Choose a gender

Tag Number *

e.g 1, 2, 3...

Date of Birth *

Image

Add Image

Breed

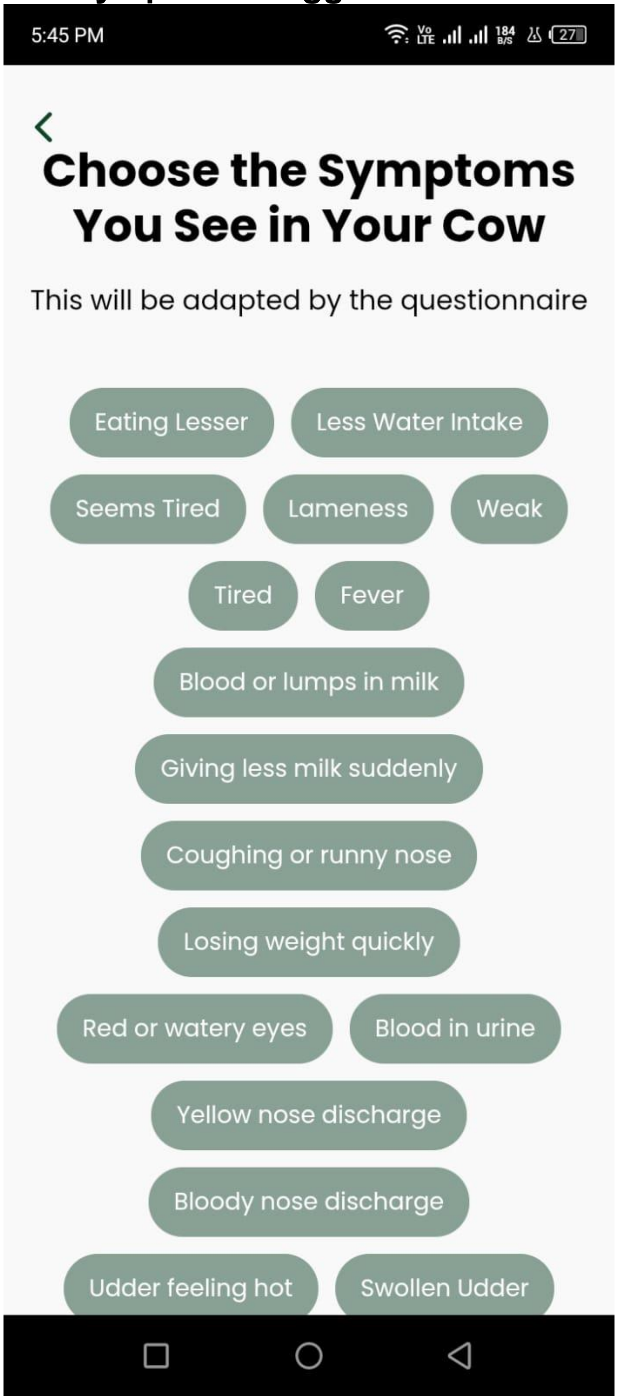
Enter the breed

Age

Enter the age

Cancel | Done

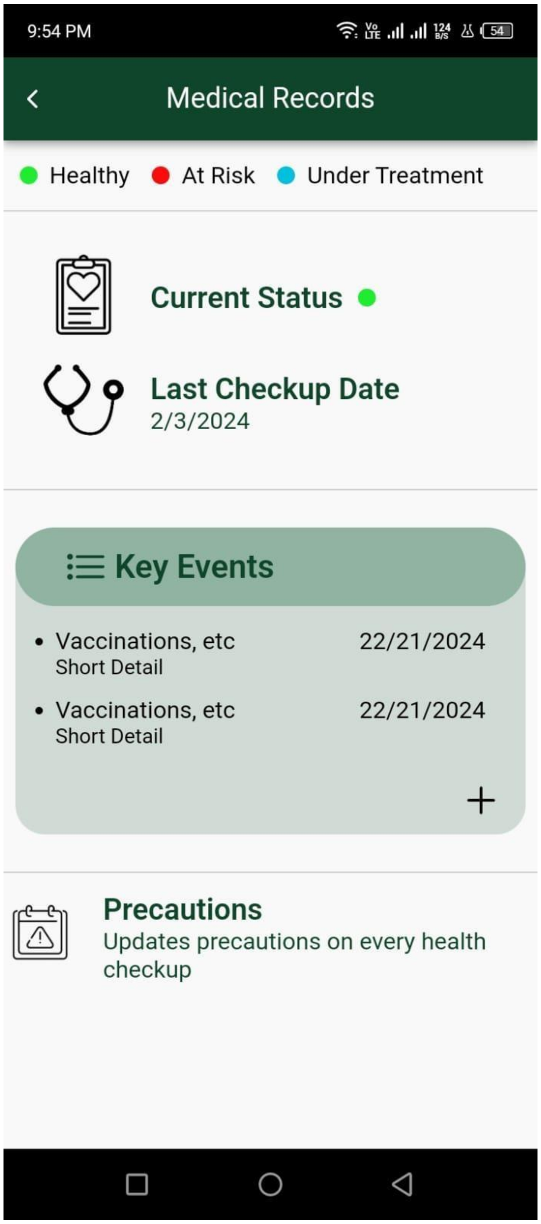
7.5.9 <Symptoms Toggle - Mock Screen 9 >



Choose the Symptoms You See in Your Cow

This will be adapted by the questionnaire

7.5.10 < Medical Records - Mock Screen 10 >



9:54 PM

Medical Records

● Healthy ● At Risk ● Under Treatment

Current Status ●

Last Checkup Date
2/3/2024

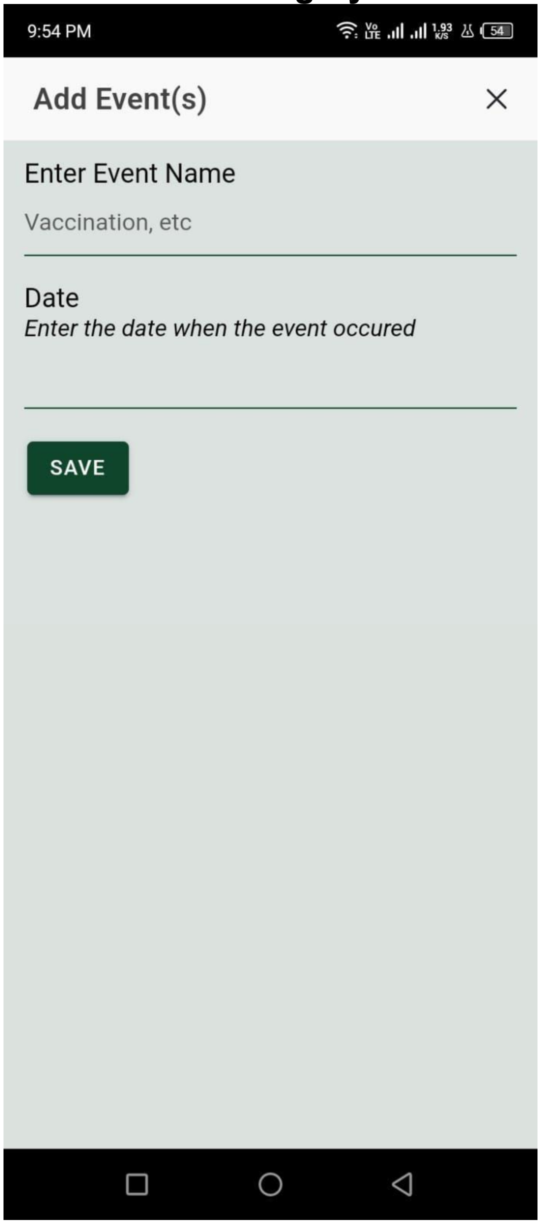
Key Events

- Vaccinations, etc 22/21/2024
  Short Detail
- Vaccinations, etc 22/21/2024
  Short Detail

+

Precautions
Updates precautions on every health checkup

7.5.11 < Vaccination/Surgery Event - Mock Screen 11 >



Add Event(s)

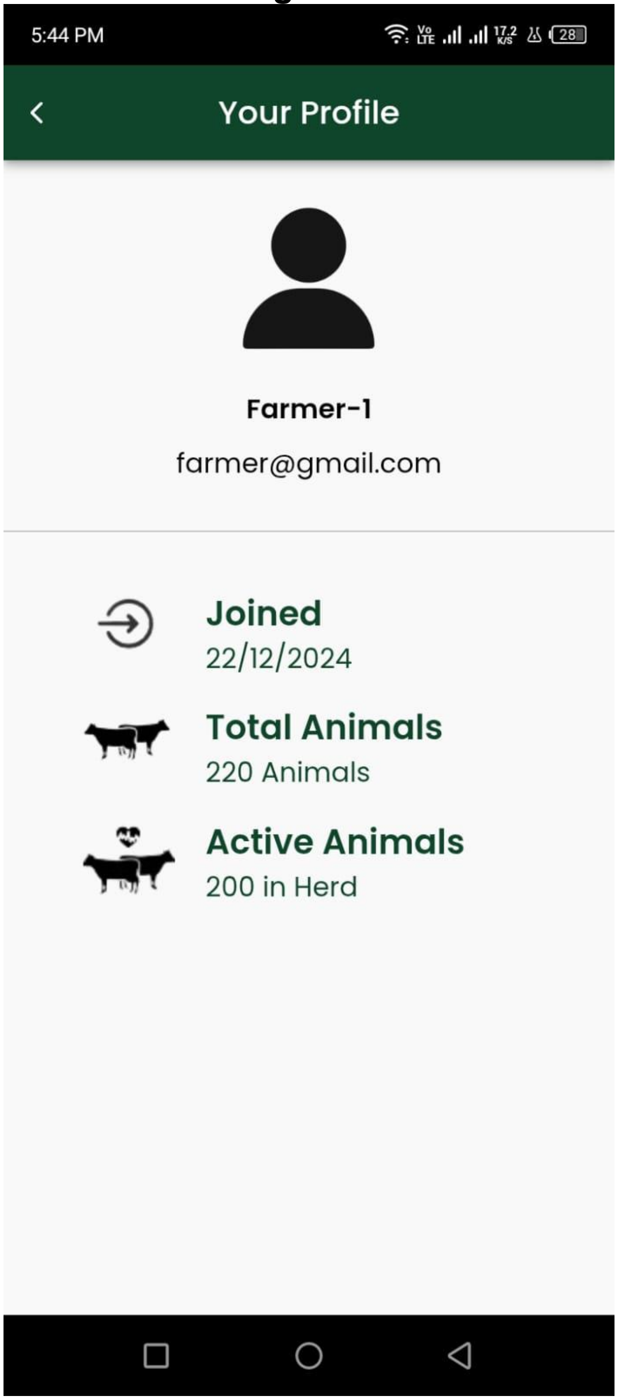
Enter Event Name

Vaccination, etc

Date

Enter the date when the event occured

7.5.12 < Settings - Mock Screen 12 >



5:44 PM

Your Profile

Farmer-1

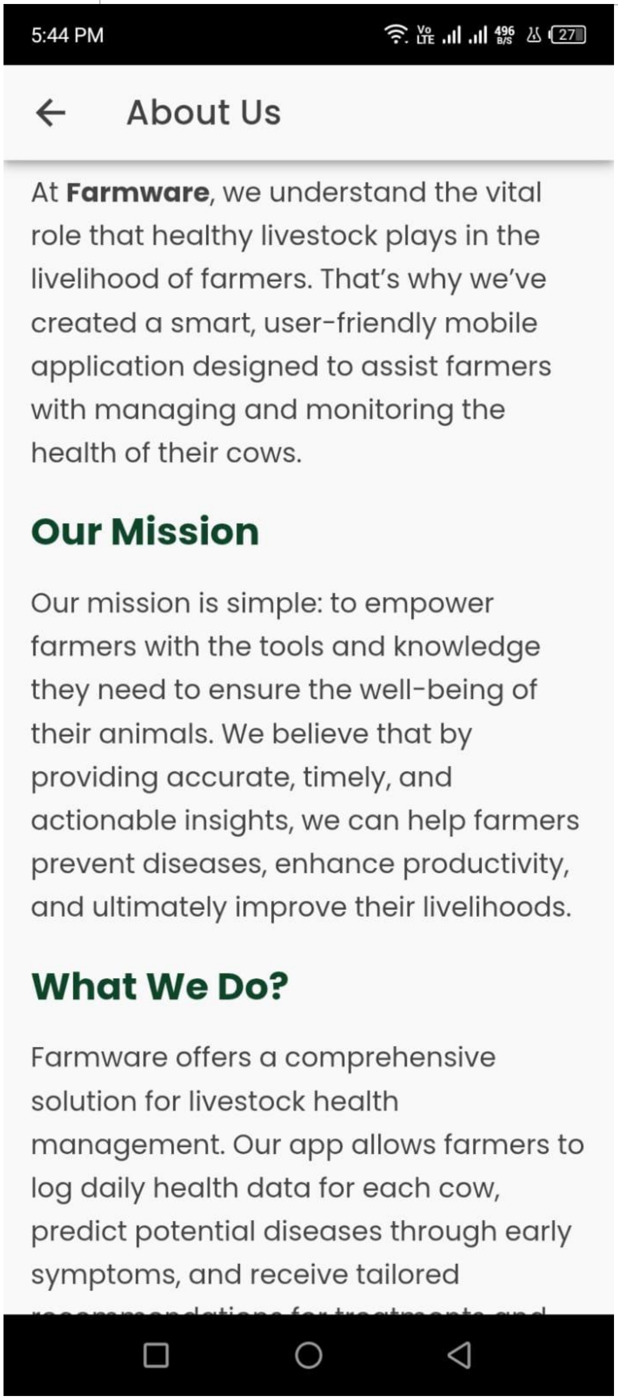
farmer@gmail.com

Joined
22/12/2024

Total Animals
220 Animals

Active Animals
200 in Herd

7.5.13 < About - Mock Screen 13 >



About Us

At **Farmware**, we understand the vital role that healthy livestock plays in the livelihood of farmers. That's why we've created a smart, user-friendly mobile application designed to assist farmers with managing and monitoring the health of their cows.

Our Mission

Our mission is simple: to empower farmers with the tools and knowledge they need to ensure the well-being of their animals. We believe that by providing accurate, timely, and actionable insights, we can help farmers prevent diseases, enhance productivity, and ultimately improve their livelihoods.

What We Do?

Farmware offers a comprehensive solution for livestock health management. Our app allows farmers to log daily health data for each cow, predict potential diseases through early symptoms, and receive tailored

8 References

Ionic Official Website: https://www.cambridgeinternational.org/
UI Reference: https://www.bivatec.com/apps/my-cattle-manager
Design Guidance: https://denovers.com/blog/what-is-modular-design/

9 Appendices

9.1.1 Appendix B: Sample Use Case

Risk	Impact	Mitigation Strategy
Internet connectivity issues	Users may not be able to submit data	Provide offline saving and sync when connected.
Inaccurate disease prediction	Can lead to wrong recommendations	Collaborate with veterinarians for better training data.
User feedback changes UI flow	May disrupt the app experience	Design a modular UI for easy updates.

9.1.2 Appendix B: Sample Use Case

Use Case Name: Daily Health Monitoring

Description: A farmer logs into the app, selects a cow from the list, and completes the daily symptom selection to monitor the cow's health. Based on the responses, the app predicts potential health risks and provides recommendations.

Actors: Farmer (User)

Preconditions:

- The farmer must be logged in.
- The cow’s profile must be added to the system.

Steps:

- The farmer selects a cow from the list.
- The Symptom Toggling Page opens with a list of common symptoms.
- The farmer toggles symptoms they observe in the cow (on/off).
- The farmer submits the selected symptoms.
- The system analyzes the symptoms and provides possible disease predictions and care recommendations.

Postconditions:

- The health record is saved in the database.
- Predictions and recommendations are displayed to the farmer.