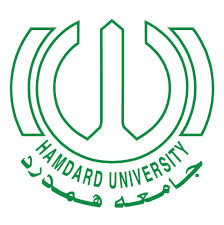
Hamdard University

Department of Computing

Final Year Project



**Smart Cattle Monitoring System with IoT and AI/ML**

**(FYP-033/FL24)**

**Software Design Specifications**

**Submitted by**

Zunair Shahid (2422-2021)

Hina Roshan (1389-2019)

Supervisor(s)

Sir Iqbal Uddin Khan

**Fall 2024**

**Document Sign off Sheet**

### Document Information

|  |  |
| --- | --- |
| **Project Title** | Smart Cattle Health Monitoring with IoT and AI/ML |
| **Project Code** | FYP-033/FL24 |
| **Document Name** | Software Design Specifications |
| **Document Version** | 1.0 |
| **Document Identifier** | FYP-033/FL24-SDS |
| **Document Status** | Final |
| **Author(s)** | Zunair Shahid, Hina Roshan |
| **Approver(s)** | Sir Iqbal Uddin Khan |
| **Issue Date** | 6-Jan-205 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Role** | **Signature** | **Date** |
| Zunair Shahid | Team Lead |  | 16-Jan-2025 |
| Hina Roshan | Team Member 2 |  | 16-Jan-2025 |
| Iqbal Uddin Khan | Supervisor |  | 16-Jan-2025 |
|  | Co-Supervisor |  |  |
| Sir Suleman Ahmed Naz | Project Coordinator |  |  |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 6-Jan-2025 | 1.0 |  | Hina, zunair |
| 3-July-2025 | 2.0 |  | Hina, zunair |
|  |  |  |  |
|  |  |  |  |

**Definition of Terms, Acronyms, and Abbreviations**

|  |  |
| --- | --- |
| **Term** | **Description** |
| IoT | Internet of Things |
| AI | Artificial Intelligence |
| ML | Machine Learning |
| GPS | Global Positioning System |
|  |  |
|  |  |
|  |  |
|  |  |

[1.1.1 Document Information 2](#_Toc188011068)

[2 Table of Contents 4](#_Toc188011069)

[1. Introduction 4](#_Toc188011070)

[1.1Purpose of Document 4](#_Toc188011071)

[1.2Intended Audience 5](#_Toc188011072)

[1.3Document Convention 5](#_Toc188011074)

[1.4Project Overview 6](#_Toc188011075)

[1.5Scope 6](#_Toc188011076)

[2. Design Considerations 7](#_Toc188011077)

[2.1Assumptions and Dependencies 7](#_Toc188011078)

[2.2Risks and Volatile Areas 7](#_Toc188011079)

3.  [System Architecture 7](#_Toc188011080)

3.1 System Level Architecture 7

3.2 Software Architecture 8

[5 Detailed System Design 8](#_Toc188011081)

[5.1Design Class Diagram 8](#_Toc188011082)

[5.2Database Design 8](#_Toc188011083)

[5.2.1 ER Diagram 9](#_Toc188011084)

[5.3Application Design 12](#_Toc188011085)

[5.3.1 Sequence Diagram: 12](#_Toc188011086)

[5.3.2 State Diagram 13](#_Toc188011087)

[5.4GUI Design(Tentative) 15](#_Toc188011088)

6 [References 16](#_Toc188011089)

[7 Appendices 17](#_Toc188011090)

# Introduction

## Purpose of Document

This document outlines the design and architecture of the Smart Cattle Health Monitoring System, including system components, interfaces, and data flow, to ensure a clear and comprehensive understanding for development and implementation.

## Intended Audience

## The primary audience for the Smart Cattle Monitoring System includes:

1. **Livestock Farmers**:

* Farmers seeking efficient, real-time monitoring of their cattle’s health and behavior.
* Those aiming to reduce operational costs and improve farm productivity through data-driven decisions.

1. **Dairy Farm Owners**:

* Owners focused on maximizing milk production by ensuring the health and fertility of their cattle.

1. **Agricultural and Livestock Industry Stakeholders**:

* Industry professionals interested in integrating technology to improve livestock management.

1. **Veterinarians**:

* Professionals requiring accurate, real-time data for diagnosis and treatment of livestock health issues.

1. **Government and Non-Governmental Organizations (NGOs)**:

* Institutions promoting technological advancements in agriculture for rural development and economic uplift.

1. **Tech Enthusiasts and Researchers**:

* Individuals or organizations interested in exploring the use of IoT and AI in the agricultural sector.

The system is particularly designed to benefit farmers and livestock owners in Pakistan, considering the challenges faced by this demographic in traditional cattle monitoring practices.

## Document Convention

This document uses Arial 12pt font size for the body and Arial 14pt bold for the headings with the line spacing on 1.

## Project Overview

The Smart Cattle Monitoring System leverages IoT and AI/ML technologies to revolutionize livestock management. The system integrates IoT devices like sensors, GPS modules, and microcontrollers to collect real-time data on cattle health, movements, and activities. Using AI algorithms, this data is analyzed for anomalies, providing actionable insights to farmers via a bilingual mobile application. The design approach employs a Waterfall methodology, ensuring a structured and sequential development process with clear deliverables and documentation.

## Scope

**In Scope**:

* Hardware integration using IoT sensors for body temperature, heart rate, and activity monitoring.
* Development of a bilingual mobile application (English and Urdu) for real-time updates and alerts.
* AI-based analysis for health trend monitoring.
* Cloud-based data storage and processing for accessibility and scalability.
* Initial testing on a small-scale farm setup.

**Out of Scope**:

* Large-scale field testing and extended performance evaluations.
* Custom hardware development beyond commercially available components.
* Long-term support, maintenance, or updates post-deployment.
* Advanced scalability and redundancy solutions.

# Design Considerations

## Assumptions and Dependencies

* Stable internet connectivity will be available at the deployment sites.
* Farmers using the system will possess basic technical literacy.
* Commercially available IoT sensors and microcontrollers will meet the project requirements.
* The allocated budget will cover the required hardware and software tools.

## Risks and Volatile Areas

* **Hardware Failures:** Components like sensors or microcontrollers may malfunction, mitigated by sourcing reliable parts and maintaining spares.
* **Budget Overruns:** Unforeseen expenses could exceed the budget; prioritization of essential features and cost-effective solutions is essential.
* **AI/ML Complexity:** Challenges in developing algorithms can delay progress; incremental development and using proven frameworks reduce risks.
* **Technological Changes:** Advancements in IoT and AI could render some components obsolete; modular design ensures easier upgrades.

1. **System Architecture**

## System Level Architecture

The system architecture comprises three primary elements:

* **IoT Device Layer:** Sensors, GPS modules, and microcontrollers collect and transmit data.
* **Cloud Processing Layer:** Cloud services process, store, and analyze data using AI algorithms.
* **User Interaction Layer:** A bilingual mobile app provides real-time monitoring and notifications.

Relationships: IoT devices send data to the cloud via Wi-Fi; processed insights are delivered to the mobile app. External interfaces include APIs for third-party cloud services. Error handling involves redundancy in data transmission and robust algorithms for fault detection.

* 1. **Software Architecture**

The software architecture is divided into three layers:

* **User Interface Layer:** Displays real-time data, notifications, and reports in a bilingual app.
* **Middle Tier:** Implements business logic and AI/ML algorithms for data processing.
* **Data Access Layer:** Manages cloud-based data storage and retrieval.

1. **Design Strategy**

The design emphasizes modularity, scalability, and user-centric functionality:

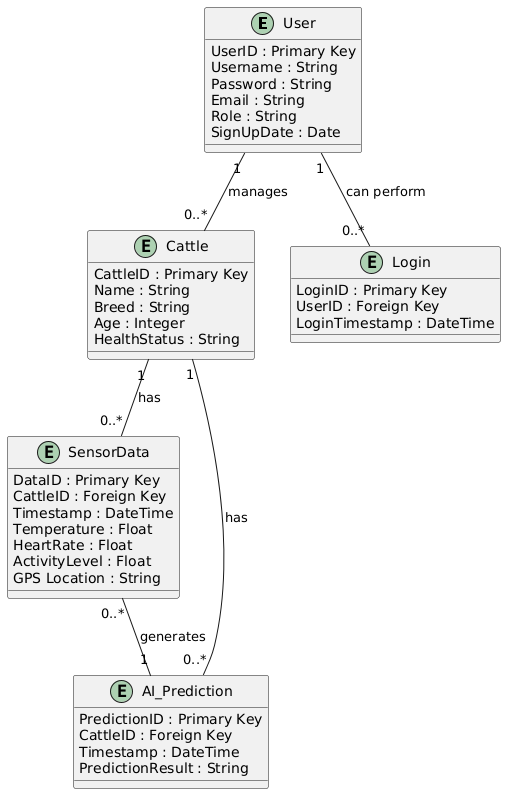
* **Future Extension:** The modular architecture allows integration of new sensors or AI models.
* **System Reuse:** Standardized components and frameworks ensure replicability for other livestock types.
* **User Interface Paradigm:** A mobile-first design ensures accessibility and ease of use for farmers.
* **Data Management:** Cloud storage ensures secure and scalable data handling, while data persistence supports historical analysis.
* **Concurrency:** Real-time data processing and synchronization are achieved using efficient protocols.

Trade-offs include balancing cost constraints with advanced features and prioritizing essential functionalities for initial deployment. The system’s design aligns with its goals of improving livestock management and enhancing productivity.

# Detailed System Design

## Database Design

### ER Diagram



**Data Dictionary**

**User:**

* UserID: Primary Key, Integer, Unique identifier for each user.
* Username: String, The username chosen by the user.
* Password: String, The password for the user account.
* Email: String, The email address of the user.
* Role: String, The role of the user (e.g., Admin, User).
* SignUpDate: Date, The date the user signed up.

**Cattle:**

* CattleID: Primary Key, Integer, Unique identifier for each cattle.
* Name: String, The name of the cattle.
* Breed: String, The breed of the cattle.
* Age: Integer, The age of the cattle.
* HealthStatus: String, The health status of the cattle.

**Login:**

* LoginID: Primary Key, Integer, Unique identifier for each login session.
* UserID: Foreign Key, Integer, References User(UserID).
* LoginTimestamp: DateTime, The timestamp of the login session.

**SensorData:**

* DataID: Primary Key, Integer, Unique identifier for each sensor data entry.
* CattleID: Foreign Key, Integer, References Cattle(CattleID).
* Timestamp: DateTime, The timestamp of the sensor data.
* Temperature: Float, The temperature reading from the sensor.
* HeartRate: Float, The heart rate reading from the sensor.
* ActivityLevel: Float, The activity level reading from the sensor.
* GPSLocation: String, The GPS location reading from the sensor.

**AI\_Prediction**:

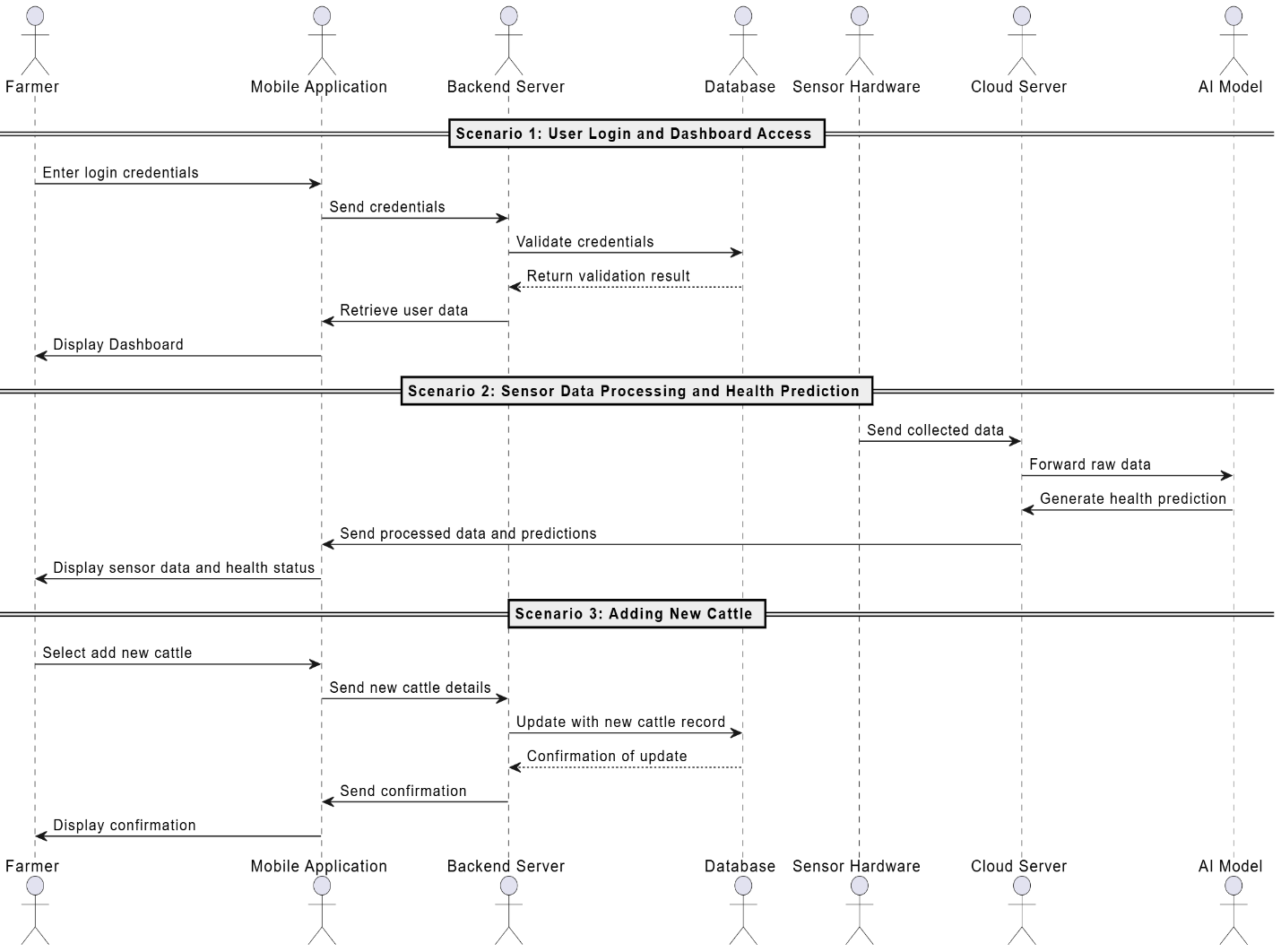
* PredictionID: Primary Key, Integer, Unique identifier for each AI prediction.
* CattleID: Foreign Key, Integer, References Cattle(CattleID).
* Timestamp: DateTime, The timestamp of the AI prediction.
* PredictionResult: String, The result of the AI prediction.
* ConfidenceLevel: Float, The confidence level of the prediction.

**Relationships:**

* User can perform multiple Logins.
* User can manage multiple Cattle.
* Cattle can have multiple entries in SensorData.
* SensorData can be used to generate multiple AI\_Prediction.
* Cattle can have multiple AI\_Prediction results over time.

## Application Design

### Sequence Diagram:



**Explanation of the Sequence Diagram:**

**Scenario 1: User Login and Dashboard Access**

* The farmer logs in via the mobile app.
* Credentials are sent to the backend server for validation through the database.
* Upon successful validation, the user data is retrieved, and the dashboard is displayed to the farmer.

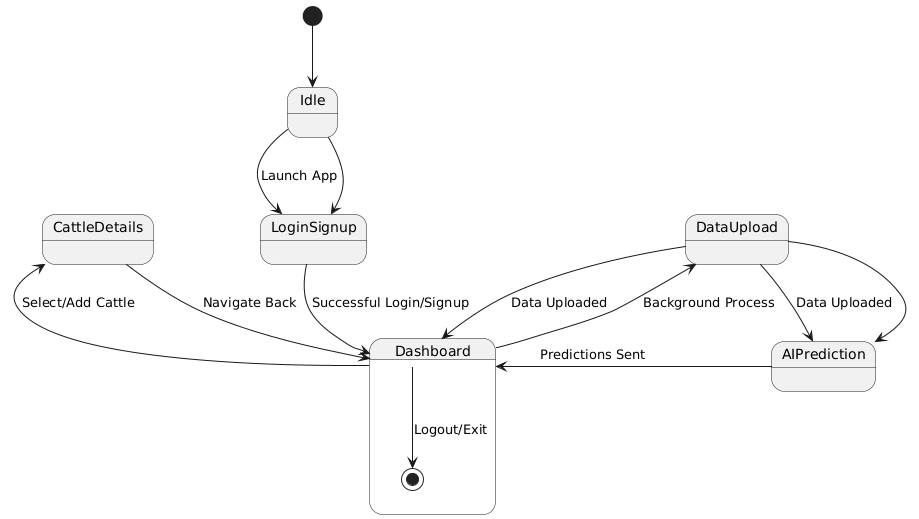
**Scenario 2: Sensor Data Processing and Health Prediction**

* Sensor hardware collects data from cattle and sends it to the cloud server.
* The cloud server forwards raw data to the AI model for health prediction.
* The AI model processes the data and returns predictions.
* The processed data and predictions are displayed to the farmer through the mobile app.

**Scenario 3: Adding New Cattle**

* The farmer selects the option to add new cattle through the mobile app.
* New cattle details are sent to the backend server, which updates the database.
* A confirmation of the update is sent back to the farmer and displayed in the app.

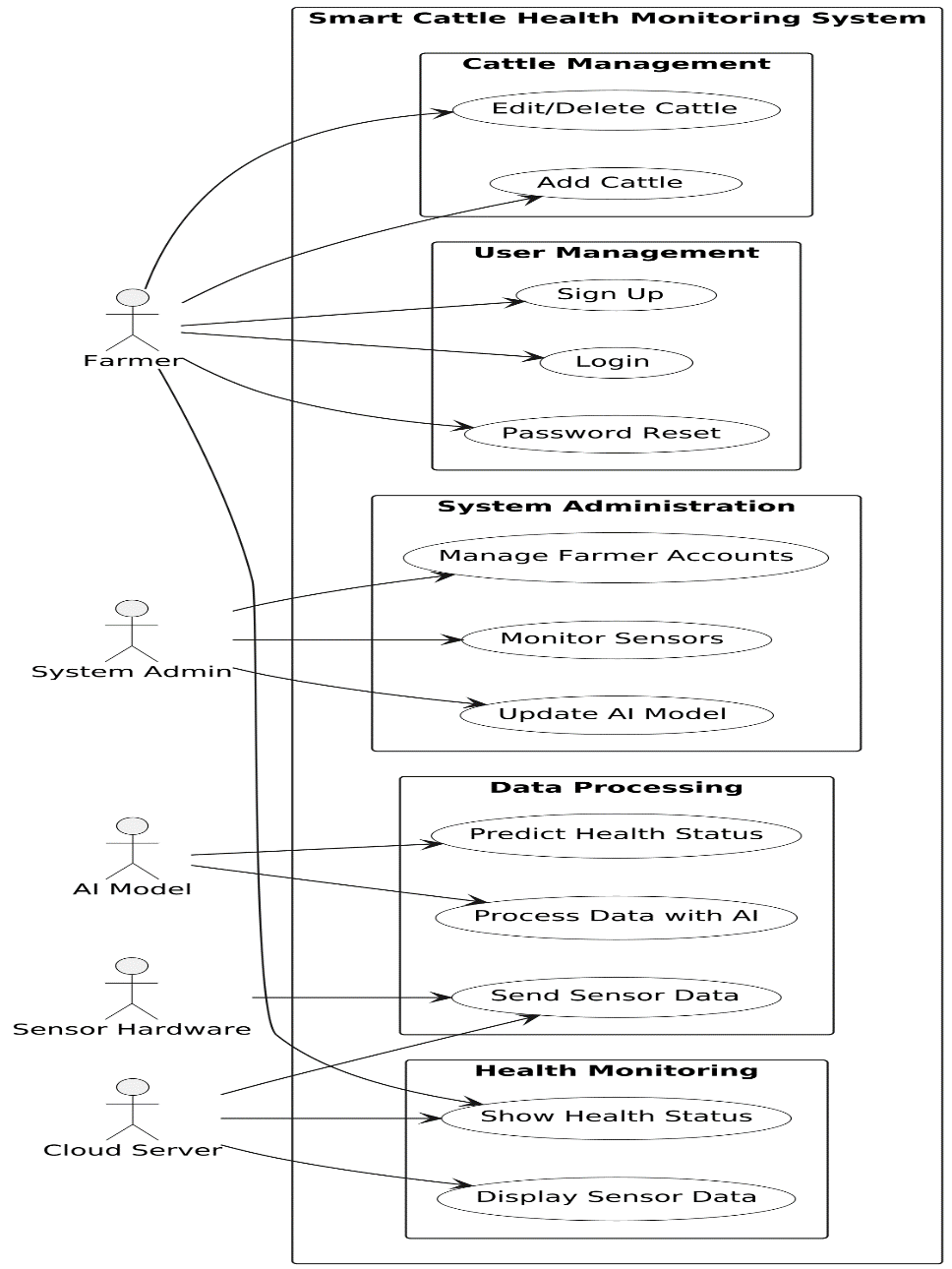
### State Diagram



## 

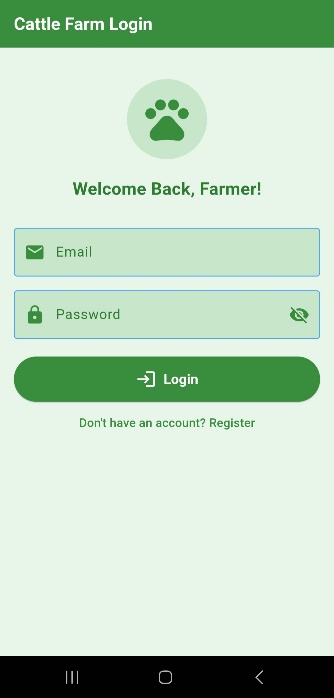
**Explanation of the State Diagram:**

* Idle: The starting state where the app launches.
* LoginSignup: The state for user login or signup. Successful login/signup transitions to the Dashboard.
* Dashboard: The main screen where users can:
  + View cattle information and AI predictions.
  + Navigate to CattleDetails to select or add cattle.
  + DataUpload: Upload cattle data, then return to the Dashboard.
  + Logout/Exit to return to Idle.
* CattleDetails: Users can navigate back to the Dashboard.
* DataUpload: Transitions to AIPrediction upon data upload and sends predictions back to the Dashboard.
  + 1. **Use Case Diagram:**

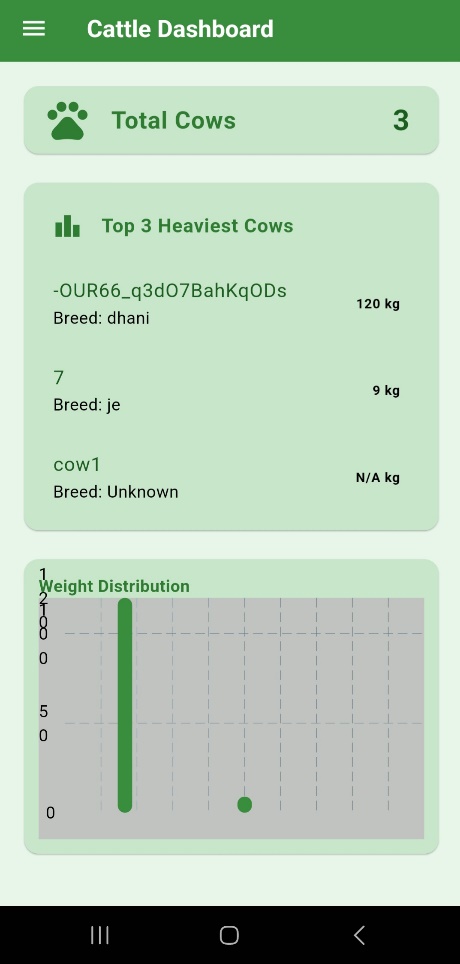
****

## 5.3 GUI Design

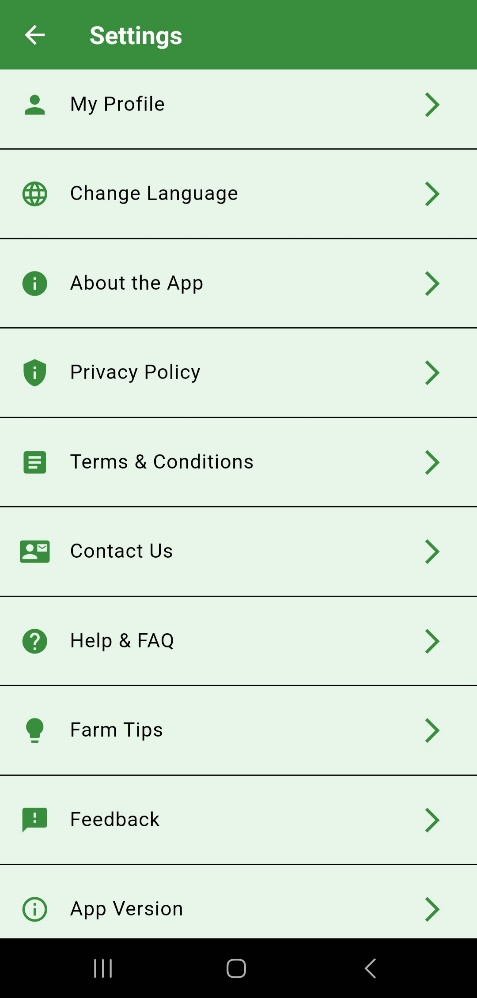
**Login Page:**



**Dashboard:**

** **

**Other Screens:**

# References

# Appendices