TITLE: THE DAIRY FARMING MANAGEMENT

FORM: MOBILE APPLICATION

BY: KAIRU ANGELA WANJIRU

ADMISSION NUMBER: 656995

UNITED STATES INTERNATIONAL UNIVERSITY-AFRICA

STATEMENT:

A Mid-Semester Project submitted to the School of Science and Technology in Partial Fulfillment of the requirements for the degree of Bachelor of Science in Applied Computer Technology

THIRD YEAR, SECOND SEMESTER
(FALL 2020)

STUDENT’S DECLARATION AND APPROVAL

I, the undersigned, declare that this is my original work and has not been submitted to any other college, institution or university other than the United States International University in Nairobi for academic credit.

Signed:...**KAIRU ANGELA WANJIR**U…..**656995**…. Date:…...**16TH DECEMBER, 2020**……

This project has been presented for examination with my approval as the appointed supervisor.

Signed:…………………………………………………Date:…………………………………

COPYRIGHT

ABSTRACT

This project is pursued as an effort to solve a persistent issue within the agricultural industry of Kenya specifically under dairy farming. Local dairy farmers have expressed their grievances and are looking for a solution I would like to provide. This therefore provides a market for my proposed solution. The main issue revolves around record keeping as well as tracking and this project ambitiously works to implement mobile application programming tools alongside the waterfall software methodology of development to create a dairy farming management application to mitigate the challenges of local dairy farmers.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Terms of Reference

DFMA - Dairy Farming Management Application

GUI - Graphical User Interface (The display)

DB - Database

IDE - Integrated Development Environment

SMS - Short Message Service

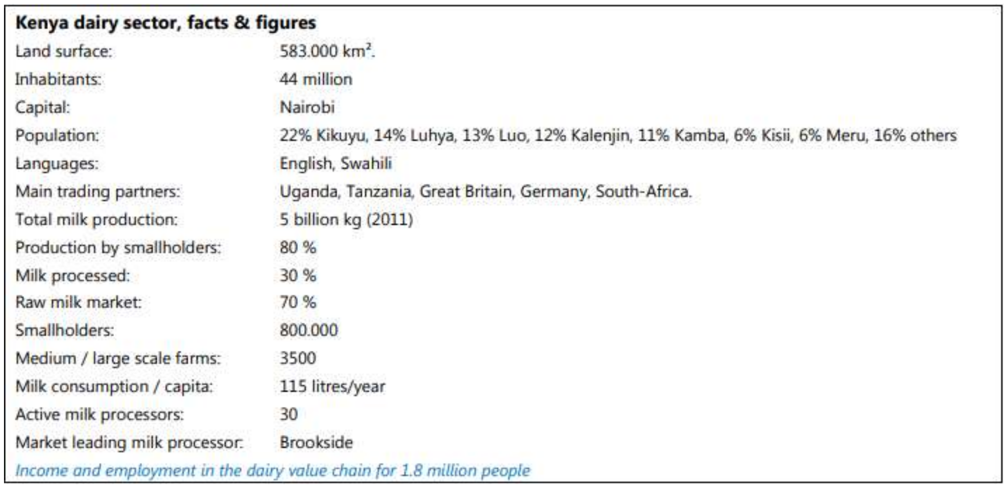
WiFi - Wireless Fidelity

CHAPTER 1: INTRODUCTION

1.0 Background of the System/Project Area

The clientele for this project are the local dairy farmers and in description, Dairy farming is the keeping of cattle for the sake of milk production. In Kenya, the dairy cattle kept are mostly goats and cows of Friesian, Ayrshire, Jersey and Guernsey breed. In a cultural perspective, Kenyans offer guests milk tea and many communities regard having many cows, goats and other cattle a sign of wealth explaining their high value during dowry payment. Dairy farming is also valued by the agricultural sector. In Africa, Kenya comes second to South Africa as a country that can produce enough milk for both domestic consumption and export with the milk processed accounting for 14% of the Agricultural sector's Gross Domestic Product. More relevant statistics are featured in the breakdown displayed in the image below.

Fig 1



Kenya dairy sector, facts & figures

Land surface:	583.000 km^2.
Inhabitants:	44 million
Capital:	Nairobi
Population:	22% Kikuyu, 14% Luhya, 13% Luo, 12% Kalenjin, 11% Kamba, 6% Kisii, 6% Meru, 16% others
Languages:	English, Swahili
Main trading partners:	Uganda, Tanzania, Great Britain, Germany, South-Africa.
Total milk production:	5 billion kg (2011)
Production by smallholders:	80 %
Milk processed:	30 %
Raw milk market:	70 %
Smallholders:	800.000
Medium / large scale farms:	3500
Milk consumption / capita:	115 litres/year
Active milk processors:	30
Market leading milk processor:	Brookside

Income and employment in the dairy value chain for 1.8 million people

In the agricultural sector, record keeping is important for tracking of relevant activities and key aspects. Dairy farmers keep records such as ancestry, breeding, heat period dates, pregnancy & health status, production, vaccination and performance. This allows for accountability and ease of tracking that can help with decision making and planning. Unfortunately, many local farmers keep these records in farm books or cards. In many farms, land owners hire workers who are the ones to keep those records by writing information into books then the owner references the book to understand the status and trends. Few tech-savvy owners try to then manage their records on spreadsheets.

1.1 Problem Statement

The Standard newspaper recently reported that the biggest challenge local dairy farmers are facing is the lack of technology to support them. They struggle keeping records and tracking their cattle with boards or books and they have to keep manually erasing and rewriting as things change. They also have to keep remembering important dates on their own and these are all cumbersome and error prone methods. The larger number of local dairy farmers are in the rural area and many of them struggle with illiteracy. Most speak and understand either their mother tongue or the national Language; Kiswahili. Therefore, it is not unexpected that they would be computer literate.

Most of the best software available for dairy farmers are web apps or computer software tailored for farmers outside this region. However, there is a company ; Farmingtech Solution Limited, that has made an effort to give a solution for the Kenyan dairy farmers' challenges with the launch of their well-liked app Digicow. Unfortunately, many new users struggle using the app because they are stuck in the log in section and thus they often give up without using it. More details on this app are covered in the Literature Review section of the paper. In an effort to solve these problems, I have thus created an app that eases these tasks and that is tailored for farmers in Kenya to mediate the record keeping problems and address the issue on the lack of systems to aid them well.

1.2 Proposed Solution

To resolve the highlighted issues, I have proposed a solution in form of a mobile application meant to aid local dairy farmers with record keeping and also with reminders of important details such as vaccination, delivery and insemination dates. It has no log in requirements and an interface that allows the user to input new records in regards to the cattle and those records are used to notify the user of events.

1.3 Project Objectives

1.3.1 General Objective

To build a dairy farming management mobile application meant to aid local farmers with record keeping and also with keeping track of important events.

1.3.2 Specific Objectives

1. To take in and store new records of cows from user input
2. To notify users of important events
3. To enable a user to see the records stored
4. To allow users to delete records they no longer need
5. To allow easy navigation on the app

1.4 Project Assumptions

In regards to this project, I made several assumptions including the assumption that providing a solution to a previously expressed challenge will not be in vain as there is an existing market for my proposed solution. Also,. I assumed that with the great smartphone penetration in Kenya, attributed to the increasing affordability of the phones, the project is viable and the app would be used by dairy farmers to perform the simple record keeping tasks on their farms with ease assuming that that is all they would need it for.

1.5 Scope and Limitation

This project focuses on the local dairy farming part of the agricultural sector with local dairy farmers as the clientele. The time and skill constraints do not allow the development of a heavily developed application and thus, it does not include other animals other than cows at least for now. It however takes in and stores new cow records, sets event reminders, displays stored records and allows deletion of records. Users are not required to log in. A well designed user interface is set up with easy navigation in a way that is clear so users can easily use it and also to ensure users understand what is required of them.

The project will involve research, my development skills, lecturer input, other forms of feedback, objectives, software and hardware platforms for development and testing and a self-constructed schedule to ensure I complete it by the end of the semester. The deliverables of the project will be a working dairy farming management application with objectives having been met.

CHAPTER 2: LITERATURE REVIEW

State of the Industry, Existing solutions and Shortcomings

2.1 Introduction

In the dairy farming industry, technology-based solutions have been developed to assist dairy farmers in their endeavors. They come in the form of software as mobile applications, web applications and computer management software or as hardware with milking machines and others that help improve production and feeding. Dairy farmers in Africa are different from those in America as they experience different challenges and have different needs. They therefore have different solutions for each of their problems. Some of their solutions can be applied locally but they are not always applicable. Western countries and some other regions outside Africa have advanced technology-based solutions with the influence and application of robotics, AI, monitoring systems and drones while in Kenya not much can be said about the same.

Those regions have many dairy farming management systems tailored for them to solve their problems and they have a variety to pick from. On Android phones, the source for apps is the Google Play Store where many developers publish their software and on iOS phones, it is the App Store. On that note, dairy farming management apps developed in other regions and made available on these platforms can be downloaded by local dairy farmers. In Kenya, there is no variety and there are few fully viable choices. The system that has gained acceptance and traction locally is DigiCow, an app tailored to support dairy farmers in Kenya. However, many local farmers are not aware of existing systems and continue to experience challenges while solutions exist. Also many current apps are generalized for farm management and not dairy specifically.

Some of the applications also have complex navigation interfaces without tutorials to help those unfamiliar with such systems. Simplicity would be key locally as smartphone penetration is still growing gradually in the rural area where many local dairy farmers are. Additionally, many local farmers are quite old and would embrace simple systems. With the growing aid of technology in dairy farming, applications have been and are still being developed to fix existing and emerging issues. Apps in the market are well appreciated but with the challenges expressed by local dairy farmers, more has to be done for their needs to be fully satisfied and their expectations to be met. There are various examples and a few of these are highlighted below.

2.2 Existing Dairy Farming Management Mobile Applications

2.2.1 DigiCow

Description, Features and Merits

DigiCow is a cow management mobile application tailored to solve the challenges Kenyan dairy farmers face. This application was launched by Farmingtech Solutions Limited, a Kenyan company started in the year 2014. They have expert knowledge in, communication, livestock production, finance, IT and other fields. With DigiCow, the targeted clientele are small scale dairy farmers in Kenya. They aim to deliver with their app the ease of dairy farmers' management and to improve their data-driven decision making process to increase their profits.

The app is available for access and installation from Google PlayStore. Registration is a requirement with a PIN needed to secure data. The users then can key in information in relevance to their herd of cattle and details captured are; milk production where they can record the amount of milk produced by individual cows on different days, they can also have records of feeding information, milk sales, health and breeding.

As a bonus, the data recorded is used by the application to give feedback in the form of reports. The reports are in relevance to finances, production, performance, breeding and health. A farmer can get alerts about the same information and take note of important events and some failed attempts of things like conception and improved milk production. The app also has a loan facility to offer farmers credit and a chart room to assist the farmers with expenses and interactions with experts or other farmers. It is well liked with many good reviews and a 4.6 rating on PlayStore.

Demerits

Unfortunately, many new users struggle using the app because they are stuck in the log in section and thus they often give up without using it. Ngura (2020) said this in the PlayStore review section, "unable to register only to say authentication error, plz rectify." He is one of many who have struggled with the issue. Also, it asks for users to enable location permissions and some individuals are not comfortable with that.

2.2.2 Cow Master

Description, Features and Merits

Cow Master is a herd management software application with many positive reviews and a 4.6 rating on PlayStore. It is considered a very low cost application of its nature and is thus well appreciated by its users. It was developed and launched to help farmers manage all herd dynamics and keep track of their cattle. The app has the capacity to keep all records of these animals throughout their different life phases.

It also has a good notification system for each of those individual breeding phases. Farmers can keep track of aspects like milk production and budgeting. The app has various modules including one that covers Incomes and Expenses with reports to aid in decision making and this information can be shared with other relevant parties. It does not require the internet at any time other than when downloading it from PlayStore where it is hosted and data is also stored locally.

Demerits

The negative side would be that with all the modularity involved, it is easy for some users to get confused especially on the first trials and this could make them give up. Fahim (2020) asked, “How does it work?” in the PlayStore review/comment section and KelimeSoftAugust responded, “This is one of the simplest app in the store. Though, we have added a contact menu for whom does not understand the app. After a little effort you will be able to find that, go on”. This shows that some people do struggle understanding their way around and the feedback is not the most polite.

2.2.3 Cattle Manager

Description, Features and Merits

As part of the Apps for Good programme, an education movement for application development, this cattle management application was developed by Wick High School students; John, Keiran and Ryanand in Scotland and it won the programme's Awards of 2013 in the "Power to do More - getting the most from your time " category. The development team worked with Novada, a development agency to professionally build on their prototype and have it launched publicly.

The app is meant to work as a tool of convenience for farmers who would like to keep tabs on dynamics of their cattle allowing them to store and read information on each animal with information such as tag numbers and breeds being recorded. It has **made a** relatively good number of users happy with an average rating of 3.6 on PlayStore.

Demerits

The application has functionality problems with users saying "The text overlaps when entering cattle... uninstalled the app due to this" Johnsen (2020) and "wont even work. as soon as I try to enter vaccinations it closes" according to Dill (2019).

2.2.4 My Solution

In addition to delivering a solution to local dairy farmers, I would like to implement solutions that correct flaws in existing applications such as those featured above. For instance; I do not intend on asking users to allow for <u>location permissions</u> as I understand how privacy is a sensitive issue and for now, I will not include any <u>registration or log in requirements</u> to allow users to freely access the app services in a fast and easy fashion.

With my application, I will have a <u>simple and straight-forward user interface</u> to ensure users can <u>navigate easily</u> and I also intend to give polite and speedy feedback to users when the time comes for me to provide technical support or to receive reports from them. I will also make sure the <u>app functions correctly and all implemented features compliment each other</u>.

CHAPTER 3 : DEVELOPMENT METHODOLOGY

3.1 General Approach

With the decision to go into the development of this application to help dairy farmers, extensive research was done to establish the best course of action that has been outlined to guide the development of this dairy farm management app. My general approach was to start by gathering knowledge on the local dairy farming field to understand what it is about, to gather relevant information and to also gather hardware and software requirements.

With the actual system development, I chose to follow a systematic approach whereby I accomplished each activity and objective in progessive stages from beginning to end. That is further described below

3.2 Software development methodology

3.2.1 Methodology

Approach of choice - Waterfall Methodology

3.2.2 Description

This is a software development approach that was vastly implemented as the first model in the early age of software engineering. The name given to it is as a description of the flow of activities that are undertaken during development. The main feature is; the whole development process is divided into main stages or phases and the output of a phase becomes the input of the next phase.

3.2.3 Application

At different phases of the process, I did different activities to achieve the different goals of each stage. This involved;

1. <u>Requirement Gathering and analysis</u> - I did an analysis of my project situation, identified all the background information and identified hardware components and software applications that I required for the development of this system.

Output: A list of the requirements - in this case; a Laptop, Android Studio software, a Flash disk and a Mobile phone.

2. System Design - In addition to acquiring all my requirements, I also understood how I would have to integrate each element. With that understanding, I developed a design of my system's architecture (back and front ends) with the use of all the equipment/requirements I had. I had sketches and flowcharts where I defined my product and how it would appear with integrated features.

   Output: Sketches and design diagrams e.g:- Use case diagram, ERD diagram, Flowchart

3. Implementation - With the system design general plan and supporting diagrams, I brought the visual representations to life by developing each highlighted system feature and focusing on the main sub-parts of the system. While applying the design, I implemented modularity and testing of each system sub-part / module in a process known as Unit Testing.

   Output: Several tested modules representing functional parts of the whole system - Milk production feature, Medical feature e.t.c

4. Integration and Testing - With the developed and tested modules from the implementation stage, I formed the full system by bringing them all together so they all collaborated. After this integration, I tested to see if the modules were well implemented together without failure and that they performed the intended unified function.

   Output: A fully functional system - The Dairy Farming Management App tested on the mobile phone I obtained as a requirement.

5. Deployment of the system - With the completion of this project and with a functional application at hand, I can exchange it with a small lot of farmers or I can publish it on Google PlayStore for it to reach an even wider market through all the appropriate procedures.

   Output: A publication of The Dairy Farming Management App

6. Maintenance - After deploying the system, I will continue to modify it with new features and implementations as I develop new ideas. Also as new needs and challenges arise I will need to revisit the system and make modifications. To avoid any major problems, I will

continuously look at my system to make sure it works in the best way and to ensure it is adaptable.

Output: System update launches

3.2.4 Justification

I chose to use the Waterfall methodology because of the many advantages that it offers. They include the following;

1. This methodology is systematic and allows for proper planning and scheduling of tasks instead of having a randomized procedure
2. The phases allow for modularity and focused thinking whereby I can focus on one feature first and make sure it works instead of doing everything all together
3. Testing is thorough and easy as errors can be tackled within specific functions and models before they are all integrated.
4. The stages are clearly defined from the beginning giving a well guided process from beginning to end.
5. Each phase has output implemented in the next stage thus showing a flow of activities that compliment each other until the final stage is reached

3.3 Functional and Non-functional requirements

3.3.1 Non-Functional

1. The user will add details of new cows
2. Users will select dates for events they want to note
3. The user will add daily records of milk produced
4. Users will access and review stored data
5. The user will delete unnecessary or wrong data
6. The user will easily navigate the application

3.3.2 Functional requirements

1. The system will store records of new cows

2. The system stores milk records every day with the right dates
3. The system will remind the user of important events
4. The system has a simple user interface with clear features
5. The system will store and retrieve stored data to show the user
6. The system gets rid of records selected by users for deletion

3.4 Design tools

3.4.1 Pen and Paper

It was a very easy way to quickly capture what I had in mind at particular moments when I was struck with ideas. I spent less time dragging shapes on a software to represent what I have in mind in the moment. If I got a new idea while far from my devices, I easily drafted the idea on a paper before translating it to the actual feature later.

3.4.2 A Flowchart

It was a neat and well-structured way to represent how modules on my application were to be integrated and how they should work with each other to deliver all the functionalities offered by the application. Also, in the soft-copy form, I saved, copied and back-up my design for future reference. I also transformed my pen and paper designs to a more sophisticated design by use of flowcharts.

3.5 Development tools

3.5.1. Database development tools

Firebase database is the external database used in the development of this program. It is a platform developed by Google for creating mobile and web applications and it provides detailed documentation as well as cross-platform SDKs.

I chose it for the convenience and effectiveness that it offers. Setting up the connection between the IDE and the database is easy and straightforward with a project being created on each side and a connection being initiated from the IDE's end.

3.5.2. Programming tools

a. Android Studio programming software

This is the Integrated Development Environment for Google's Android operating system as it is specifically used for android development and it will be used to develop the application. It has a good emulator, code templates, guides, a vast set of libraries and a good user interface. I used it because of these features and also because it is the software I am most familiar with when it comes to the development of android applications.

b. Java programming language

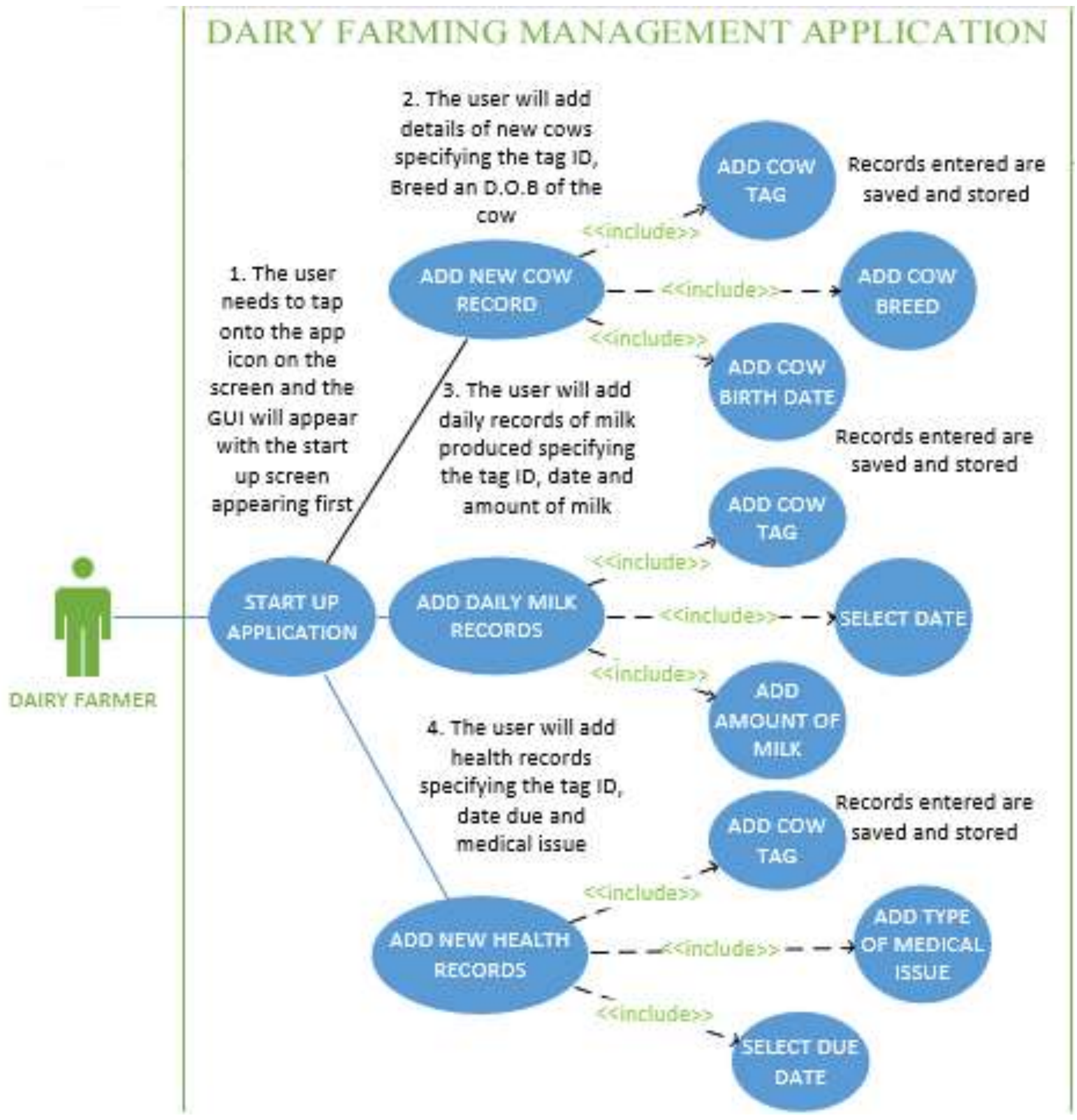
This is the language that I used to program the application as I used Android Studio for it supports programming in Java. I wrote code in this language to implement all the features and functions needed for the proper development and running of this app. I chose to use this language because I am familiar with it and the IDE I am using has it as an option.

CHAPTER 4: SYSTEM ANALYSIS AND DESIGN

UML USE CASE DIAGRAM -Screen design (GUI)

The diagram below is a physical representation of what the system's graphical user interface is supposed to look like and what main functions are to be performed.

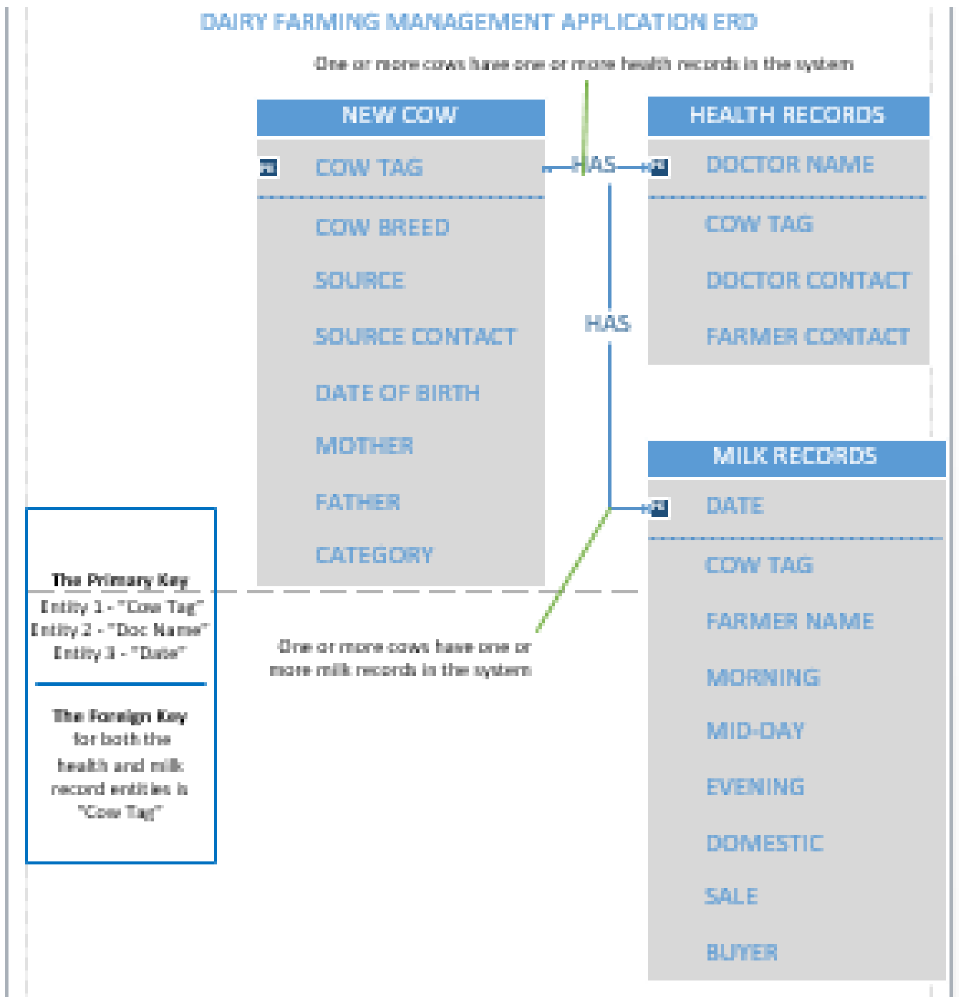
Fig 2


DAIRY FARMING MANAGEMENT APPLICATION
DAIRY FARMER
START UP APPLICATION
1. The user needs to tap onto the app icon on the screen and the GUI will appear with the start up screen appearing first
2. The user will add details of new cows specifying the tag ID, Breed an D.O.B of the cow
ADD NEW COW RECORD
<<include>>
ADD COW TAG
ADD COW BREED
ADD COW BIRTH DATE
Records entered are saved and stored
3. The user will add daily records of milk produced specifying the tag ID, date and amount of milk
ADD DAILY MILK RECORDS
ADD COW TAG
SELECT DATE
ADD AMOUNT OF MILK
Records entered are saved and stored
4. The user will add health records specifying the tag ID, date due and medical issue
ADD NEW HEALTH RECORDS
ADD COW TAG
ADD TYPE OF MEDICAL ISSUE
SELECT DUE DATE
Records entered are saved and stored

ERD DIAGRAM - Database Schema

The diagram below is a basic representation of the simple database schema that will operate in the back end of this application. The diagram shows key entities within the system as well as relationships, conditions and dependencies between them.

Fig 3



From the GUI, it is represented that the intention is to have a user start the application, move on from the welcome page, then select an activity which is either adding a new record or viewing existing ones. From there, they can either go on to add new cow, milk, health or reminder records if they pick add. If they pick view, they can proceed to have a look at existing records of individual cows, milk records, health records of the cows and reminders of important events that they set. All the data is saved to and retrieved from the firebase db as shown.

CHAPTER 5: CONCLUSION AND RECOMMENDATION

a. Conclusion

As a result of my development efforts and with the use of all the required tools and a well-structured schedule, I have managed to develop the DFMA successfully meeting all my objectives. I have developed all the required modules for adding and viewing; cow, milk, health and reminder records. The user interface is simple and the application is easy to navigate as it was aimed to be. The connection between the application and a database (Firebase) was successful allowing saving and retrieval of records just by pressing the required buttons that are well labelled. The app also has a functional reminder functionality with a popup notification implemented.

b. Recommendation

This application can be installed on mobile phones of farmers that need to ease the record keeping and management processes of their dairy farm. A farm with many cows can make use of this app because it provides advantages such as; convenience, ease of access, backup on an actual database, fast insertion and retrieval, clarity and order, preservation of records and the advantage of having a reminder. Farmers can introduce a naming and tagging system on their farms if they do not have one and each time they need to add a new record or to review existing records, they can keep track of each individual cow.

The system can be improved by adding a search functionality so users can view records of specific animals or of specific dates. It can also have an additional feature to send sms reminders to farmers and veterinarians. Local dairy farmers need to change how they are storing records because as most of them use books and files, they can lose these records, they have to manually add, cancel and delete details, they spend a lot of time adding and reviewing records because the records may not be in order, they do not have backups for these records and carrying books, papers or files is not as convenient as having your phone.

REFERENCES

Cattle Manager | Agriculture Apps | Farms.com. (2020). Retrieved 27 September 2020, from https://www.farms.com/agriculture-apps/livestock/cattle-manager

Cow Master - Herd Management App for Dairy Farms. (2020). Retrieved 29 September 2020, from https://play.google.com/store/apps/details?id=com.kelimesoft.suruyonetimi&hl=en&showAllReviews=true

Dairy Farming in Kenya - An Overview. (2014). Retrieved 25 September 2020, from https://africafarming.info/kenya-dairy-farming-an-overview/

Dairy Smartphone Apps compiled by Hoards Dairyman. (2012). Retrieved 27 September 2020, from https://hoards.com/article-6336-dairy-smartphone-apps-%e2%80%93-compiled-by-hoard%e2%80%99s-dairyman.html

DigiCow. (2020). Retrieved 28 September 2020, from https://play.google.com/store/apps/details?id=info.digicow.com&hl=en_US&showAllReviews=true

Elliott, R. (2018). Data Report on Farming in Kenya and Mobile Phone Usage - GeoPoll. Retrieved 26 September 2020, from https://www.geopoll.com/blog/data-farming-kenya-mobile-phone/#:~:text=In%20September%202018%2C%20GeoPoll%20conducted%20an%20in-depth%20survey,their%20perceptions%20of%20the%20latest%20trends%20in%20farming.

Ettema, F. *Dairy Development In Kenya* [Ebook] (pp. 1,5). Retrieved from https://www.dairyfarmer.net/fileadmin/user_upload/40_downloads/kenya-dairying-ETTEMA.pdf

Farm Source Dairy Diary. (2020). Retrieved 28 September 2020, from https://play.google.com/store/apps/details?id=nz.co.solnet solutions.mobile.fonterra.dqmsApp&hl=en_US

Gachuiri, C., Lukuyu, M., & Ahuya, C. (2012). *Dairy Farmers Training Manual* [Ebook] (1st ed., pp. 2-6). Nairobi: Ministry of Livestock Development. Retrieved from http://www.biyinzika.co.ug/wpcontent/uploads/2018/02/Dairy_Farmers_Training_Manual.pdf#:~:text=Kenya%20has%20over%20one%20million%20small%20scale%20dairy,of%204.2%20billion%20litres%20of%20milk%20every%20year.

Kamau, S. (2018). Dairy farm record keeping in Kenya. Retrieved 23 September 2020, from https://www.tuko.co.ke/278041-dairy-farm-record-keeping-kenya.html#:~:text=The%20Kenya%20Stud%20Book%20keeps%20ancestry%20records.%20Performance,maintained%20with%20the%20Dairy%20Recording%20Services%20of%20Kenya.

Kenya: The Digital Life of Kenya's Smallholder Farmers - Who's Using What Phones to Access Information and Loans. (2018). Retrieved 24 September 2020, from https://allafrica.com/stories/201812020151.html

How to keep proper dairy records using the DigiCow app. (2020). Retrieved 23 September 2020, from https://graduatefarmer.co.ke/2018/04/11/how-to-keep-proper-dairy-records-using-the-digicow-app/

Riley, J. (2020). 12 must-have phone apps for cattle farmers - Farmers Weekly. Retrieved 27 September 2020, from https://www.fwi.co.uk/machinery/technology/12-must-have-phone-apps-for-cattle-farmers

SDLC Methodologies | Top 6 Useful SDLC Models and Methodologies. (2020). Retrieved 7 October 2020, from https://www.educba.com/sdlc-methodologies/

Songok, F. (2020). DigiCow Adding Value to Dairy Cooperatives in Nandi – County Government Of Nandi. Retrieved 24 September 2020, from https://nandicounty.go.ke/news/digicow-adding-value-to-dairy-cooperatives-in-nandi/

APPENDIXES

A. Budget and Resources

For full development of this application and the overall success of this project, I have several requirements and resources to procure. To facilitate this effort, I have a budget set as below :

Table No.1

ITEM	QUANTITY	PRICE PER ITEM	TOTAL COST
Laptop	1	Kshs.35,0000	Kshs.35,000
Mobile Phone	1	Kshs 17,000	Kshs.17,000
Android Studio Software	1	Free	Kshs.0
Flash Disk	1	Kshs.800	Kshs.800
TOTAL SUM	**4 Items**	**Kshs.52,800**	**Kshs.52,800**

B. Project Schedule

This is a breakdown of all the tasks that were carried out in the course of my project endeavors. The table below is of my Work breakdown Schedule and it gives an estimate of the duration of each task day as I expect to complete each task with objectives for each

Table No.2

Task Name	Duration	Start	Finish	Predecessor	Deliverables
DAIRY FARMING MANAGEMENT APP	**74 days**	**Fri 11/09/20**	**Wed 09/12/20**		Project Plan

Planning	**6 days**	**Fri 11/09/20**	**Thu 17/09/20**		Information to guide my efforts and objectives
Background research	1 day	Fri 11/09/20	Sat 12/09/20		A project to embark on
Identify a system of choice	1 day	Sun 13/09/20	Sun 13/09/20	2	Conclusion on viability of the project
Conduct feasibility analysis	1 day	Mon 14/09/20	Mon 14/09/20	3	List of requirements
Identify Requirements	1 day	Tue 15/09/20	Tue 15/09/20	4	Budget Plan
Budget for the requirements	1 day	Wed 16/09/20	Wed 16/09/20	5	A guiding schedule
Create a project schedule	1 day	Thu 17/09/20	Thu 17/09/20	6	Deliverables
Analysis/Software Requirements	**6 days**				
Conduct analysis and identify best methodology	1 day	Fri 18/09/20	Fri 18/09/20	7,6	A methodology to go with and a proper approach
Develop objectives	1 day	Sat 19/09/20	Sat 19/09/20	9	A list of goals and objectives

Gather and obtain Requirements	2 days	Sun 20/09/20	Mon 21/09/20	10	Actual software and hardware requirements
Begin and Update Proposal	2 days	Tue 22/09/20	Wed 23/09/20	11	Started chapters of the Proposal document
System Design	**16 days**				
Match requirements	2 days	Thu 24/09/20	Fri 25/09/20	12	Connection between all elements
Develop Functional requirements	2 days	Sun 27/09/20	Mon 28/09/20	14	List of Functional requirements
Develop Non-Functional requirements	2 days	Tue 29/09/20	Wed 30/09/20	15	List of Non-Functional requirements
Create a pencil sketch	2 days	Thu 01/10/20	Fri 02/10/20	16,15,14	Paper sketch of the system
Develop a Use Case diagram	2 days	Sun 04/10/20	Mon 05/10/20	17	UML system interface representation
Develop an ERD	2 days	Tue 06/10/20	Wed 07/10/20	18	ERD system database representation
Create a summary Flow Chart	2 days	Thu 08/10/20	Fri 09/10/20	19	Flow chart system representation
Update Proposal and complete	2 days	Sat 10/10/20	Sun 11/10/20	20	Complete 5 chapter proposal

Implementation	**33 days**				
Use design to ID modules	1 day	Mon 12/10/20	Mon 12/10/20		Modular system design
Develop Modules	15 days	Tue 13/10/20	Mon 02/11/20	23	Functional system units
Test modules	2 days	Tue 03/11/20	Wed 04/11/20	24	Tested and error free units
Integrate modules	20 days	Thu 05/11/20	Tue 01/12/20	25	Fully formed, unified system
Final Testing	**4 days**				
Test running of the app	3 days	Wed 02/12/20	Fri 04/12/20		Complete assurance of proper running
Test app functionality on phone	1 day	Sat 05/12/20	Sat 05/12/20		Complete assurance of functionality
Deployment	**0 days**				
Publish the application	0 days	Sun 06/12/20	Sun 06/12/20		-
Documentation	**3 days**				
Complete Project report	3 days	Mon 07/12/20	Wed 09/12/20		A fully detailed report
Maintenance	**0 days**	**Sun 11/12/20**	**Sun 11/12/20**		

Update and modify	0 days	Sun 11/12/20	Sun 11/12/20		-

The Schedule was obscured around the testing time as a result of unanticipated obstacles and challenges such as; a cracked screen, failing Safaricom WiFi in our area and Android Studio hangs and crushes.

However, as intended, the project system was complete by the 11th of December 2020 with testing being completed on the previous day. Documentation of the project has been completed as of 17th December, 2020. Maintenance as a continuous effort is underway.

C. Sample Codes

Below is a set of codes that I used to implement one module of the application i.e; The Cow Module. They describe how I implemented the functionality for adding and viewing cow records.

MainActivity.java

This is the starting activity whereby the user selects either to add or view records

```
package com.example.dfma_app_656995;

import android.annotation.SuppressLint;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.viewpager.widget.ViewPager;

public class MainActivity extends AppCompatActivity {

    DataBaseHelper myDb;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ViewPager mViewPager = findViewById(R.id.viewPage);
        ImageAdapter adapterView = new ImageAdapter(this);
        mViewPager.setAdapter(adapterView);

        Button addButton = findViewById(R.id.add_bt);
        addButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                addOnButtonClick();
            }
        });

        Button viewButton = findViewById(R.id.view_bt);
        viewButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                viewOnButtonClick();
            }
        });

        Button AboutButton = findViewById(R.id.info_bt);
        AboutButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                aboutOnButtonClick();
            }
        });

      }

    private void aboutOnButtonClick() {
        Intent i = new Intent(MainActivity.this, About.class);
        startActivity(i);
    }

    public void addOnButtonClick() {
        Intent i = new Intent(MainActivity.this, SelectEventActivity.class);
        startActivity(i);
    }

    public void viewOnButtonClick() {
        Intent i = new Intent(MainActivity.this, ViewRecordsActivity.class);
        startActivity(i);
    }
    @SuppressLint("NonConstantResourceId")
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // TODO Auto-generated method stub
        Intent i;
        switch (item.getItemId()) {

            case R.id.add_bt:
                i = new Intent(MainActivity.this, SelectEventActivity.class);
                startActivity(i);
                Toast.makeText(this, item.getTitle(), Toast.LENGTH_LONG).show();
                return true;

            case R.id.view_bt:
                Toast.makeText(this, item.getTitle(), Toast.LENGTH_LONG).show();
                i = new Intent(MainActivity.this, ViewRecordsActivity.class);
                startActivity(i);
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

Activity Main.xml

This is supporting layout activity for the Main Activity above that generates the interface that the user sees and interacts with

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <LinearLayout
        android:id="@+id/LinearLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <androidx.viewpager.widget.ViewPager
            android:id="@+id/viewPage"
            android:layout_width="match_parent"
            android:layout_height="212dp"
            android:layout_marginTop="20dp"
            android:background="@android:color/background_dark" />

        <ImageView
            android:layout_width="214dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:layout_marginLeft="70dp"
            android:src="@drawable/hbaryellow"
            android:contentDescription="@string/todo"
            android:layout_marginStart="70dp" />

        <ImageView
            android:layout_width="227dp"
            android:layout_height="120dp"
            android:layout_gravity="center"
            android:src="@drawable/farmers15"
            android:contentDescription="@string/todo" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="10dp"
        android:text="@string/choice"
        android:textSize="20sp"
        android:textStyle="bold" />

        <Button
            android:id="@+id/add_bt"
            android:layout_width="221dp"
            android:layout_height="51dp"
            android:layout_gravity="center"
            android:layout_marginTop="20dp"
            android:background="@color/light"
            android:drawableLeft="@drawable/drawable_add"
            android:onClick="addOnButtonClick"
            android:text="@string/new_records"
            android:textSize="15sp"
            android:drawableStart="@drawable/drawable_add" />

        <Button
        android:id="@+id/view_bt"
        android:layout_width="221dp"
        android:layout_height="51dp"
        android:layout_gravity="center"
        android:layout_marginTop="15dp"
            android:background="@color/light"
        android:drawableLeft="@drawable/drawable_view"
        android:onClick="viewOnButtonClick"
        android:text="@string/view_records"
        android:textSize="15sp"
            android:drawableStart="@drawable/drawable_view" />

        <include layout="@layout/settings_panel" />

        <ImageView
            android:layout_width="214dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="70dp"
            android:src="@drawable/hbaryellow"
            android:contentDescription="@string/todo"
            android:layout_marginStart="70dp" />

    </LinearLayout>

</ScrollView>
```

Cow Records Activity.java

This is the activity that supports the taking in of user-inputted data from the layout below

```
package com.example.dfma_app_656995;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.app.DatePickerDialog;
import android.app.Dialog;
import android.app.TimePickerDialog;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
```

```java
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.TimePicker;
import android.widget.Toast;

import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import java.io.Serializable;
import java.util.Calendar;

public class CowRecordsActivity extends Activity implements Serializable {
    EditText edit;
    private int mYear;
    private int mMonth;
    private int mDay;
    static final int DATE_DIALOG_ID = 0;

    public FirebaseDatabase Database = FirebaseDatabase.getInstance();
    public DatabaseReference ref;
    public DataBaseHelper myDb;
    public EditText tag, breed, source, sourceContact, dob, mother, father, category;

    @SuppressLint("WrongViewCast")
    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.cow_records);
        tag = (EditText) findViewById(R.id.e_tag);
        breed = (EditText) findViewById(R.id.e_breed);
        source = (EditText) findViewById(R.id.e_source);
        sourceContact = (EditText) findViewById(R.id.e_source_contact);
        dob = (EditText) findViewById(R.id.e_date_born);
        mother = (EditText) findViewById(R.id.e_mother);
        father = (EditText) findViewById(R.id.e_father);
        category = findViewById(R.id.e_category);
        myDb = new DataBaseHelper();
        ref = Database.getReference().child("Cow");

        Button cowButton = findViewById(R.id.view_cow_bt);
        cowButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                viewCowOnButtonClick();
            }
        });

        Button saveCow = (Button) findViewById(R.id.save_cow);
        saveCow.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                if (tag.getText().toString().isEmpty()) {
                    tag.setError("Please Enter Tag");
                }
                else if(breed.getText().toString().isEmpty()) {
                    breed.setError("Please Enter Breed");
                }
                else if(source.getText().toString().isEmpty()) {
                    source.setError("Please Enter the Source");
                }
                else if(dob.getText().toString().isEmpty()) {
                    dob.setError("Please Enter Date of Birth");
                }
                else if(category.getText().toString().isEmpty()) {
                    category.setError("Please Enter Cow Category");
                }
                else {
                    String id = ref.push().getKey();
                    //myDb.setId(id);
                    myDb.setTag(tag.getText().toString().trim());
                    myDb.setBreed(breed.getText().toString().trim());
                    myDb.setSource(source.getText().toString().trim());
                    //myDb.setSourceContact(sourceContact.getText().toString().trim());
                    myDb.setDob(dob.getText().toString().trim());
                    myDb.setMother(mother.getText().toString().trim());
                    myDb.setFather(father.getText().toString().trim());
                    myDb.setCategory(category.getText().toString().trim());

                    ref.push().setValue(myDb);
                    Toast.makeText(CowRecordsActivity.this, "Insertion Success",
Toast.LENGTH_SHORT).show();
 //
                }
            }
        });
    }

    private void viewCowOnButtonClick() {
        Intent i=new Intent(this,ViewCowActivity.class);
        startActivity(i);
    }

    public void datePickOnButtonClick(View v) {
        edit = (EditText) findViewById(R.id.e_date_born);
        EditText datePick = (EditText) findViewById(R.id.e_date_born);

        final Calendar c = Calendar.getInstance();
        mYear = c.get(Calendar.YEAR);
        mMonth = c.get(Calendar.MONTH);
        mDay = c.get(Calendar.DAY_OF_MONTH);
        updateDisplay();

        datePick.setOnClickListener(new OnClickListener() {

            public void onClick(View v) {

                showDialog(DATE_DIALOG_ID);

            }
        });
    }

    private void updateDisplay() {
        edit.setText(new StringBuilder().append(mDay).append("-")
                .append(mMonth + 1).append("-").append(mYear).append(" "));
    }

    private DatePickerDialog.OnDateSetListener mDateSetListener = new
DatePickerDialog.OnDateSetListener() {
        public void onDateSet(DatePicker view, int year, int monthOfYear,
                              int dayOfMonth) {
            mYear = year;
            mMonth = monthOfYear;
            mDay = dayOfMonth;
            updateDisplay();
        }
    };

    @Override
    protected Dialog onCreateDialog(int id) {
        switch (id) {
            case DATE_DIALOG_ID:
                return new DatePickerDialog(this, mDateSetListener, mYear, mMonth,
                       mDay);
        }

        return null;
    }

    private static String pad(int c) {
        if (c >= 10)
            return String.valueOf(c);
        else
            return "0" + String.valueOf(c);
    }
}
```

Cow Records.xml

This is the layout activity that facilitates the input of new cow records by a user and supports the functionality of the above java activity.

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/scrollView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <LinearLayout
        android:id="@+id/LinearLayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >

        <ImageView
            android:layout_width="173dp"
            android:layout_height="142dp"
            android:layout_gravity="center"
            android:layout_marginTop="10dp"
            android:src="@drawable/cows8" />
        <TextView
            android:id="@+id/cow_title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="5dp"
            android:text="@string/cow_name"
            android:textSize="25dp"
            android:textStyle="bold" />

        <LinearLayout
            android:id="@+id/LinearLayout1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="5dp"
            android:orientation="horizontal" >

            <TextView
                android:id="@+id/tag"
                android:layout_width="70dp"
                android:layout_height="wrap_content"
                android:layout_marginLeft="10dp"
                android:text="@string/tag" />

            <EditText
                android:id="@+id/e_tag"
                android:layout_width="match_parent"
                android:layout_height="40dp"
                android:layout_marginLeft="5dp"
                android:layout_marginRight="10dp"
                android:ems="10"
                android:hint="@string/hint_tag" >

                <requestFocus />
            </EditText>
        </LinearLayout>

        <LinearLayout
            android:id="@+id/LinearLayout2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="10dp"
            android:orientation="horizontal" >

            <TextView
                android:id="@+id/breed"
                android:layout_width="70dp"
                android:layout_height="wrap_content"
                android:layout_marginLeft="10dp"
                android:text="@string/breed" />

            <EditText
                android:id="@+id/e_breed"
                android:layout_width="match_parent"
                android:layout_height="40dp"
                android:layout_marginLeft="5dp"
                android:layout_marginRight="10dp"
                android:ems="10"
                android:hint="...e.g; Jersey..." >

                <requestFocus />
            </EditText>
        </LinearLayout>

        <LinearLayout
            android:id="@+id/LinearLayout3"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="10dp"
            android:orientation="horizontal" >

            <TextView
                android:id="@+id/source"
                android:layout_width="70dp"
                android:layout_height="wrap_content"
                android:layout_marginLeft="10dp"
                android:text="@string/source" />

            <EditText
                android:id="@+id/e_source"
                android:layout_width="match_parent"
                android:layout_height="40dp"
                android:layout_marginLeft="5dp"
                android:layout_marginRight="10dp"
                android:ems="10"
                android:hint="@string/hint_source" >

                <requestFocus />
            </EditText>
        </LinearLayout>

        <LinearLayout
            android:id="@+id/LinearLayout4"
            android:layout_width="match_parent"
            android:layout_height="65dp"
            android:layout_gravity="center"
            android:layout_marginTop="10dp"
            android:orientation="horizontal">

            <TextView
                android:id="@+id/source_contact"
                android:layout_width="70dp"
                android:layout_height="40dp"
                android:layout_marginLeft="10dp"
                android:text="@string/source_contact" />

            <EditText
                android:id="@+id/e_source_contact"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_marginLeft="5dp"
                android:layout_marginRight="10dp"
                android:ems="10"
                android:hint="@string/hint_source_contact"
                android:inputType="phone"
                android:maxLength="10">
                <requestFocus />
            </EditText>
        </LinearLayout>

        <LinearLayout
            android:id="@+id/LinearLayout5"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="10dp"
            android:orientation="horizontal">

            <TextView
                android:id="@+id/date_born"
                android:layout_width="71dp"
                android:layout_height="wrap_content"
                android:layout_marginLeft="10dp"
                android:inputType="date"
                android:text="@string/date_born" />

            <EditText
                android:id="@+id/e_date_born"
                android:layout_width="match_parent"
                android:layout_height="40dp"
                android:layout_marginLeft="5dp"
                android:layout_marginRight="10dp"
                android:ems="10"
                android:hint="@string/hint_date_born"
                android:inputType="date"
                android:onClick="datePickOnButtonClick" />
        </LinearLayout>

        <LinearLayout
            android:id="@+id/LinearLayout7"
            android:layout_width="match_parent"
            android:layout_height="39dp"
            android:layout_gravity="center"
            android:layout_marginTop="10dp"
            android:orientation="horizontal">

            <TextView
                android:id="@+id/mother"
                android:layout_width="70dp"
                android:layout_height="wrap_content"
                android:layout_marginLeft="10dp"
                android:text="@string/mother" />

            <EditText
                android:id="@+id/e_mother"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_marginLeft="5dp"
                android:layout_marginRight="10dp"
                android:ems="10"
                android:hint="@string/hint_mother" />
        </LinearLayout>
```

```xml
        <LinearLayout
            android:id="@+id/LinearLayout8"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="10dp"
            android:orientation="horizontal" >

            <TextView
                android:id="@+id/father"
                android:layout_width="70dp"
                android:layout_height="wrap_content"
                android:layout_marginLeft="10dp"
                android:text="@string/father" />

            <EditText
                android:id="@+id/e_father"
                android:layout_width="match_parent"
                android:layout_height="40dp"
                android:layout_marginLeft="5dp"
                android:layout_marginRight="10dp"
                android:ems="10"
                android:hint="@string/hint_father"/>
        </LinearLayout>

        <LinearLayout
            android:id="@+id/LinearLayout9"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp" >

            <TextView
                android:id="@+id/category"
                android:layout_width="70dp"
                android:layout_height="wrap_content"
                android:layout_marginLeft="10dp"
                android:text="@string/category" />

            <EditText
                android:id="@+id/e_category"
                android:layout_width="match_parent"
                android:layout_height="40dp"
                android:layout_marginLeft="5dp"
                android:layout_marginRight="10dp"
                android:ems="10"
                android:hint="@string/hint_category"/>
        </LinearLayout>

        <LinearLayout
            android:id="@+id/LinearLayout10"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="10dp"
            android:orientation="horizontal" />
        <LinearLayout
            android:orientation="horizontal"
            android:padding="8sp"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content">

            <Button
                style="@style/edit_button"
                android:id="@+id/save_cow"
                android:drawableTop="@drawable/drawable_add1"
                android:background="@color/light"
                android:text="@string/save"
                android:textColor="@color/inverted" />

            <Button
                style="@style/edit_button"
                android:id="@+id/view_cow_bt"
                android:drawableTop="@drawable/drawable_view1"
                android:background="@color/light"
                android:text="@string/view_records"
                android:textColor="@color/inverted" />

        </LinearLayout>
    </LinearLayout>

</ScrollView>
```

DataBase Helper.java

This is the database activity that hosts setters (functions that help add input into the firebase database) of data for the above activities and getters (functions that help retrieve data from the firebase database) for the below activities.

```
package com.example.dfma_app_656995;

public class DataBaseHelper {

    private String Id, Tag, Breed, Source, Dob, Mother, Father, Category;

    public DataBaseHelper() {
    }

    public String getId() {
        return Id;
    }

    public void setId(String id) {
        Id = id;
    }

    public String getTag() {
        return Tag;
    }

    public void setTag(String tag) {
        Tag = tag;
    }

    public String getBreed() {
        return Breed;
    }

    public void setBreed(String breed) {
        Breed = breed;
    }

    public String getSource() {
        return Source;
    }

    public void setSource(String source) {
        Source = source;
    }

    public String getDob() {
        return Dob;
    }

    public void setDob(String dob) {
        Dob = dob;
    }

    public String getMother() {
        return Mother;
    }

    public void setMother(String mother) {
        Mother = mother;
    }

    public String getFather() {
        return Father;
    }

    public void setFather(String father) {
        Father = father;
    }

    public String getCategory() {
        return Category;
    }

    public void setCategory(String category) {
        Category = category;
    }
}
```

Cow Adapter.java

This is the activity that facilitates the interaction between the layout activities below and the firebase database so as to retrieve data and display it onto the textviews within the listview

```
package com.example.dfma_app_656995;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;

import com.google.firebase.database.DatabaseReference;

import java.util.ArrayList;

public class CowAdapter extends BaseAdapter {

    ArrayList<DataBaseHelper> cowHelper=new ArrayList<DataBaseHelper>();
    Context context;
    DatabaseReference ref;

    public CowAdapter(Context context, int simple_list_item_1, ArrayList<DataBaseHelper>
cowHelp){
        cowHelper= cowHelp;
        this.context=context;
    }

    @Override
    public int getCount() {
        return cowHelper.size();
    }

    @Override
    public Object getItem(int position) {
        return null;
    }

    @Override
    public long getItemId(int position) {
        return 0;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {

        LayoutInflater inflater= (LayoutInflater)
context.getSystemService(context.LAYOUT_INFLATER_SERVICE);
        View view=inflater.inflate(R.layout.view_cow_rows,parent,false);

        TextView tag = (TextView) view.findViewById(R.id.tag);
        TextView breed = (TextView) view.findViewById(R.id.breed);
        TextView source = (TextView) view.findViewById(R.id.source);
        //TextView sourceContact = (TextView) view.findViewById(R.id.source_contact);
        TextView dob = (TextView) view.findViewById(R.id.dob);
        TextView mother = (TextView) view.findViewById(R.id.mother);
        TextView father = (TextView) view.findViewById(R.id.father);
        TextView category = (TextView) view.findViewById(R.id.category);

        DataBaseHelper cowHelper1=cowHelper.get(position);

        tag.setText(cowHelper1.getTag());
        breed.setText(cowHelper1.getBreed());
        source.setText(cowHelper1.getSource());
        //sourceContact.setText(helper1.getSourceContact());
        dob.setText(cowHelper1.getDob());
        mother.setText(cowHelper1.getMother());
        father.setText(cowHelper1.getFather());
        category.setText(cowHelper1.getCategory());

        return view;
    }
}
```

Cow Image Adapter.java

This is the activity that facilitates the display of images within a viewPage that can be scrolled in the view layout activity below.

```
package com.example.dfma_app_656995;

import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;

import androidx.viewpager.widget.PagerAdapter;
import androidx.viewpager.widget.ViewPager;

public class CowImageAdapter extends PagerAdapter {
    Context mContext;

    CowImageAdapter(Context context) {
        this.mContext = context;
    }

    @Override
    public boolean isViewFromObject(View view, Object object) {
        return view == ((ImageView) object);
    }

    private int[] sliderImageId = new int[]{
            R.drawable.cowpics2, R.drawable.cowpics7, R.drawable.herdpics1,
    };

    @Override
    public Object instantiateItem(ViewGroup container, int position) {
        ImageView imageView = new ImageView(mContext);
        imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
        imageView.setImageResource(sliderImageId[position]);
        ((ViewPager) container).addView(imageView, 0);
        return imageView;
    }

    @Override
    public void destroyItem(ViewGroup container, int position, Object object) {
        ((ViewPager) container).removeView((ImageView) object);
    }

    @Override
    public int getCount() {
        return sliderImageId.length;
    }
}
```

View Cow Activity.java

This is the activity that facilitates display of retrieved data from the firebase database on the listview placed in the layout activity below

```
package com.example.dfma_app_656995;

import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.ListView;
import android.widget.Toast;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.viewpager.widget.ViewPager;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;

public class ViewCowActivity extends AppCompatActivity {
    ListView listView;
    ProgressDialog progressDialog;
    Button add, home;
    DatabaseReference databaseReference;
    ArrayList<DataBaseHelper> cowHelper = new ArrayList<>();
    ArrayList<String> helper = new ArrayList<>();
    DataBaseHelper Db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.view_cow_activity);

        ViewPager mViewPager = findViewById(R.id.viewCowPage);
        CowImageAdapter cowAdapterView = new CowImageAdapter(this);
        mViewPager.setAdapter(cowAdapterView);
        add = findViewById(R.id.add);
        add.setOnClickListener(new View.OnClickListener() {
                                   @Override
                                   public void onClick(View v) {
                                       addOnClickListener();
                                   }
                               }
        );

        home = findViewById(R.id.home);
        home.setOnClickListener(new View.OnClickListener() {
                                    @Override
                                    public void onClick(View v) {
                                        homeOnClickListener();
                                    }
                                }
        );

        listView = (ListView) findViewById(R.id.cowView);
        final CowAdapter cowAdapter = new CowAdapter(ViewCowActivity.this,
android.R.layout.simple_list_item_1,cowHelper);
        listView.setAdapter(cowAdapter);

        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long
id) {
                Db.setId(helper.get(position));
                //Toast.makeText(getApplicationContext(), "You have selected an item:" +
cowAdapter.getItem(position),Toast.LENGTH_LONG).show();
            }
        });

        progressDialog = new ProgressDialog(this);
        databaseReference =
FirebaseDatabase.getInstance().getReferenceFromUrl("https://dairy-farming-
management.firebaseio.com//Cow");
        progressDialog.setMessage("Server Loading, Please wait");
        progressDialog.show();

        databaseReference.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                String Id;
                progressDialog.dismiss();
                for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                    DataBaseHelper cowHelper1 = snapshot.getValue(DataBaseHelper.class);
                    cowHelper.add(cowHelper1);
                }

                CowAdapter adapterCow = new CowAdapter(ViewCowActivity.this,
android.R.layout.simple_list_item_1, cowHelper);
                listView.setAdapter(adapterCow);
                listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long
id) {
                Toast.makeText(getApplicationContext(), "You have selected an item:" +
cowAdapter.getItem(position),Toast.LENGTH_LONG).show();
```

```
            }
        });
                listView.setOnItemLongClickListener(new
AdapterView.OnItemLongClickListener() {
            @Override
            public boolean onItemLongClick(AdapterView<?> parent, View view, int position,
long id) {

                final int which_item = position;
                new AlertDialog.Builder(ViewCowActivity.this)
                        .setIcon(android.R.drawable.ic_delete)
                        .setTitle("Are you sure?")
                        .setMessage("Would you like to delete this?")
                        .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                cowHelper.remove(which_item);
                                cowAdapter.notifyDataSetChanged();
                            }
                        })
                        .setNegativeButton("No",null)
                        .show();

                    return true;
                    }
                    });

            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });
    }

    private void homeOnClickListener() {
        Intent i = new Intent(ViewCowActivity.this, MainActivity.class);
        startActivity(i);
    }

    private void addOnClickListener() {
        Intent i = new Intent(ViewCowActivity.this, CowRecordsActivity.class);
        startActivity(i);
    }

}
```

View Cow Activity.xml

This is the layout activity that works with the java activity above to produce a view of the retrieved data on a listview so that the user sees the data within a scrollable space

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/scrollView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <TextView
        android:id="@+id/cow_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginLeft="100dp"
        android:layout_marginTop="20dp"
        android:text="@string/cow_title"
        android:textSize="25dp"
        android:textStyle="bold" />

    <androidx.viewpager.widget.ViewPager
        android:id="@+id/viewCowPage"
        android:layout_width="match_parent"
        android:layout_height="212dp"
        android:layout_below="@+id/cow_title"
        android:layout_marginTop="20dp"
        android:background="@android:color/background_dark" />

    <LinearLayout
        android:orientation="horizontal"
        android:id="@+id/layout"
        android:padding="8sp"
        android:layout_below="@+id/viewCowPage"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">

        <Button
            style="@style/edit_button"
            android:id="@+id/add"
            android:drawableTop="@drawable/drawable_add1"
            android:background="@color/light"
            android:text="@string/add"
            android:textColor="@color/inverted" />

        <Button
            style="@style/edit_button"
            android:id="@+id/home"
            android:drawableTop="@drawable/drawable_welcome"
            android:background="@color/light"
            android:text="@string/home"
            android:textColor="@color/inverted" />
    </LinearLayout>

    <ListView
            android:id="@+id/cowView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@+id/layout"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="10dp"
            android:textSize="15dp" >
    </ListView>
```

```
</RelativeLayout>
```

View Cow Rows.xml

This is the layout activity that works with the view cow layout activity above to display the retrieved data on individual text views within the larger listview

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/scrollView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <RelativeLayout
        android:id="@+id/RelLayout2"
        android:layout_width="match_parent"
        android:layout_height="153dp"
        android:layout_marginTop="10dp"
        android:orientation="vertical">

        <de.hdodenhof.circleimageview.CircleImageView
            android:id="@+id/cow_icon"
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:layout_marginLeft="30dp"
            android:layout_marginTop="40dp"
            android:src="@drawable/cowicon" />

        <ImageView
            android:id="@+id/bar_icon_1"
            android:layout_width="10dp"
            android:layout_height="135dp"
            android:layout_marginLeft="20dp"
            android:layout_toRightOf="@+id/cow_icon"
            android:src="@drawable/baryellow" />

        <TextView
            android:id="@+id/tv_view"
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:layout_below="@+id/cow_icon"
            android:layout_gravity="center_horizontal"
            android:layout_marginLeft="30dp"
            android:layout_marginTop="5dp"
            android:text="@string/Retrieved"
            android:textSize="15dp"
            android:textStyle="italic" />

        <TextView
            android:id="@+id/tv_tag"
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginLeft="40dp"
            android:layout_toRightOf="@+id/cow_icon"
            android:text="@string/tag"
            android:textSize="15dp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/tag"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginLeft="5dp"
            android:layout_toRightOf="@+id/tv_tag"
            android:text="@string/tag"
            android:textSize="15dp" />

        <TextView
            android:id="@+id/tv_breed"
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:layout_below="@+id/tag"
            android:layout_gravity="center_horizontal"
            android:layout_marginLeft="40dp"
            android:layout_toRightOf="@+id/cow_icon"
            android:text="@string/breed"
            android:textSize="15dp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/breed"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/tag"
            android:layout_gravity="center_horizontal"
            android:layout_marginLeft="5dp"
            android:layout_toRightOf="@+id/tv_breed"
            android:text="@string/breed"
            android:textSize="15dp" />

        <TextView
            android:id="@+id/tv_source"
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:layout_below="@+id/breed"
            android:layout_gravity="center_horizontal"
            android:layout_marginLeft="40dp"
            android:layout_toRightOf="@+id/cow_icon"
            android:text="@string/source"
            android:textSize="15dp"
            android:textStyle="bold" />

        <TextView
```

```
        android:id="@+id/source"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/breed"
        android:layout_gravity="center_horizontal"
        android:layout_marginLeft="5dp"
        android:layout_toRightOf="@+id/tv_source"
        android:text="@string/source"
        android:textSize="15dp" />

    <TextView
        android:id="@+id/tv_dob"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_below="@+id/source"
        android:layout_gravity="center_horizontal"
        android:layout_marginLeft="40dp"
        android:layout_toRightOf="@+id/cow_icon"
        android:text="@string/date_born"
        android:textSize="15dp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/dob"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/source"
        android:layout_gravity="center_horizontal"
        android:layout_marginLeft="5dp"
        android:layout_toRightOf="@+id/tv_dob"
        android:text="@string/date_born"
        android:textSize="15dp" />

    <TextView
        android:id="@+id/tv_mother"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_below="@+id/dob"
        android:layout_gravity="center_horizontal"
        android:layout_marginLeft="40dp"
        android:layout_toRightOf="@+id/cow_icon"
        android:text="@string/mother"
        android:textSize="15dp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/mother"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/dob"
        android:layout_gravity="center_horizontal"
        android:layout_marginLeft="5dp"
        android:layout_toRightOf="@+id/tv_mother"
        android:text="@string/mother"
        android:textSize="15dp" />

    <TextView
        android:id="@+id/tv_father"
        android:layout_width="100dp"
```

```
            android:layout_height="wrap_content"
            android:layout_below="@+id/mother"
            android:layout_gravity="center_horizontal"
            android:layout_marginLeft="40dp"
            android:layout_toRightOf="@+id/cow_icon"
            android:text="@string/father"
            android:textSize="15dp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/father"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/mother"
            android:layout_marginLeft="5dp"
            android:layout_toRightOf="@+id/tv_father"
            android:text="@string/father"
            android:textSize="15dp" />
        <TextView
            android:id="@+id/tv_category"
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:layout_below="@+id/father"
            android:layout_marginLeft="40dp"
            android:layout_toRightOf="@+id/cow_icon"
            android:text="@string/category"
            android:textSize="15dp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/category"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/father"
            android:layout_marginLeft="5dp"
            android:layout_toRightOf="@+id/tv_category"
            android:text="@string/category"
            android:textSize="15dp" />
        <ImageView
            android:id="@+id/bar_icon_3"
            android:layout_width="90dp"
            android:layout_height="135dp"
            android:layout_marginLeft="200dp"
            android:layout_toRightOf="@+id/cow_icon"
            android:src="@drawable/baryellow" />
    </RelativeLayout>
</androidx.cardview.widget.CardView>
```

D.Screenshots

Below is a set of screenshots showing; the main activity (activity main layout) for add or view selection buttons upon which we select add. The next layout shows selection of cow, milk, health or reminder additions upon which we select cow. The cow records activity is called upon and the layout shown (cow records layout). We add records as demonstrated and in case of any fields being left blank, there is validation and checking to ensure no record is blank. Upon filling the fields and saving, records are inserted and upon clicking “view records”. The view cow activity is called upon and the layout shown (view cow activity layout in conjunction with view cow rows layout) with the retrieved data displayed and our demonstration/example data is shown.

Fig4 - First page

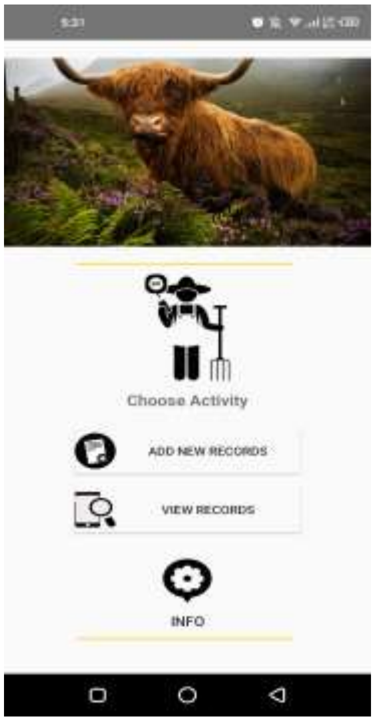


Fig5 - Selection Page

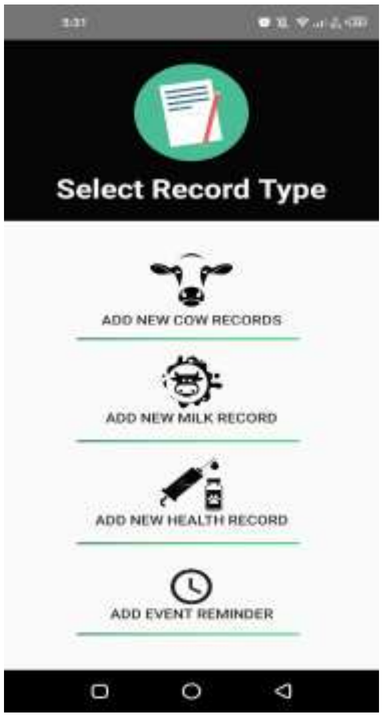
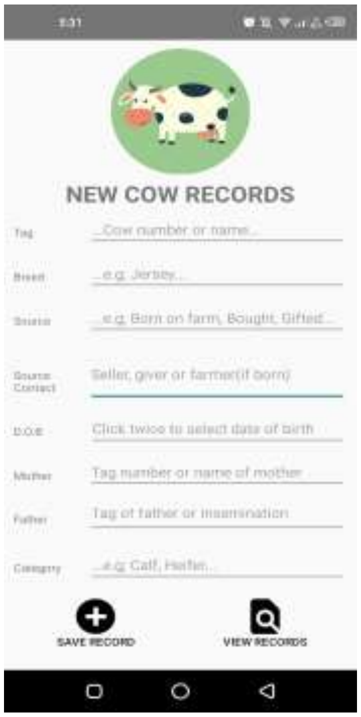
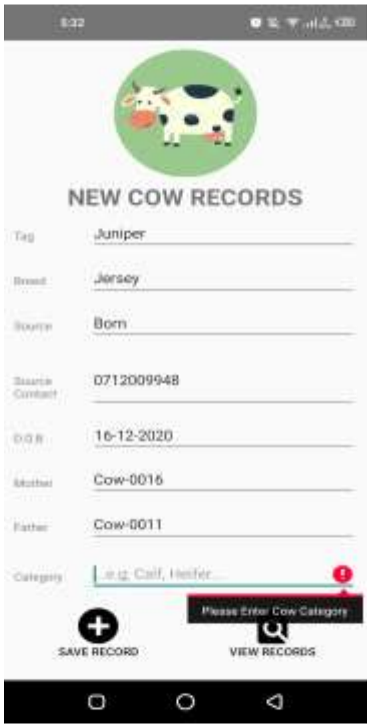


Fig6 - Cow Record Addition Page


NEW COW RECORDS

Tag	...Cow number or name...
Breed	...e.g. Jersey...
Source	...e.g. Born on farm, Bought, Gifted...
Source Contact	Seller, giver or farmer(if born)
D.O.B	Click twice to select date of birth
Mother	Tag number or name of mother
Father	Tag of father or insemination
Category	...e.g. Calf, Heifer...

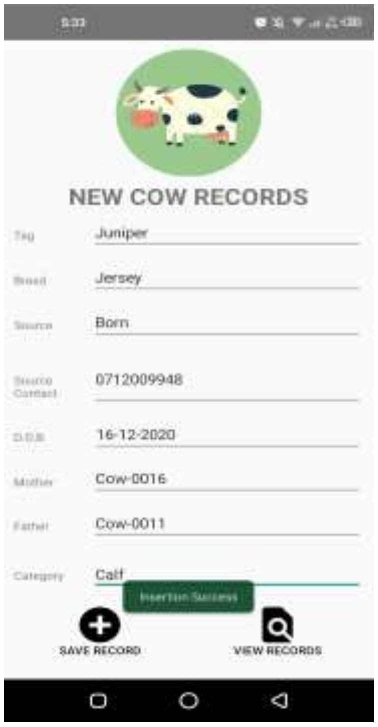
Fig7 - Filling & Validation


NEW COW RECORDS

Tag	Juniper
Breed	Jersey
Source	Born
Source Contact	0712009948
D.O.B	16-12-2020
Mother	Cow-0016
Father	Cow-0011
Category	...e.g. Calf, Heifer...

Please Enter Cow Category

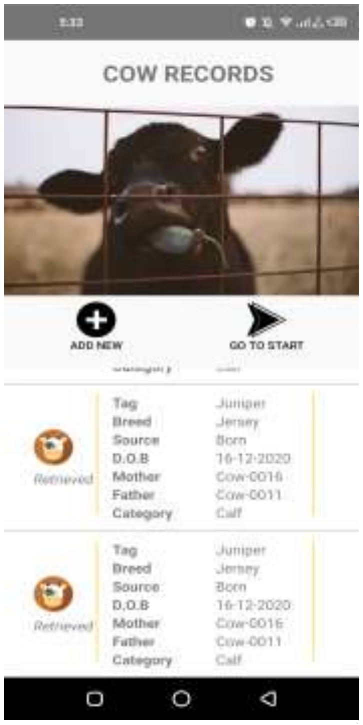
Fig8 - Saving and Insertion


NEW COW RECORDS

Tag	Juniper
Breed	Jersey
Source	Born
Source Contact	0712009948
D.O.B	16-12-2020
Mother	Cow-0016
Father	Cow-0011
Category	Calf

Insertion Success

Fig 9 - Records View


COW RECORDS

Retrieved

Tag	Juniper
Breed	Jersey
Source	Born
D.O.B	16-12-2020
Mother	Cow-0016
Father	Cow-0011
Category	Calf

Retrieved

Tag	Juniper
Breed	Jersey
Source	Born
D.O.B	16-12-2020
Mother	Cow-0016
Father	Cow-0011
Category	Calf