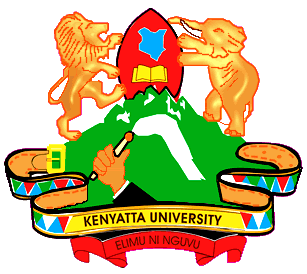
****

**KENYATTA UNIVERSITY**

**SCHOOL OF ENGENEERING AND TECHNOLOGY**

**C.I.T DEPARTMENT**

**B.SC COMPUTER SCIENCE**

**SCO400: PROJECT FINAL DOCUMENT**

**CROSS-PLATFORM MOBILE APPLICATION FOR DAIRY COOPERATIVES**

**CHERUIYOT IAN K**

**J17/0601/2014**

**PROJECT SUPERVISOR: Ms. GYANTI THAKUR**

**A PROJECT PAPER SUBMITTED IN PARTIAL FULFILMENT OF THE**

**REQUIREMENT FOR THE AWARD OF THE DEGREE OF BACHELOR OF SCIENCE (COMPUTER SCIENCE) SUBMITED TO THE DEPARTMENT OF COMPUTING INFORMATION & TECHNOLOGY**

# **ABSTRACT**

Project development is a crucial and fundamental unit in Computer Science. This is because the project is what is used to gauge the practical understanding of the student in most of the units taught during the curriculum. It tests the student’s skills in system analysis and design during the project development, Entrepreneurship in terms of what solution is the student trying to provide in the current market and system skills development in the language of the student’s choice together with tools used. The proposed system targets milk farmers by providing a mobile application, native to both android and iOS systems which automates the existing manual system used by farmers to keep track of their milk collection and account for the returns on collected milk. The system is client-based for the farmer and expected time to completion is three months.

# **DEDICATION**

I dedicate this project to my dear mum, relatives, supervisor, classmates and friends who have been a source of encouragement and insight throughout its development and who have in different ways supported and inspired me from the start of the project to its completion.

# **DECLARATION**

I have submitted this project on my own behalf. I declare that the project submitted is original except for the source material acknowledged. The piece of work has not been submitted for more than one purpose without declaration.

I acknowledge that I am aware of the University policy and regulations on the honesty in academic work and the disciplinary guidelines and procedure that come with breach of these policies and regulations as contained in the universities student’s handbook.

This project work is submitted in the partial fulfilment of the requirements for the award of the degree of Bachelor of Science in Computer Science. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

**STUDENT:** **CHERUIYOT IAN K**

Signature ……………………………………

Date ……………………………………

**Approvals**

I hereby submit this project documentation for examination with the approval of the project supervisor.

**SUPERVISOR:** **Ms. GYANTI THAKUR**

Signature ……………………………………

Date ……………………………………

Table of Contents

[**ABSTRACT** iii](#_Toc55784874)

[**DEDICATION** iv](#_Toc55784875)

[**DECLARATION** v](#_Toc55784876)

[**CHAPTER 1: INTRODUCTION** 1](#_Toc55784877)

[**1.1 Background** 1](#_Toc55784878)

[**1.2 Problem Statement** 2](#_Toc55784879)

[**1.3 Objectives** 2](#_Toc55784880)

[**1.4 Scope** 3](#_Toc55784881)

[**1.5 Limitation** 3](#_Toc55784882)

[**1.6 Justification** 3](#_Toc55784883)

[**CHAPTER 2: LITERATURE REVIEW** 4](#_Toc55784884)

[**CHAPTER 3: METHODOLOGY** 7](#_Toc55784885)

[**3.1 Introduction** 7](#_Toc55784886)

[**3.2 Development Methodology** 7](#_Toc55784887)

[**3.3 Data collection Methods** 7](#_Toc55784888)

[**3.3.1 Observation** 8](#_Toc55784889)

[**3.3.2 Documents and record review** 8](#_Toc55784890)

[**3.4 Data Analysis Techniques** 8](#_Toc55784891)

[**3.4.1 Data Flow Diagrams** 8](#_Toc55784892)

[**3.4.2 Use Case Diagram** 8](#_Toc55784893)

[**3.5 Application Development Tools** 8](#_Toc55784894)

[**3.5.1 Software tools** 8](#_Toc55784895)

[**3.6 Application Testing tools** 9](#_Toc55784896)

[**3.7 Project Schedule** 9](#_Toc55784897)

[**3.8 Time Schedule** 9](#_Toc55784898)

[**3.8 Project Budget** 10](#_Toc55784899)

[**CHAPTER 4: SYSTEM ANALYSIS AND REQUIREMENT MODELING** 12](#_Toc55784900)

[**4.1 Introduction** 12](#_Toc55784901)

[**4.2 Description of the current system** 13](#_Toc55784902)

[**4.3 Current System Data flow Diagrams** 14](#_Toc55784903)

[**4.3.1Context diagram of the proposed system(Level 0 DFD)** 14](#_Toc55784904)

[**4.3.2 Use Case Modelling of Current System** 15](#_Toc55784905)

[**4.4 Description of the proposed system** 15](#_Toc55784906)

[4**.5 Proposed System Data Flow Diagrams** 16](#_Toc55784907)

[**4.5.1 Context Diagram (Level 0 DFD)** 16](#_Toc55784908)

[**4.5.2 Context Diagram (Level 1 DFD)** 16](#_Toc55784909)

[**4.6 Methods of Data Collection** 17](#_Toc55784910)

[**4.6.1 Analysis of existing documents** 17](#_Toc55784911)

[**4.6.2 Interviews** 20](#_Toc55784912)

[**CHAPTER 5: SYSTEM DESIGN** 22](#_Toc55784913)

[**5.1 Introduction** 22](#_Toc55784914)

[**5.2 Database entities** 23](#_Toc55784915)

[**5.2.1 Database Collections** 23](#_Toc55784916)

[**5.2 User Interface** 25](#_Toc55784917)

[**CHAPTER 6: SYSTEM IMPLEMENTATION** 29](#_Toc55784918)

[**6.1 TOOLS USED FOR CODING AND IMPLEMENTATION** 29](#_Toc55784919)

[**6.1.1 Operating System** 29](#_Toc55784920)

[**6.1.2 Development Platforms** 29](#_Toc55784921)

[**6.1.3 Development Languages** 29](#_Toc55784922)

[**6.1.4 Hardware Tools** 29](#_Toc55784923)

[**6.2 Testing** 30](#_Toc55784924)

[**6.2.1 Unit Test** 30](#_Toc55784925)

[**6.2.3 Functional testing** 30](#_Toc55784926)

[**6.2.4 Performance testing** 31](#_Toc55784927)

[**6.2.5 User acceptance test** 31](#_Toc55784928)

[**6.3 System Maintenance** 31](#_Toc55784929)

[**CHAPTER 7: LIMITATIONS, RECOMMENDATIONS AND CONCLUSION** 33](#_Toc55784930)

[7.1 Challenges 33](#_Toc55784931)

[7.2 Limitations 33](#_Toc55784932)

[7.3 Recommendations 33](#_Toc55784933)

[**7.4 Conclusion** 34](#_Toc55784934)

[**REFERENCES** 35](#_Toc55784935)

[**APPENDIX A: USER MANUAL** 36](#_Toc55784936)

[**APPENDIX B: IMPORTANT CODE** 37](#_Toc55784937)

# **CHAPTER 1: INTRODUCTION**

## **1.1 Background**

Farmer cooperatives play a crucial role in facilitating the productivity and growth of small scale farmers in rural areas. Profitable agricultural value chains in Kenya operate through farmer cooperatives, that play a crucial role in market access, access to farm inputs, and access to information. However, structural challenges have always undermined the potential and productivity of the farmer cooperatives. Cooperatives stand a position of great influence in helping empower farmers and to help stimulate ensuring the sustainability of agriculture in many value chains. There is a need for a multi-stakeholder approach in equipping these cooperatives with the right tools to enhance growth and transparency as well as to increase the profitability of the associated value chains. Among the key areas of interest are record-keeping and transparency of cooperative activities as well as transparency between the farmers themselves and their respective cooperatives.

Ngucici Dairy is a dairy farmer’s cooperative that was started in 04-12-1978 in Kisima ward, Meru County with registration number CS/2963, with an initial target for their milk production being to Isiolo and KCC. However, its operations were grounded in 1991. In the year 2010, its operations were revived by Maritati, Mutarakwa, and Lucerne Dairy Groups.The main goal being the aggregation of milk and selling to Meru Dairy Sacco. However, they broke off due to the marketing approach with Lucerne.Presently, Ngucici is made up of roughly 600 members with 12 collection centers and approximately 300 active members.

Initially, the cooperative handled all its data by storing in multiple excel sheets, for example, an excel sheet dedicated to expenses (payroll), milk production, payments, and member deductions. The shortcomings using this approach included a lot of time was needed in transferring records to the excel sheets. Secondly the work itself was cumbersome and finally, updates to this approach are not instantaneous as the excel sheets are updated at the end of the month. However, the cooperative has embraced a web-based portal responsible for carrying out these tasks.

Despite the new system curbing the initial milestones faced by the cooperative, the cooperative members themselves have no real-time access to this information. They have no means of receiving reports of the amount of milk delivered to the cooperative at the end of the month as well as an automated means of managing their expenses.

## **1.2 Problem Statement**

On observation of the current system used at Ngucici Cooperative, there is a need for a mobile client working on both mobile operating systems(Android and iOS) for its members. The mobile client will negate the need for members to have to visit the offices to receive the reports for their output over a given duration(one month). It will also allow members to access this information from any location and will be updated on a real-time basis. The mobile application will also have its expense manager for the members which will compute the findings over a specified period, for example, a week, a fortnight or a month.

The existing gap in the system is the lack of a mobile client for individual farmers belonging to the cooperatives from which they have access to their data as well as monitoring of their data. In addition to, they have a comprehensive reports of their contributions to the cooperative.My proposed system will close this gap by being a mobile client extending the services of the existing system built on native code to allow deployment in mobile devices running on either Android or iOS. Dart programming language bundled with the flutter framework allows for native integration with either mobile operating system. The system will run concurrently with the existing system and will contain the cooperative member information, reports on the amount of milk collected, expenses incurred over a given duration. The system will need a mobile device as well as an active internet connection for real-time updates.

## **1.3 Objectives**

* To provide system users real-time access to data changes
* To enable the system to back-trace their information for making past references
* To allow transparency between the cooperative and the system users
* To allow reporting of expenses and milk collected over a defined period using graphs.
* To enable seamless expense management

## **1.4 Scope**

The development of this project is to give the cooperative members a platform from which will ensure there is transparency between the cooperative and it's members. The mobile client will be able to keep track of the member's information, amount and monetary value of milk delivered to the cooperative as well as timely changes and updates to work done. It will also enable users to have a projection of expenses incurred.

## **1.5 Limitation**

Some of the constraints that might be encountered during the design and implementation of the project include the following

* Sensitive personal information, for example, the next of kin is restricted from alteration by the user. This is because such information requires a defined procedure within the cooperative for change.
* User has no rights in changing inputted milk records as this poses a risk of compromising the system integrity.
* The user is restricted from signing up to the system with the mobile client as he/she has only read privileges in this scenario. Signing up to the system is done by the cooperative using the parent system.
* User has no right from editing the expenses billed to him/her by the cooperative

## **1.6 Justification**

Farmers need a transparent communication channel with their respective cooperatives which in turn motivates them to work with the cooperatives. The proposed system delivers exactly this plus allows monitoring and evaluation by the farmers remotely. The proposed system also negates any previous inconveniences to the farmer such as having to visit the cooperative to obtain progress on inputs brought to the cooperative. Also, the system works with any operating system which implies the farmer is not tied down to a specific mobile device. Finally, the farmer has his/her expense manager responsible for handling any and computing any expenses incurred.

# **CHAPTER 2: LITERATURE REVIEW**

Flutter is Google's mobile app SDK(Software Development Kit), complete with a framework, widgets, and tools, that gives developers an easy way to build and deploy visually attractive, fast mobile apps on both Android and iOS platforms(Agnieszka Mroczkowska, 2019). Agnieszka further explains flutter enables a smooth and easy cross-platform mobile app development. This implies there is no need to develop an iOS and Android app separately. The only needed requirement is one code base for both platforms. This will help in saving time and reducing the workload need in building the proposed system. Instead of duplication of work, the only needed requirement is a different environment for compiling the codebase.

Flutter has more tricks up its sleeve that makes theming your app a breeze. You could go through and manually change the fonts, colors, and looks for everything one by one, but that takes way too long. Instead, Flutter provides us with something called Theme-Data that allows us to set values for colors, fonts, input fields, and much more. This feature is great for keeping the look of your app consistent.(Grandt, 2019).

According to Ganesh(2018) At its heart, Flutter might look like a hodgepodge of various Google technologies and concepts, however, this results in an improbably powerful mobile framework. He further states that flutter is based on Dart which is Google's in-house programming language, therefore, allowing fluter exclusive access to Skia graphic's library. This ensures the material design used in making the basics widgets in the application are well presented and fluid. Overall giving the mobile application an outstanding look and feel.

According to Tylor, P.S and Millar, R.J.(2010) software management tools are useful and often necessary, but the true art in software project management is the application of correct methods and then using those tools in supporting the methods. The article further warns that without a method, tools are counted worthless. Software project management methods are still evolving, but the current trend leads away from the waterfall model to a more cyclic project delivery model that imitates a software release life cycle.

Clinton(2000) explains how computer systems have become intimately involved with the objectives of an organization to the point where many organizations would be incapable of operating effectively without proper computer systems. He examines the technological issues encircling the introduction of information technology in everyday life. He describes how information technology has been used as an instrument in organizations and the effects it has had on the purpose of benefiting the organization in the long term and further improve the desired aims of the organization.

Developing mobile apps can take a lot of time considering you need to use a different codebase for Android and iOS. That is unless you use an SDK like Flutter, where you have a single codebase that allows you to build your app for both operating systems. Not only that, but you can run them completely natively. This means things such as scrolling and navigation, to name a few, act just like they should for the OS being used. To keep with the theme of simplicity, as long as you have a device or simulator running, Flutter makes building and running your app for testing as simple as clicking a button.(Grandt, 2019).

In past years, if an app needed to be [cross-platform](https://searchmobilecomputing.techtarget.com/definition/cross-platform-mobile-development) and run on multiple operating systems, there was little -- if any -- code that could be re-used from the initial development project. Essentially, each device required its own mobile app development project with its own code base. Modern cross-platform tools use common languages such as [C#](https://searchwindevelopment.techtarget.com/definition/C) and [JavaScript](https://www.theserverside.com/definition/JavaScript) to share code across projects; more importantly, they integrate well with application lifecycle management tools, such as [Jenkins](https://searchsoftwarequality.techtarget.com/definition/Jenkins). This allows developers to use a single [code base](https://whatis.techtarget.com/definition/codebase-code-base) for Apple iOS, Google Android and [progressive web apps](https://whatis.techtarget.com/definition/progressive-web-app-PWA) (PWAs). A progressive web app is a website that looks and behaves as if it is a mobile app.(Rouse, 2016)

**Case study of Dairy Milk, MilkMan**

Dairy Milk, Milkman is an android application on the google play store which serves as a milk collection center management software. The application is designed by the GDWKCG group located in Pradesh in India. Notable strongholds offered by Dairy Milk, MilkMan include an online method of milk collection. Besides, one can receive a milk summary as well as a rating feature for the milk received. The platform offers an adding farmer functionality where you can add farmers to the platform. Finally, the platform offers an editable tab for the collection center information. This entails all the information regarding a particular software is deployed.

The gap in the existing system is the slow performance due to its large application file. In addition to, a poor user interface and experience. Finally the existing system is flooded with ads making navigation within the application cumbersome. The proposed system runs on native code which delivers small application size but very efficient and scalable. The systems backbone is based on the BloC(Business Logic Component) architecture recommended by google developers which further accounts for modularity and scalability in the application. Flutter frameworks is equipped with widgets which allow for building user-friendly interfaces as well as a comprehensive user experience for the end-user.

**Case study forMdairyman app for dairy**

Mdairyman application is free to download on the google play store and is produced by Kudos Software. The target audience for the application in the Indian market. The application allows for adding milk and storing of milk records. Through the application, a user can rate his/her product as well as account for miscellaneous expenses and cattle feeds.

The gap in the existing system is the lack for an iOS client for the platform as well as a non-user friendly interface and experience. My proposed system is based on material design in terms of textures and widgets supported by google which are readily available in flutter, the core framework for my system. Using material design for my proposed system offers a friendly user-interface as well as experience. The native support for my proposed system allows portability into iOS mobile device in a quicker and efficient way.

# **CHAPTER 3: METHODOLOGY**

## **3.1 Introduction**

This chapter focuses on the methods that have been used in the collection and analysis of data for the project. It explains the research design and data collection methods used and describes how data collected from the research has been analyzed.

The research method to be used is qualitative research which is usually associated with getting people's views and thoughts. This involves collecting data, data analysis and attempting to uncover the deeper meaning to the collective data. Collection of data will be done through observation and interviews and both descriptive data analysis and inferential data analysis will be used to analyze the data.

## **3.2 Development Methodology**

Rapid Application Development (RAD) is the methodology that will be used in the development of this project as it emphasizes iterative programming rather than the typical waterfall development methods which emphasize planning first

The RAD programming approach is suitable for the proposed system as it features the inclusion g and heavy use of prototyping. The use of prototypes throughout the development cycle provides several benefits:

* User involvement – a rapid prototype allows users to use the system and provide feedback rather than attempting to provide abstract evaluations of a design documentation
* Feasibility – prototyping allows the development team to quickly evaluate the feasibility of a particular complex component. By recognizing and working on complicated systems early in the development cycle, the application will be more robust, less error-prone and better structured for future design additions
* Error reduction and debugging – with rapid prototype releases during a project, it is far more likely that errors will be both discovered and fixed earlier in the development cycle

## **3.3 Data collection Methods**

### **3.3.1 Observation**

One of the methods that will be used in collecting facts is through observation. We have different types of observation techniques: natural observation, controlled observation, and participant observation. The premise behind this technique is to understand how the administrators interact with the web portal and replicate a similar feel for the mobile application

### **3.3.2 Documents and record review**

This method involves reviews of existing documents such as data files. The reason for using this approach is because the information required has already been collected for other purposes. I will use the cashbook, milk collection and member information excel sheets provided by Ngucici cooperative to have a better grasp on the current system.

## **3.4 Data Analysis Techniques**

### **3.4.1 Data Flow Diagrams**

This will be used to show the way information is supplied into the system and its output. It will also how the processed data will be stored. DFD's who the workflow of the users and their interaction with the system.

### **3.4.2 Use Case Diagram**

Use Case Diagrams will be used to ensure the correct system is developed by capturing the requirements from the user’s point of view.

## **3.5 Application Development Tools**

The following software and hardware tools and technologies will be used in the development of the application:

### **3.5.1 Software tools**

The software tools to be used include:

* Kubuntu 18.04
* Visual Studio IDE
* Cloud Firestore Database
* Dart Programming language
* Flutter Framework

The hardware tools to be used include:

* Laptop( 4 x Intel Core i7-3667u CPU @ 2.00GHZ
* Android smartphone u
* Flash Disk 16gb for back up

## **3.6 Application Testing tools**

Testing the application will be done by different users to check for the full functionality of the application. Each of the users will use the mobile application to check if it conforms with the expected output and for the user-friendliness of the mobile application.

The user will have an android smartphone running on different versions to the operating system to check cross-compatibility across different versions of android

## **3.7 Project Schedule**

|  |  |  |  |
| --- | --- | --- | --- |
| **ACTIVITY** | **DURATION** | **START DATE** | **END DATE** |
| Feasibility study | 2 weeks | 30/9/2019 | 14/10/2019 |
| Project Proposal | 4 weeks | 14/10/2019 | 11/11/2019 |
| Analysis | 4 weeks | 11/11/2019 | 2/12/2019 |
| Design | 3 weeks | 2/12/2019 | 23/2/2019 |
| Development | 8 weeks | 23/2/2019 | 17/2/2020 |
| Testing | 2 weeks | 17/2/2020 | 2/3/2020 |
| Implementation | 3 weeks | 2/3/2020 | 23/3/2020 |

## **3.8 Time Schedule**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ACTIVITY** |  |  |  |  |  |  |  |  | **TIME(weeks)** | | | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |  | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Concept Paper drafting |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Literature review |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Project planning |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Analysis of the system |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Proposal writing |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Software design |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Coding |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| System testing |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| System implementation |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Documentation |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

# **3.8 Project Budget**

|  |  |
| --- | --- |
| **PRODUCT** | **PRICE(KES)** |
| Laptop(Core i7 8gb ram) | 67000 |
| Android smartphone(Android 6.0.1 and above) | 15000 |
| Active Internet Connection (5 months) | 14500 |
| Google Play store account | 2500 |
| Transport | 5000 |
| **Total** | **104,000** |

# **CHAPTER 4: SYSTEM ANALYSIS AND REQUIREMENT MODELING**

## **4.1 Introduction**

This chapter provides a breakdown of the processes and functions involved in the current web-based system and highlights those of the proposed mobile client application. It describes the user requirements in terms of functional and non-functional requirements. The data, processes and involved entities are going to be broken down into detail. Also, the current system will be explained through graphical representation by the use of data flow diagrams coupled with use case diagrams. The essence of taking such measures is to ensure all functionalities of the system are included in the design to create a clear guideline for system development.

Finally, it helps in answering questions such as: what problem the system solves, who the intended user of the system is, where the system is supposed to be used and how the system is supposed to be used. This helps in coming up with a system/application which is well suited for business and customer needs.

The following will be involved in the system analysis process:

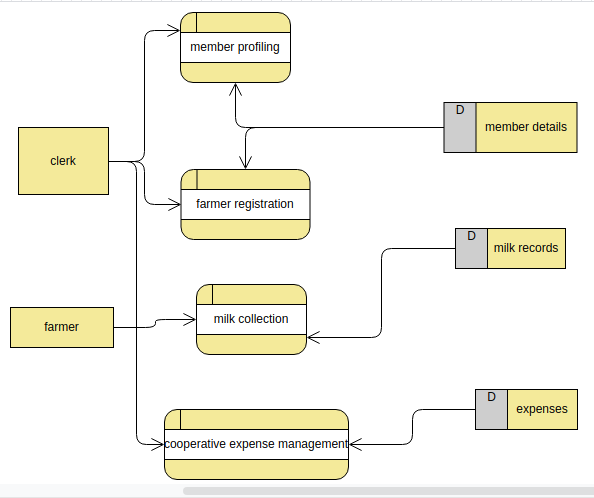
* Investigating the current environment - in this stage, different methods are involved in aiding the analyst in fully comprehending the system at the initial stage. This includes the combination observations and analysis of existing documents which help in construction the data models and defining the boundary lines of the system
* Business System Options - this involves developing a set of system options by using the previous system outputs by way of analysis.
* Requirement specifications - in this stage the functional requirements of the system and non-functional requirements are identified into detail. It will involve what the system can do and what it cannot do.
* Logical system specification - This stage involves creating the technical system options for the system. Each option is analyzed and weighed against the cost to enable the user to arrive at a solution
* Physical Data Design - On completion of the logical design, it is transformed into a data design which will run in the targeted environment

## **4.2 Description of the current system**

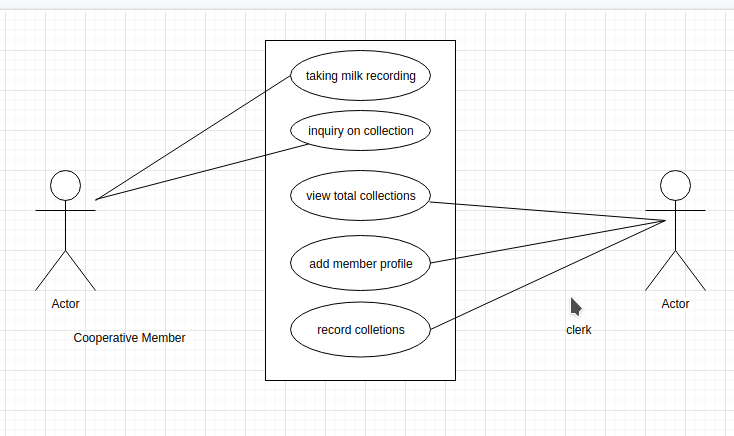
The current system used by the Ngucici milk cooperative society has both a manual and an automated approach. Milk collections within the cooperatives are taken in three phases that are: in the morning, afternoon and evening. Ngucici milk cooperative society has 12 collection centers distributed within the region. These centers are Rugirandu, America, Mbanda, Kiguru, Mijogene, Kathera, Gakiriri, Lecern, Kambi, Kabubungi, Red House and Maritati. A farmer takes his/her milk collection to the center after which the records are written down by the clerk representative. At the end of the day when all the collections have been made, the records are fed to an excel sheet that contains milk collection data for the month. The excel sheet contains a list of the 12 centers plus the attending clerk at the time of taking the records. Finally, it contains the amount collected(in liters) plus the selling price. Each member of the cooperative has a profile created as soon as he/she joins the cooperative. The profile contains the members' personal information. All this data is fed into the web portal by the clerk.

## **4.3 Current System Data flow Diagrams**

### **4.3.1Context diagram of the proposed system(Level 0 DFD)**



### **4.3.2 Use Case Modelling of Current System**

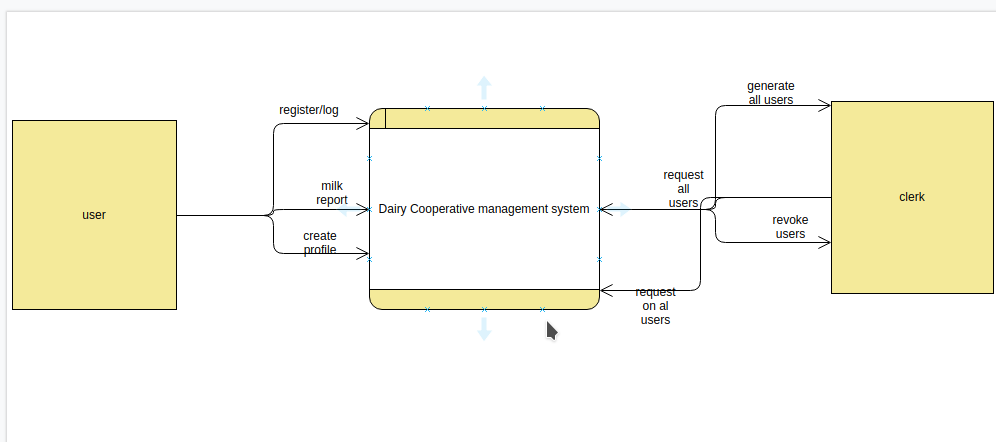


## **4.4 Description of the proposed system**

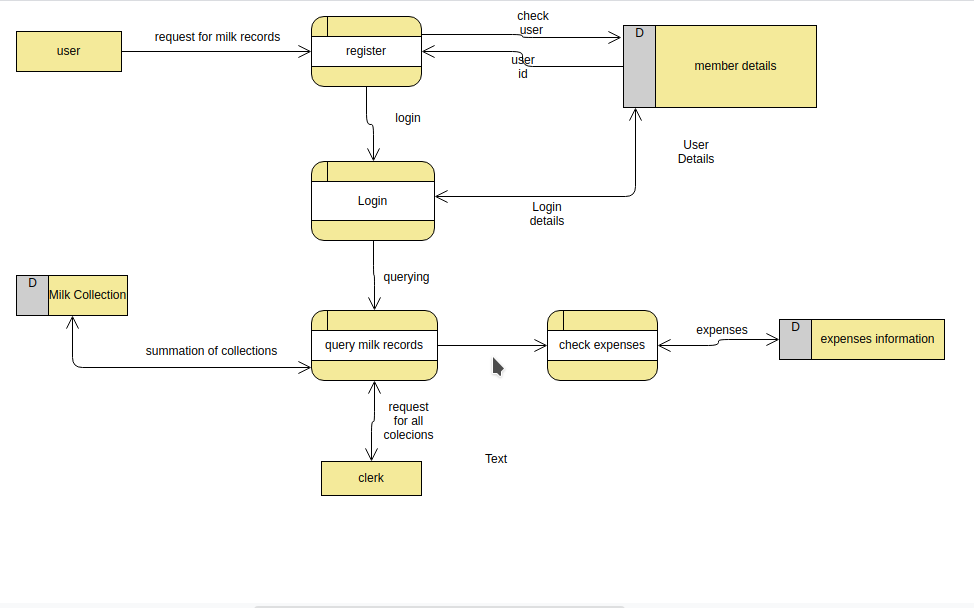
The proposed system is a mobile extension module built on flutter which is a cross-mobile framework that allows the development of mobile applications that are deployable in both android and iPhone Operating systems. The app can be downloaded from the google play store or apple store from which the user has access to his or her profile. The customer user has real-time access to his profile which can be updated at any given moment without the need to visit the cooperative. In addition, the user has a log of milk collections made, day and time in which they were made as well as the money spent on delivering the collections. Finally, the user has a personal expense manager which aids in keeping tabs on the expenses the member incurred over a given time span. The user can add or delete milk records in his/ her profile.

## 4**.5 Proposed System Data Flow Diagrams**

### **4.5.1 Context Diagram (Level 0 DFD)**



### **4.5.2 Context Diagram (Level 1 DFD)**



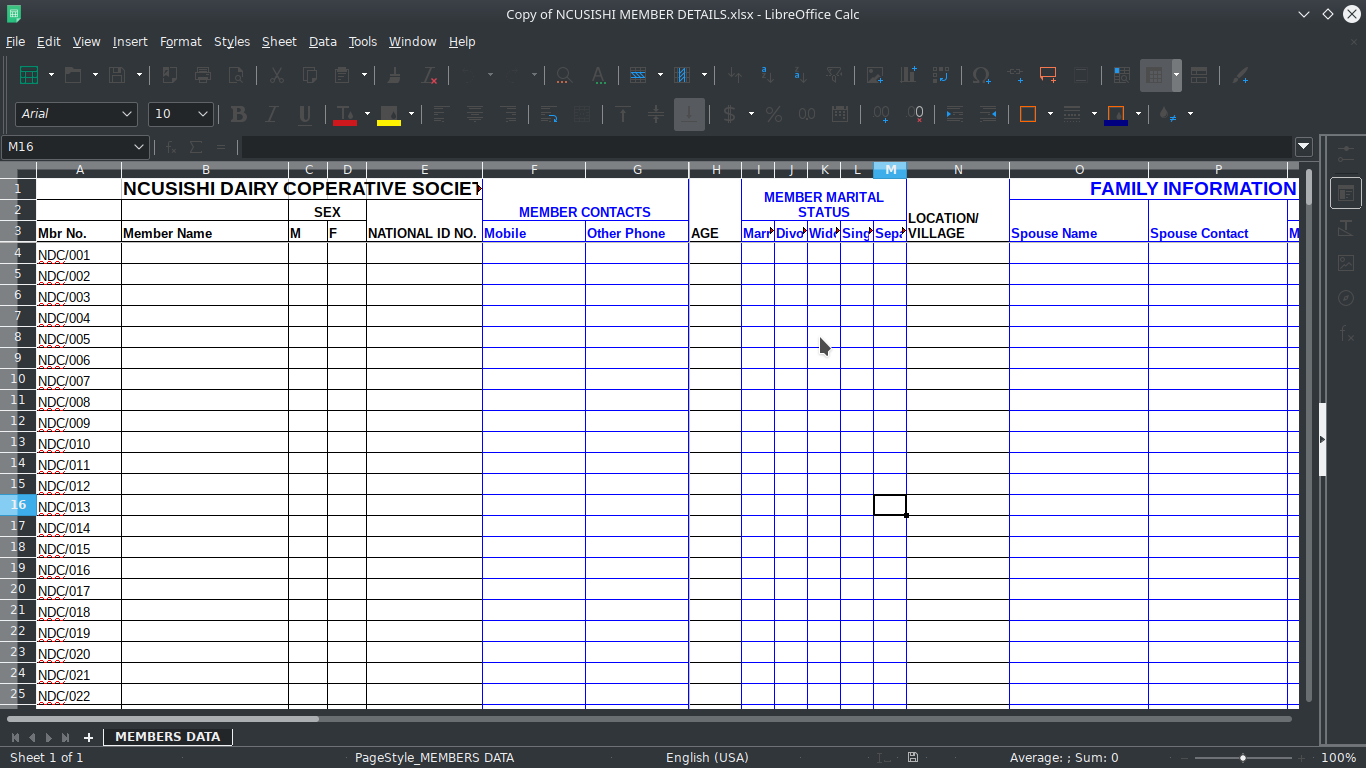
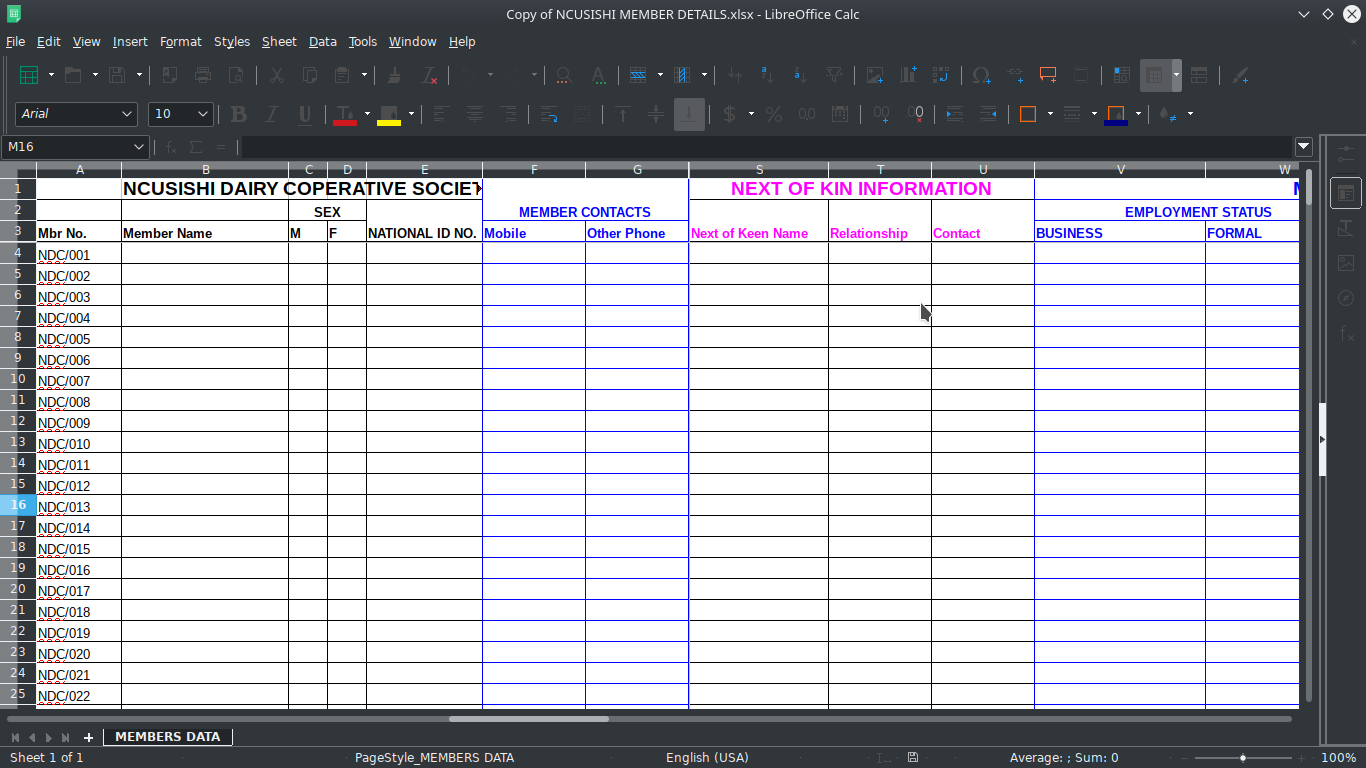
## **4.6 Methods of Data Collection**

In order to identify the nature of the current system, a combination of interviews and analysis on existing documents were carried out.

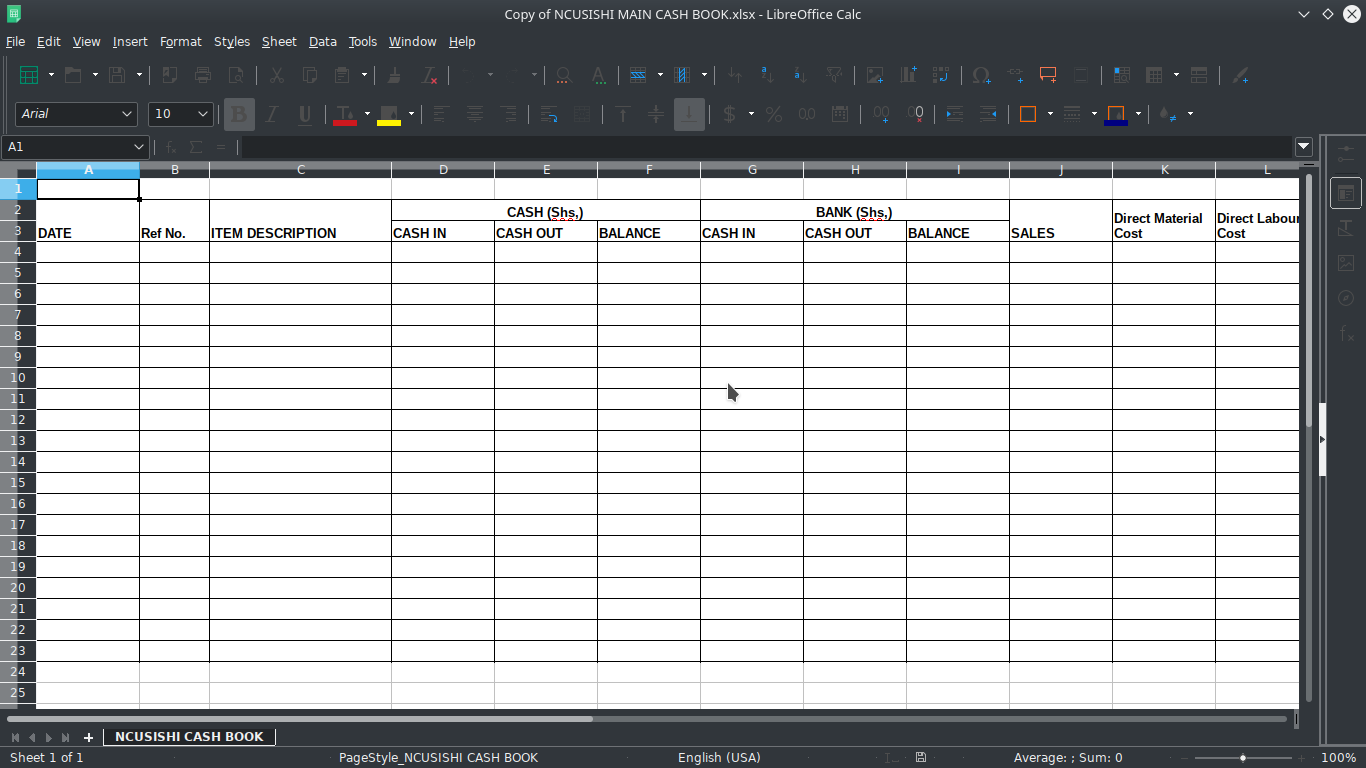
### **4.6.1 Analysis of existing documents**

In order to get a grasp on the design of the proposed system a couple of documents used by the Ngucici cooperative were studied. These documents are as follows

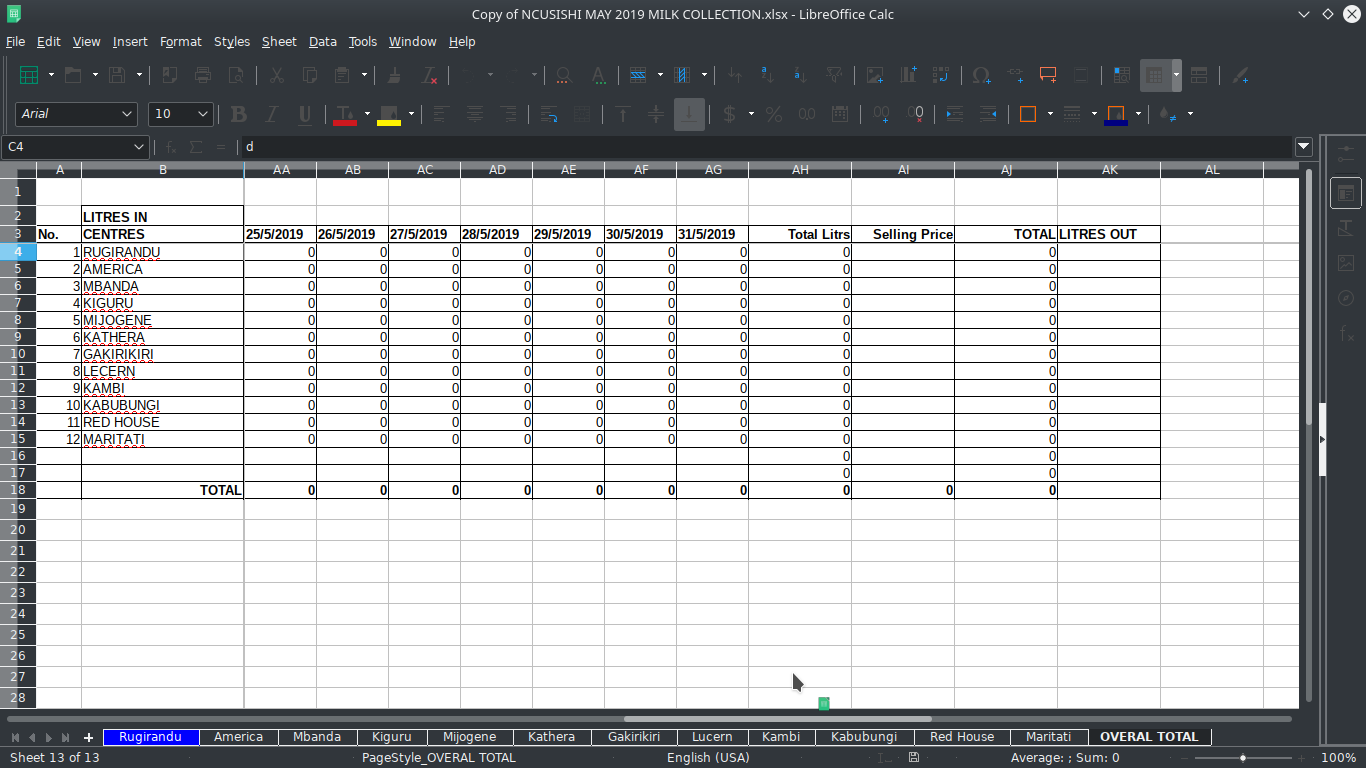
* Member Detail excels sheet - this document contains the template used in profiling the members of the cooperative society. It contains the information of all the members belonging to the cooperative society. The figure below shows this information.



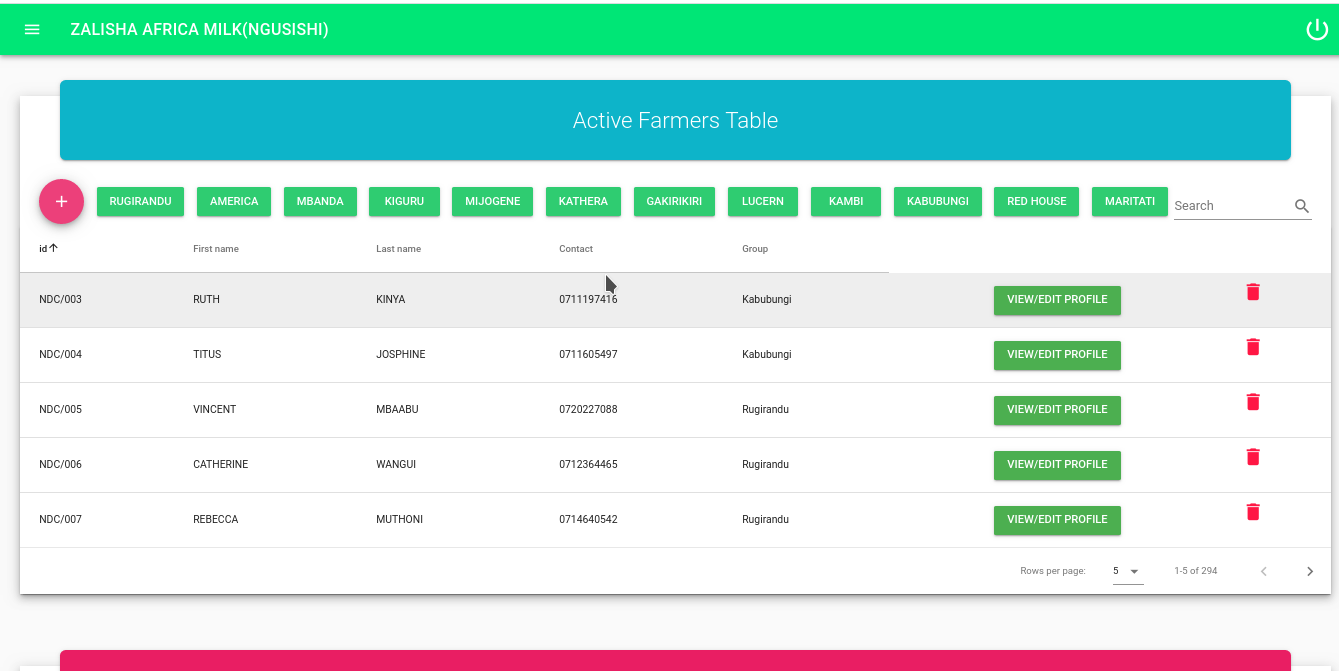
* Main CashBook excel sheet - this document contains the information pertaining to the expenses incurred by the example purchase of pioneer feeds. The figure below shows the main cash book excel sheet



* Milk Collection excels sheet - this document contains the information on the collections made to the cooperative society. It contains the amount of milk collected in liters in all the collection centers plus the selling price. The figure below highlights this information.



* Analysis of the website used by the cooperative admins on milk collection and expenses incurred. The figure below shows this information.



### **4.6.2 Interviews**

This method involves asking questions to the relevant stakeholders using the system. This method is useful as it aids in understanding the system from the user's perspective. Through baraza's held by the cooperative, the farmers were questioned on the convenience brought about by having a mobile application and the need to have a mobile application.

**Requirement Definition and Specification**

This refers to the activities which the use of the system should be able to do within the system and is subdivided into the following.

**Non-Functional Requirements**

* The system should be accommodative on various smartphone devices regardless of screen size or operating system.
* The system should have a friendly user interface
* The system should have a decent user experience
* The system should provide secure authentication of user accounts
* The system should be of reasonable size and light on memory.
* The system should be scalable
* The system should be able to accommodate additional models in the future.

**Functional Requirements**

* Users should be able to keep track of collections.
* Users should be able to update their profile
* Users should be able to manage their expenses
* Users should be able to add or remove records

# **CHAPTER 5: SYSTEM DESIGN**

## **5.1 Introduction**

System design involves the development of the system logically, and also establishes the inputs and outputs from the system, the processes involved to give the intended results. The main aim is to develop an architecture and precise structure of the proposed system. The main models are outlined that captures the key components of the system at different levels. The design of the system consists of the activities which yield system specifications that meet the functional requirements of the user. It involves the following

* Process Design
* Database Design
* User Interface Design

**Process Design**

In process design, the activities that users engage with while using the app are captured for example the stages involved in sorting the milk collections in the mobile application. This is concerned with how data moves through the system and how it is validated and transformed as it flows through the system.

**User Interface Design**

This mainly focuses on the interface used by the user and the interactions between the user and the interfaces. The main point of focus in user-interface design is the use of widgets in the system as well as the fluidity in transitions between widgets bundles up with the animations.

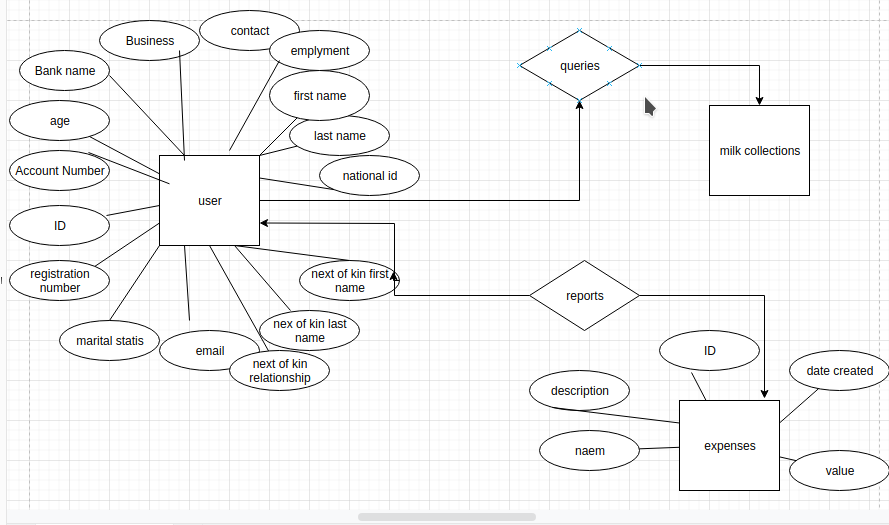
**Database design**

This is concerned with the representation of data and storage within the system. The system uses Firebase online storage which is a NoSQL based database system offered by Google. Rather than relational tables, Firebase uses a JSON-like schema to store data in collections. Collections may contain documents or other sub-collections. The database design will include identifying entities, attributes, and relationships of the database. Below is an entity-relationship diagram for the proposed system

## **5.2 Database entities**

The figure below higlights the existing entity relationship in the database.

**Entity Relationship of the database is as shown below**



### **5.2.1 Database Collections**

1. **User**

This collection contains information about the user. Attributes of the user include the following.

1. ID - Unique string which identifies each user
2. Account number - String that holds the account number for the user
3. Age - String which identifies the age of the user
4. Bank Name - String which holds the bank which the user belongs
5. Business - String which identifies the business affiliated with the user
6. Contact - String which holds the phone number for the user
7. Employment - String which holds the type of employment the user is under
8. First-Name - String which holds the first name of the user
9. Last Name - String which holds the last name of the user
10. Gender - String which holds the gender of the user
11. Group - String which holds the group the user belongs
12. National ID - String which holds the National ID number of the user
13. Kin-Contact - String which holds the contact of the next of kin
14. Kin First Name -String which holds the first name of the next of kin
15. Kin Last Name - String which holds the last name of the next of kin
16. Kin Relationship - String which holds the relationship between the user and the next of kin
17. Email - String which holds the email of the user
18. Marital Status - String which holds the marital status of the user
19. Reg/No - String which holds the registration number of the user to the cooperative

**2. Milk Collection**

This collection contains information on the milk records made and has the following attributes

1. Morning - This is a map object with 2 fields. The first field is a Document ID which is of type string and refers to a particular user. The second field is the date which is of type string.
2. Afternoon - This is a map object with 2 fields. The first field is a Document ID which is of type string and is nested to a particular user. The second field is the date which is of type string.
3. Evening - This is a map object with 2 fields. The first field is a Document ID which is of type string and is nested to a particular user. The second field is the date which is of type String

**3. Expenses**

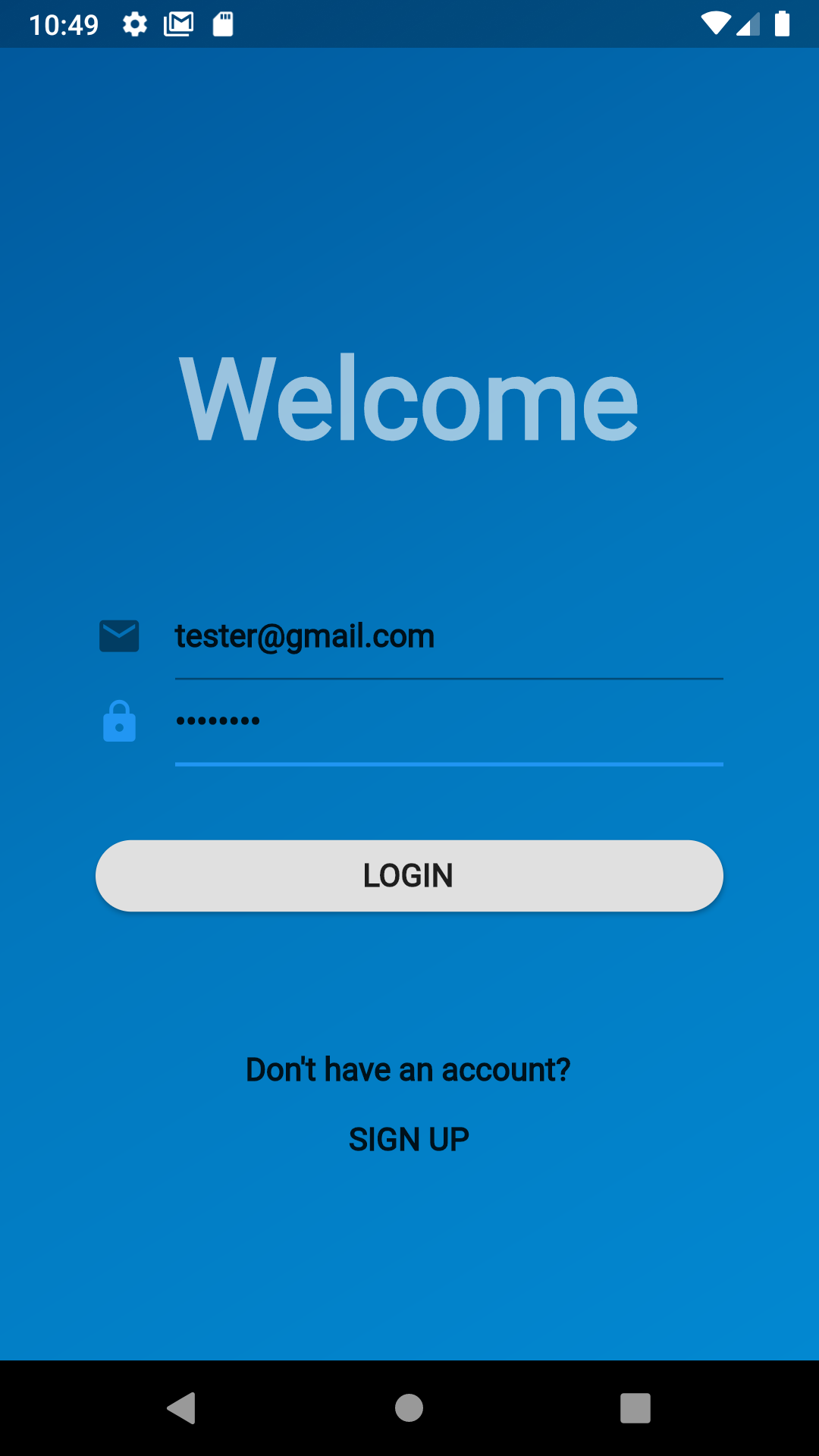
This collection contains information on the expenses incurred by the farmer and has the following attributes

1. ID - this is a unique string which holds the id for an expense
2. Description - this is a string which contains a brief description of the expense
3. Name - a string which contains the name of the expense
4. Created - a string which contains the date which the expense was created
5. Value - a string which contains the value of the expense

## **5.2 User Interface**

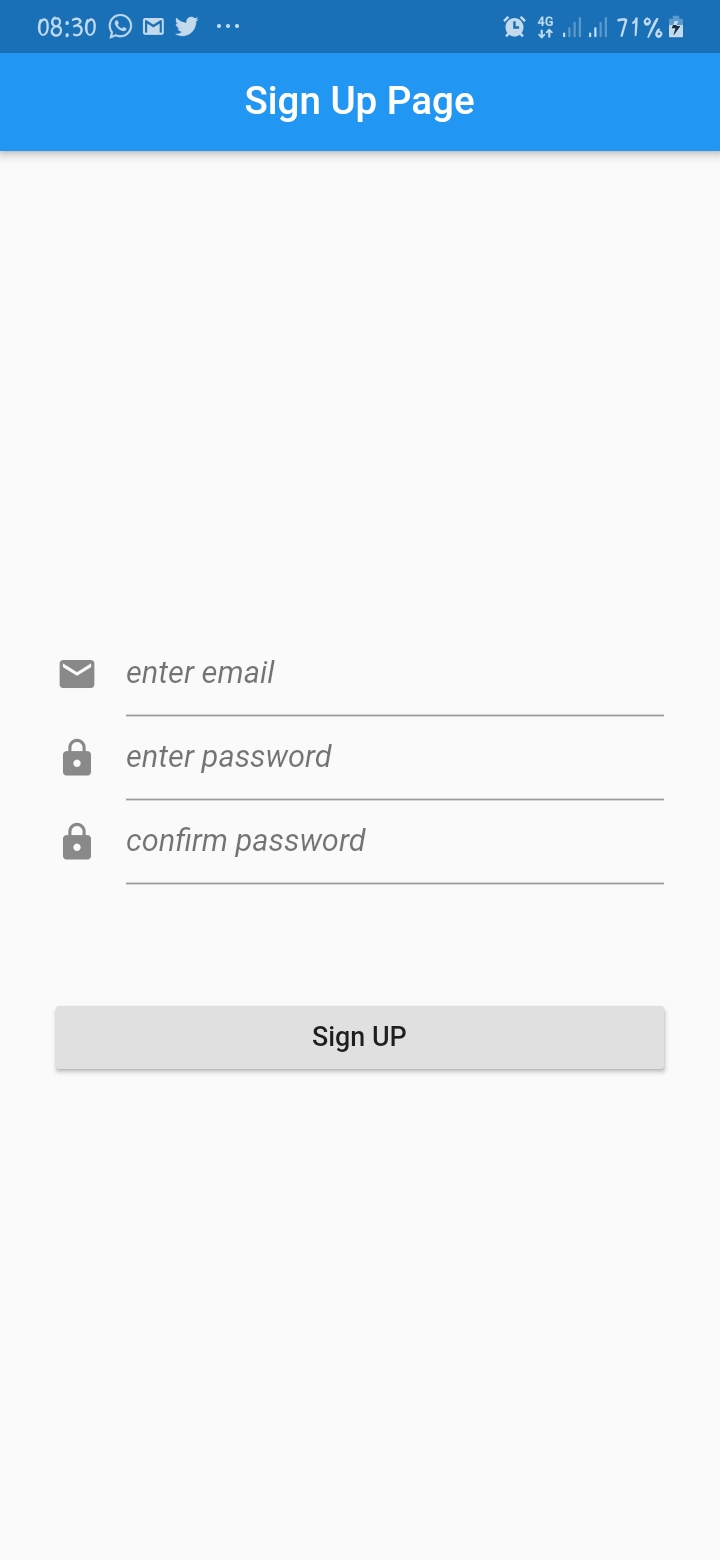
Below are sample UI interfaces for the system

1. **Login Screen**



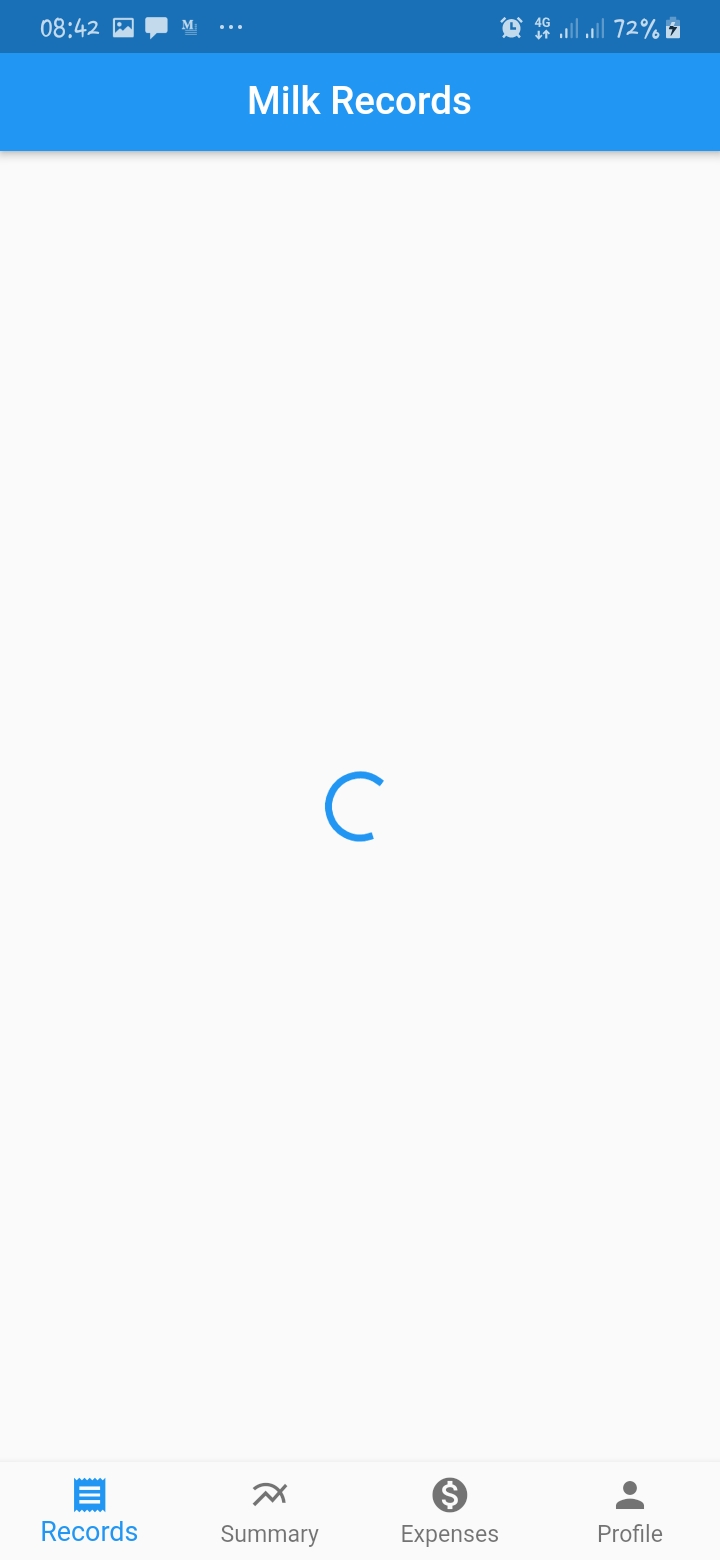
The figure above shows a login screen which is the landing page as you open the application. Users are required to log in using an email and password. In the event where a user is non-existent, the user has to create an account through signing

1. **Sign up page**



The figure above highlights the sign up page which the user navigates to in the event of creating a new user

1. Milk Records Screen



The figure above highlights the milk records landing page with a circular indicator progress bar at its center, this is the first page met by the user on successful authentication to the application. It will contain a list of all the records taken by the user.

# **CHAPTER 6: SYSTEM IMPLEMENTATION**

## **6.1 TOOLS USED FOR CODING AND IMPLEMENTATION**

### **6.1.1 Operating System**

1. Kubuntu 20.0.4 LTS (Bionic Beaver) - Used on the development machine
2. Android OS - Used on the testing and deployment devices

### **6.1.2 Development Platforms**

1. Android Studio IDE - IDE for building flutter applications which comes with and Android Emulator used to emulate a real working android device
2. Visual Studio Code - IDE for writing dart code
3. Firebase - A server less backend solution used to host the data.

### **6.1.3 Development Languages**

1. Dart - This is a client-optimized programming language developed by Google and is used to build mobile, desktop, server and web applications. It is an object-oriented, class-based, garbage-collected language with C-style syntax and can compile to either native code or JavaScript.
2. Flutter - framework built on Dart programming language used to build native applications which run on either Android Operating Systems or iOS

### **6.1.4 Hardware Tools**

1. HP Pavilion 15 laptop - used for analysis, design the application
2. Android smartphone - Used for testing the application

## **6.2 Testing**

The system was subjected to a variety of tests in each module, removing any bugs that were available. Testing was done to achieve the following

1. The system worked as designed
2. The system maintained data integrity
3. The system had no bugs
4. The user interface of the system of the system was interactive and highly responsive
5. The system input and output gave expected results

Some of the tests that were carried out in the system include:

### **6.2.1 Unit Test**

Here the system was divided into individual modules and sample data used to test each module. Any bugs that occurred were easily traced, isolated and fixed. Each of the modules create, read, edit , update and delete methods were tested using correct, incomplete and invalid data to ascertain the functionality.

### **6.2.3 Functional testing**

This test was carried out at the end of the development cycle. It involved verification of whether all the functional requirements of the system were met. The testing involved:

1. Input validation – Ensured that only required and predefined inputs were accepted by the system. Any invalid input was rejected.
2. Output validation – Correct outputs were produced by the system after a function has been initiated.
3. Functions – All the required functions of the system had been developed and were working as required.
4. Procedures – All the data flows in the system worked as required and must be invoked by the occurrence of certain events.

### **6.2.4 Performance testing**

Here, the system was tested to see how it would perform under different environments and under different constraints. This was to determine how the system is expected to cope under all possible environments of its use and that it functioned as required and no unexpected mishaps came up.

### **6.2.5 User acceptance test**

This test was carried out to ensure that the system worked as expected to the users and that they were able to use it efficiently.

## **6.3 System Maintenance**

The system will have to be periodically monitored as various issued may occur during its usage. A function was implemented that saved any bug that occurred and the details sent to the server.

The developer can then use these logs to identify the areas causing the issues and work on fixing them and sending an update to the users.

# **CHAPTER 7: LIMITATIONS, RECOMMENDATIONS AND CONCLUSION**

### **7.1 Challenges**

Below are some of the challenges that were faced in the development of the system:

1. It was costly to move around and collect data on the existing system.
2. Slow unreliable internet when sending data between the system and the server

## **7.2 Limitations**

Below are some of the limitations of the current system

1. The backend used (Firebase) is maintained by a third party thus any issue that may occur to the server is out of the control of the developer.
2. The system does not provide a payment avenue to users
3. The system can only be used on smartphones and by users fluent in English.

## **7.3 Recommendations**

Possible solutions to the above mentioned limitations include:

1. Migrating the backend to self-maintained server with which the developer has full control
2. Integrate the system with available payment gateways.
3. Add support for multiple languages
4. Add functionality to filter records by a given duration

## **7.4 Conclusion**

In conclusion, this study shows the adoption of a mobile client for the Ngucici dairy cooperative goes a long way in helping both the farmers and the cooperative. With continued use of the platform such that it can scale and gain more traction the data collected from the farmer's contribution to the cooperative can be used as a method to credit score the farmers. The implications of this mean one can approach a financial institution like a bank with this data from where mobile loans can be disbursed through the platform to the farmers.

# **REFERENCES**

Ganesh. (2019, 1 8). *13 Reasons why you should consider movign to Flutter*. Retrieved from Medium : https://medium.com/flutter-community/13-reasons-why-you-should-choose-consider-to-move-to-flutter-in-2019-24323ee259c1

Grandt, E. (2019, 5 20). *Why I think Flutter is the future of mobile app development*. Retrieved from FreeCodeCamp: https://www.freecodecamp.org/news/why-i-think-flutter-is-the-future-of-mobile-app-development-768332b73c0d/

J, G. (n.d.). *13 Reasons why you should consider moving to flutter*. Retrieved from https://medium.com/flutter-community/13-reasons-why-you-should-choose-consider-to-move-to-flutter-in-2019-24323ee259c1

Mroczkowska, A. (2019, 9 10). *Flutter in Mobile App Development - Pros & Cons for App owners*. Retrieved from Droids on Roids: https://www.thedroidsonroids.com/blog/flutter-in-mobile-app-development-pros-and-cons-for-app-owners

P.S, T., & Millar, R. (2002). Journal of Information and Software Technology. *Software development process - an assessment*, 44-46.

Rouse, M. (2016). *Mobile Application Development*. Retrieved from SearchAppArchitecture: https://searchapparchitecture.techtarget.com/definition/mobile-application-development

# **APPENDIX A: USER MANUAL**

**Requirements for installation**

* Android smartphone
* Android OS 4.1.1 (KitKat)
* Google Play Services installed

**Installation Procedure**

* Copy the mdairy.apk file from the CD to your smartphone
* On your device:
  1. Open Settings app
  2. Tap on settings
  3. Check „Allow installation from unknown sources'
* Open your file manager application and navigate to where you copied the mdairy.apk file
* Tap on it and follow the installation procedure.
* After installation is complete, tap on open to launch the application

You can then login/ register in order

# **APPENDIX B: IMPORTANT CODE**

Below are some code snippets of importance.

**Code for plugins and dependencies used:**

The plugins are stored in a yaml file as shown below

name: m\_dairy

description: A new Flutter project.

version: 1.0.0+1

environment:

sdk: ">=2.1.0 <3.0.0"

dependencies:

flutter:

sdk: flutter

cupertino\_icons: ^0.1.2

cloud\_firestore: ^0.13.0+1

firebase\_auth: ^0.15.4

rxdart: ^0.23.1

email\_validator: '^1.0.0'

flutter\_cupertino\_date\_picker: ^1.0.12

intl: ^0.16.1

floating\_search\_bar: ^0.3.0

simple\_search\_bar: ^0.1.7

bezier\_chart: "^1.0.16"

date\_range\_picker: ^1.0.5

pie\_chart: ^3.1.1

validators: ^2.0.0+1

shared\_preferences: ^0.5.6+3

flutter\_icons:

android: "launcher\_icon"

ios: true

image\_path: "assets/launchericon.png"

dev\_dependencies:

flutter\_test:

sdk: flutter

flutter\_launcher\_icons: "^0.7.3"

flutter:

uses-material-design: true

assets:

- assets/signupscreen.png

fonts:

# - family: Schyler

# fonts:

# - asset: fonts/Schyler-Regular.ttf

# - asset: fonts/Schyler-Italic.ttf

# style: italic

# - family: Trajan Pro

# fonts:

# - asset: fonts/TrajanPro.ttf

# - asset: fonts/TrajanPro\_Bold.ttf

# weight: 700

#

**Code for authorization and fetching data**

import 'dart:async';

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/services.dart';

import 'package:m\_dairy/src/models/expenses.dart';

import 'package:m\_dairy/src/models/milkrecord.dart';

import 'package:m\_dairy/src/models/users.dart';

import 'package:m\_dairy/src/resources/storedvalues.dart';

class FireBaseAuthProvider {

final \_auth = FirebaseAuth.instance;

final \_firestore = Firestore.instance;

final \_storedValues = StoredValues();

Future signOut() async {

await \_storedValues.removeValues();

await \_auth.signOut();

}

Future<User> userInfoPrefs() async {

var email = await \_storedValues.getFirebaseUserEmail();

return await \_firestore

.collection('users')

.document(email)

.get()

.then((documentSnapshot) => User.fromDocuments(documentSnapshot));

}

Future<User> userInfo(FirebaseUser firebaseUser) async {

return await \_firestore

.collection('users')

.document(firebaseUser.email)

.get()

.then((ds) => User.fromDocuments(ds));

}

Stream<User> userProfile(FirebaseUser firebaseUser) {

return \_firestore

.collection('users')

.document(firebaseUser.email)

.snapshots()

.map((i) => User.fromDocuments(i));

}

Stream<List<MilkRecord>> allMilkRecords(FirebaseUser user) {

return \_firestore

.collection('users')

.document(user.email)

.collection('MilkRecords')

.orderBy('date', descending: true)

.snapshots()

.map(

(i) => i.documents.map((i) => MilkRecord.fromDocument(i)).toList());

}

Stream<List<MilkRecord>> sortedByDate(String date, FirebaseUser user) {

return \_firestore

.collection('users')

.document(user.email)

.collection('MilkRecords')

.where('date', isEqualTo: date)

.snapshots()

.map((snapshot) =>

snapshot.documents.map((i) => MilkRecord.fromDocument(i)).toList());

}

Future<FirebaseUser> signedInUser() async {

return await \_auth.currentUser();

}

void deleteDocs(String id, String date, String collectionPeriod,

String litresCollected) async {

var qs = await \_firestore

.collection('users')

.document(id)

.collection('MilkRecords')

.where('date', isEqualTo: date)

.where('collectionPeriod', isEqualTo: collectionPeriod)

.where('litresCollected', isEqualTo: litresCollected)

.getDocuments();

String docId = qs.documents.map((i) => i.documentID.toString()).toString();

print(docId);

\_firestore

.collection('users')

.document(id)

.collection('MilkRecords')

.document(docId)

.delete();

}

Future<double> totalCollections() async {

var qs = await \_firestore

.collection('users')

.document('date@gmail.com')

.collection('MilkRecords')

.getDocuments();

double t = qs.documents

.map((i) => MilkRecord.fromDocument(i))

.map((i) => double.parse(i.litresCollected))

.toList()

.reduce((a, b) => a + b);

print(t);

}

Future<List<String>> sortedDates(String firstDate, String lastDate) async {

var qs = await \_firestore

.collection('users')

.document('date@gmail.com')

.collection('MilkRecords')

.orderBy('date')

.where('date', isGreaterThanOrEqualTo: firstDate)

.where('date', isLessThanOrEqualTo: lastDate)

.getDocuments();

List<String> dates = qs.documents

.map((i) => MilkRecord.fromDocument(i))

.map((i) => i.date)

.toSet()

.toList();

dates.forEach((i) => print(i));

return dates;

}

Future<List<String>> alldates() async {

var qs = await \_firestore

.collection('users')

.document('date@gmail.com')

.collection('MilkRecords')

.orderBy('date')

.getDocuments();

List<String> alldates = qs.documents

.map((i) => MilkRecord.fromDocument(i))

.map((i) => i.date)

.toSet()

.toList();

alldates.forEach((i) => print(i));

return alldates;

}

Future<List<double>> collectionsforalldays() async {

List<String> dates = await alldates();

List<double> collperday = [];

for (int i = 0; i < dates.length; i++) {

var qs = await \_firestore

.collection('users')

.document('date@gmail.com')

.collection('MilkRecords')

.where('date', isEqualTo: dates[i])

.getDocuments();

double coll = qs.documents

.map((i) => MilkRecord.fromDocument(i))

.map((i) => double.parse(i.litresCollected))

.toList()

.reduce((a, b) => a + b);

collperday.add(coll);

}

collperday.forEach((i) => print(i));

return collperday;

}

Future<List<double>> collectionForSortedDates(

String firstDate, String lastDate) async {

List<double> collperday = [];

List<String> sortes = await sortedDates(firstDate, lastDate);

for (int i = 0; i < sortes.length; i++) {

var qs = await \_firestore

.collection('users')

.document('date@gmail.com')

.collection('MilkRecords')

.where('date', isEqualTo: sortes[i])

.getDocuments();

double coll = qs.documents

.map((i) => MilkRecord.fromDocument(i))

.map((i) => double.parse(i.litresCollected))

.toList()

.reduce((a, b) => a + b);

collperday.add(coll);

}

return collperday;

}

Future<List<String>> fromFdtoLd(String fd, String ld) async {

String email = await \_storedValues.getFirebaseUserEmail();

var qs = await \_firestore

.collection('users')

.document(email)

.collection('MilkRecords')

.orderBy('date')

.where('date', isLessThanOrEqualTo: ld)

.where('date', isGreaterThanOrEqualTo: fd)

.getDocuments();

return qs.documents

.map((i) => MilkRecord.fromDocument(i))

.map((i) => i.date)

.toSet()

.toList();

}

Future<List<double>> collsperday(String fd, String ld) async {

String email = await \_storedValues.getFirebaseUserEmail();

List<String> dates = await fromFdtoLd(fd, ld);

List<double> colls = [];

for (int i = 0; i < dates.length; i++) {

var qs = await \_firestore

.collection('users')

.document(email)

.collection('MilkRecords')

.where('date', isEqualTo: dates[i])

.getDocuments();

double l = qs.documents

.map((i) => MilkRecord.fromDocument(i))

.map((i) => double.parse(i.litresCollected))

.toList()

.fold(0, (a, b) => a + b);

colls.add(l);

}

return colls;

}

Future<String> totalEarnings(String fd, String ld) async {

// String email = await \_storedValues.getFirebaseUserEmail();

// List<String> dates = await fromFdtoLd(

// fd,

// ld,

// );

// List<double> toalprices = [];

// for (int i = 0; i < dates.length; i++) {

// var qs = await \_firestore

// .collection('users')

// .document(email)

// .collection('MilkRecords')

// .where('date', isEqualTo: dates[i])

// .getDocuments();

// double l = qs.documents

// .map((i) => MilkRecord.fromDocument(i))

// .map((i) => double.parse(i.totalPricing))

// .toList()

// .reduce((a, b) => a + b);

// toalprices.add(l);

// }

// return toalprices.reduce((a, b) => a + b).toString();

String email = await \_storedValues.getFirebaseUserEmail();

QuerySnapshot qs = await \_firestore

.collection('users')

.document(email)

.collection('MilkRecords')

.where('date', isLessThanOrEqualTo: ld)

.where('date', isGreaterThanOrEqualTo: fd)

.getDocuments();

return qs.documents

.map((i) => MilkRecord.fromDocument(i))

.map((i) => double.parse(i.totalPricing))

.toList()

.fold(0, (a, b) => a + b)

.toString();

}

uploadMilkRecord(String date, String collectionPeriod, String litresCollected,

String pricePerLitre) async {

String totalPricing =

(double.parse(litresCollected) \* double.parse(pricePerLitre))

.toString();

var user = await signedInUser();

\_firestore

.collection('users')

.document(user.email)

.collection('MilkRecords')

.document()

.setData({

'date': date,

'collectionPeriod': collectionPeriod,

'litresCollected': litresCollected,

'pricePerLitre': pricePerLitre,

'totalPricing': totalPricing,

}, merge: true);

}

deleteMilkRecord(

String date, String collectionPeriod, String litresCollected) async {

var user = await signedInUser();

var qs = await \_firestore

.collection('users')

.document(user.email)

.collection('MilkRecords')

.where('date', isEqualTo: date)

.where('collectionPeriod', isEqualTo: collectionPeriod)

.where('litresCollected', isEqualTo: litresCollected)

.getDocuments();

List<String> docIDs = qs.documents.map((i) => i.documentID).toList();

docIDs.forEach((i) => \_firestore

.collection('users')

.document(user.email)

.collection('MilkRecords')

.document(i)

.delete());

}

uploadExpense(

String expenseType, String description, String value, String date) async {

var user = await signedInUser();

\_firestore

.collection('users')

.document(user.email)

.collection('Expenses')

.document()

.setData({

'expenseType': expenseType,

'description': description,

'value': value,

'createdAt': date,

}, merge: true);

}

Stream<List<Expenses>> allexpenses(FirebaseUser user) {

return \_firestore

.collection('users')

.document(user.email)

.collection('Expenses')

.snapshots()

.map((i) =>

i.documents.map((i) => Expenses.fromDocumentSnapshot(i)).toList());

}

Future deleteExpense(

String expenseType,

String description,

String createdAt,

String value,

FirebaseUser firebaseUser,

) async {

var qs = await \_firestore

.collection('users')

.document(firebaseUser.email)

.collection('Expenses')

.where('expenseType', isEqualTo: expenseType)

.where('description', isEqualTo: description)

.where('createdAt', isEqualTo: createdAt)

.where('value', isEqualTo: value)

.getDocuments();

List<String> docIds = qs.documents.map((i) => i.documentID).toList();

docIds.forEach((i) => \_firestore

.collection('users')

.document(firebaseUser.email)

.collection('Expenses')

.document(i)

.delete());

}

Future<double> feedsValue() async {

var user = await signedInUser();

var qs = await \_firestore

.collection('users')

.document(user.email)

.collection('Expenses')

.where('expenseType', isEqualTo: 'Feeds')

.getDocuments();

double q = qs.documents

.map((i) => Expenses.fromDocumentSnapshot(i))

.map((i) => double.parse(i.value))

.toList()

.fold(0, (a, b) => a + b);

return q;

}

Future<double> billsValue() async {

String email = await \_storedValues.getFirebaseUserEmail();

var qs = await \_firestore

.collection('users')

.document(email)

.collection('Expenses')

.where('expenseType', isEqualTo: 'Bills')

.getDocuments();

double q = qs.documents

.map((i) => Expenses.fromDocumentSnapshot(i))

.map((i) => double.parse(i.value))

.toList()

.fold(0, (a, b) => a + b);

return q;

}

Future<double> equipmentValue() async {

String email = await \_storedValues.getFirebaseUserEmail();

var qs = await \_firestore

.collection('users')

.document(email)

.collection('Expenses')

.where('expenseType', isEqualTo: 'Equipment')

.getDocuments();

double q = qs.documents

.map((i) => Expenses.fromDocumentSnapshot(i))

.map((i) => double.parse(i.value))

.toList()

.fold(0, (a, b) => a + b);

return q;

}

Future<double> houseHold() async {

var user = await signedInUser();

var qs = await \_firestore

.collection('users')

.document(user.email)

.collection('Expenses')

.where('expenseType', isEqualTo: 'Household')

.getDocuments();

double q = qs.documents

.map((i) => Expenses.fromDocumentSnapshot(i))

.map((i) => double.parse(i.value))

.toList()

.fold(0, (a, b) => a + b);

return q;

}

Future<double> othersValue() async {

String email = await \_storedValues.getFirebaseUserEmail();

var qs = await \_firestore

.collection('users')

.document(email)

.collection('Expenses')

.where('expenseType', isEqualTo: 'Other')

.getDocuments();

double q = qs.documents

.map((i) => Expenses.fromDocumentSnapshot(i))

.map((i) => double.parse(i.value))

.toList()

.fold(0, (a, b) => a + b);

return q;

}

Future<String> signIn(String email, String password) async {

try {

var authResult = await \_auth.signInWithEmailAndPassword(

email: email, password: password);

return authResult.user.email.toLowerCase();

} on PlatformException catch (e) {

return e.code.toUpperCase();

}

}

Future<String> signUp(String email, String password, String fname,

String lname, String gender, String age, String phoneNumber) async {

try {

var authResult = await \_auth.createUserWithEmailAndPassword(

email: email, password: password);

await \_firestore

.collection('users')

.document(authResult.user.email)

.setData({

'id': authResult.user.uid,

'fname': fname,

'lname': lname,

'gender': gender,

'age': age,

'email': authResult.user.email,

'phoneNumber': phoneNumber,

}, merge: true);

return authResult.user.email;

} on PlatformException catch (e) {

return e.code.toUpperCase();

}

}

}