

# Declaration:

Student Number: Niamh Mc Nulty - C16305423

Supervisor’s Name: Don Ryan

Second Reader: Paddy Matthews

Program (degree): DT354 Business Computing

Year: DT354/4

Module: Final Year Project

Submission Date: 30/04/2020

Title of Assignment: File A Farm - Final Year Project Documentation

GitHub Repository: <https://github.com/niamhmc99/Farm>

Declaration of Ownership: I declare that the attached work is entirely my own and that all sources have been acknowledged: X

Signature: Niamh Mc Nulty

# Acknowledgements

I would like to take this opportunity to thank everyone who has helped along my journey in this project and throughout the college years. Firstly, I would like to thank my family who have kept my spirits up on a regular basis. I really couldn’t have done it without their support, drive and kindness. It has been a challenging new experience for me and I’m both surprised and proud of how far I’ve come.

I would like to thank my supervisor, Don Ryan, for his guidance, patience and support throughout my final year project. I would also like to thank Paddy Matthews, my second reader, for his support and direction on my project. Their interest in agriculture proved beneficial throughout the development of my app.

Overall, I would like to say a huge thanks to all my lecturers over the past 4 years of study. I never imagined being able to get through first year never mind reaching the last hurdle of final year and for that I am forever grateful for all the effort the lecturers have put into our course. Without their guidance and persistent help this work would have never been possible.

I would like to thank Janet Copeland from Workday, who hired me for my internship and pushed me out of my comfort zone. I learnt so much from the company and I’ve gained confidence and knowledge of the working environment. I’m excited to be joining them as a graduate once I successfully finish my final year.

Lastly, I would like to acknowledge my classmates who have supported me throughout the years. There are too many to name, but all of them were willing to help when needed. A special thanks to my classmate, Gillian O’Connor who has been a great friend and joy to sit beside the last four years, I couldn’t have done it without her.

Table of Contents

[Declaration: 1](#_Toc39165997)

[Acknowledgements 2](#_Toc39165998)

[Introduction 4](#_Toc39165999)

[Project Introduction 4](#_Toc39166000)

[Project Overview 5](#_Toc39166001)

[Business Case 5](#_Toc39166002)

[Business Rules 7](#_Toc39166003)

[Requirements 7](#_Toc39166004)

[Business Actor 7](#_Toc39166005)

[Services 7](#_Toc39166006)

[Animals 7](#_Toc39166007)

[Weather 7](#_Toc39166008)

[Vets 8](#_Toc39166009)

[To Do List 8](#_Toc39166010)

[Nearby Places 8](#_Toc39166011)

[Emissions 8](#_Toc39166012)

[Invoices/ Receipts 8](#_Toc39166013)

[Background Processes 8](#_Toc39166014)

[Functional Requirements 9](#_Toc39166015)

[User Use Case Diagram 9](#_Toc39166016)

[Requirements 10](#_Toc39166017)

[Technology overview: 12](#_Toc39166018)

[Android Studio: 12](#_Toc39166019)

[Database 12](#_Toc39166020)

[Cloud Firestore 12](#_Toc39166021)

[Other Technologies 13](#_Toc39166022)

[Firebase Cloud Messaging 13](#_Toc39166023)

[Cloud Storage and Cloud Functions 13](#_Toc39166024)

[MP Android Chart 14](#_Toc39166025)

[Open Weather API 14](#_Toc39166026)

[Google Maps API 14](#_Toc39166027)

[Google Places API 15](#_Toc39166028)

[Glide 15](#_Toc39166029)

[Java 16](#_Toc39166030)

[JSON 16](#_Toc39166031)

[Version Control 16](#_Toc39166032)

[GitHub 16](#_Toc39166033)

[System Architecture 17](#_Toc39166034)

[Database Model 18](#_Toc39166035)

[Implementation – UI Design and Functionality Reference Guide 19](#_Toc39166036)

[Home Screen – Main Activity: 21](#_Toc39166037)

[Side Navigation – Main Activity: 22](#_Toc39166038)

[Animals – Animal Activity: 22](#_Toc39166039)

[Insert Animal – Insert Animal Activity: 23](#_Toc39166040)

[Calving Expectation Alarm Date Implementation: 24](#_Toc39166041)

[Scan Barcode Implementation: 25](#_Toc39166042)

[Weather – Weather Activity: 27](#_Toc39166045)

[Nearby Places using Current Location: 28](#_Toc39166046)

[To Do List - To Do List Activity: 29](#_Toc39166047)

[Vet Appointments - Vet Activity: 29](#_Toc39166048)

[Farm Emissions – Emissions Activity: 30](#_Toc39166049)

[Calculate Beef Emissions Activity: 31](#_Toc39166050)

[Average Dairy Emissions Results 32](#_Toc39166051)

[Calculate Dairy Emissions Activity: 33](#_Toc39166052)

[Average Dairy Emissions Results 33](#_Toc39166053)

[Recommendations - Ways to Reduce Emissions Activity: 34](#_Toc39166054)

[Reduce Beef Emissions 34](#_Toc39166055)

[Reduce Dairy Emissions 34](#_Toc39166056)

[**External Links from the Way to Reduce Emissions Activity** 35](#_Toc39166057)

[Invoice/ Receipts – Invoice/ Receipts Activity: 35](#_Toc39166058)

[Test Plan 36](#_Toc39166059)

[Test Case 36](#_Toc39166060)

[Test Scenario 36](#_Toc39166061)

[Expected Result 36](#_Toc39166062)

[Test Result: Pass/Fail 36](#_Toc39166063)

[Implementation Issues 43](#_Toc39166064)

[Conclusion 44](#_Toc39166065)

[References 45](#_Toc39166066)

[Footnotes 46](#_Toc39166067)

# Introduction

## Project Introduction

The main objective of this project is to provide a platform for farmers to easily track and record every day scenarios on the farm. The idea of this project stems from being reared on a farm and listening to farmers who visit my home on a daily basis. My aim is to provide a simplistic yet effective way for farmers to achieve this.

## Project Overview

It provides a number of useful tools for the user. The main purpose of the app is to create, update and retrieve animal records. This is extended to record important events such as veterinary visits and critical details such as calving date reminders. One of the main objectives of this app is to make it as convenient to use as possible.

A barcode scanner is incorporated allowing the farmer to scan the animal's passport also known as the “Blue Card” or the farmer can scan the animals tag itself for quick retrieval of their record. There is an area for the farmer to record expenses and allow them to take photos of bills/receipts.

I added a safety element where a user can send their location and a pre typed message to a connected user e.g. if a user is alone in a field and needs help you can send your location to a connected user by a press of a button.

Other useful elements of the app include a weather page with real time up to date data which includes humidity, wind speed, temperature, sunrise and sunset. Every farmer has a busy schedule and I’ve created a feature to handle this, where farmers can record their daily tasks in the form of a to-do list. I thought a simple map feature would be useful for farmers to search for local vets, mechanics and other necessities so this is also included.

I’ve also developed a farm emissions calculator, to track and analyse the emissions on a farm which then recommends ways to reduce Greenhouse Gas (GHG) emissions depending on the farmers’ cattle herd type. I've also provided links to Teagasc’s external website for further information as they are the leading experts on Agriculture Emissions in Ireland. This will involve entering data about your farm, and through the calculator we convert this into a carbon footprint.

## Business Case

When the tractor is a farmers’ main office, keeping an accurate record of day to day events of the farm can be extremely difficult. Farming records have traditionally been paper-based. While paper records can suffice, this is highly impractical for farmers working long hours’ outdoors. By the time a farmer is finished work for the day important events or details may be forgotten. A laptop is not practical to carry around which is why a mobile app solution is a perfect fit for the farmer and their pocket. I wanted to create an app where farmers can manage their animal records from anywhere whether it be in the mart or in a field. Managing records can be tedious so by simplifying the process through utilising the phones camera to allow scanning of animal tags and storing receipts, prevents and reduces human error.

With the alarming rise of farm accidents, having a safety element within the app could prevent another person becoming another statistic. From living on a farm my whole life, it has opened up my eyes on how easy it is for an accident to occur.

Counting farm emissions has become a necessity as agriculture emissions results as one of the largest contributors to Irelands Greenhouse Gas (GHG) emissions. However, incorporating a farm emissions calculator into the app gives farmers recommendations of ways to reduce GHG emissions on their farm and allows them to start making a difference for the future.

## Business Rules

* User’s must register their account in order to use the app.
* User’s must be logged into the app to use the services it offers.
* Each user can only have one account associated to their email address and herd number. The email address must have a valid email format or else an error will be displayed. Also, the password must compose of seven characters.

## Requirements

### Business Actor

The User is the only type of business actor in my application.

#### Users

Once the app is downloaded, anyone who registers is automatically stored into the database and given the role of a user. The person needs to enter certain fields to create an account; email, password and herd number. Each user can carry out the same activities. The user’s email address is unique to each account which acts as the username for their profile. Upon successful registration, the user’s email and password are used for their login credentials.

Once the user has logged in successfully, they are brought to the main home screen where there are 7 visual buttons to choose from. If they slide left, there is a navigation drawer where the logged in user’s email address appears on top of the menu navigation options.

## Services

### Animals

This is where the user can view their animals, insert a new animal into their herd and use the scan barcode option for quick retrieval of the animal's details. This is where the users list of animals is displayed. There is an add button which directs the user to the Insert Animal page. If the inserted animal is female, the user is given the option to enter calving details related to the insemination date. The user will be sent a notification 7 days prior to the expected due date.

### Weather

The user can get a visual of the weather conditions in their current location. This offers information such as the exact temperature with a description of the weather, the time of sunrise and sunset, wind force, the pressure, min and max temperatures.

### Vets

This is where the user can view their upcoming and past vet visits. This allows the user to enter upcoming appointments with the ability to choose the date from a calendar. This also offers the user to compare their vet bills against stored invoices and receipts.

### To Do List

This is where the user can enter tasks they must complete on the farm. This gives the option to tick it off on completion of the task, while moving the completed task to the end of the list marked as done.

### Nearby Places

The current location of the user is picked up and displayed on a visual map. This contains buttons for the nearest “hospitals”, “mechanics” and “vets” and once pressed it will display the nearest options in their vicinity. A search bar is also given to the user to allow for own preference searches. There is also a button which sends the users current location to other app users if the current user is in search of help.

### Emissions

This is where the user can calculate the average emissions on their farm. There are two options to calculate emissions on, Beef or Dairy. For Beef, the user has to enter the average weight of both cows and bulls which then calculates the total average emissions produced from beef livestock. This takes into account everything that is needed to get a beef animal to sale. For Dairy, the user has to enter the number of cows they have producing milk and their average milk yield per animal/ per annum. Recommendations are given for either category with a link for further research corresponding to the option chosen i.e. Dairy option will display ways for dairy farmers to reduce emissions. A link to www.teagasc.ie is also provided.

### Invoices/ Receipts

The user can view both their invoices/ receipts for a multitude of categories including livestock, hardware, contractors, machinery, feed and supplies. This also allows users to upload a picture of the receipt or invoice, along with the total amount and date.

## Background Processes

The system is required to keep track of the insemination date of cows and supply the user with a notification one week in advance of the expected due date. The system is also responsible for sending the current user's location to other app users in the search of help.

# Functional Requirements

## User Use Case Diagram



## Requirements

|  |  |  |
| --- | --- | --- |
| **Function** | **Requirement** | **Justification** |
| **User Register** | The ability for the user to create an account by inputting their credentials. | In order to use the app functionalities, the user must log in. For the user to be able to log in, they must have an account. |
| **Login / Logout** | The ability for the user to login and logout of the app. | Login function enables data associated with the current user to be manipulated while logged in. The user must also be able to log out of the application. |
| **Enter Animal Section** | The ability to enter the animal section of the app and view their animals. | A user should be able to view their animals, insert/ update animal info and search an animal via barcode scanner. |
| **Enter Weather Section** | The ability to enter the weather section of the app and view current weather forecast. | A user should be able to view the current weather conditions and view the measurement of such events e.g. wind, min/max temp. |
| **Enter Vet Section** | The ability to enter the vet section of the app and view upcoming appointments. | A user should be able to view past and upcoming vet appointments while also being app to enter/update/delete an appointment. |
| **Enter To-Do List Section** | The ability to enter the to-do list section of the app and tick off tasks on completion. | A user should be able to insert/ update/delete tasks on their to-do list while also be able to tick off once complete. |
| **Enter Nearby Places Section** | The ability to enter the nearby places section of the app and view their current location on a visual map. | A user should be able to view location, press a descriptive button which displays the nearest resources in regard to button pressed. E.g. Show nearest mechanics.  The user can search for their own preference location while also being able to send their location to other app users in the search for help. |
| **Enter Emissions Section** | The ability to enter the emissions section of the app and calculate the average emissions for either beef or dairy based categories. | A user should be able to calculate beef/dairy emissions and get a visual of the results found with the option to view recommendations on how to reduce emissions for both categories while also being able to be directed to the Teagasc website. |
| **Enter Invoice/Receipt Section** | The ability to enter the invoice/receipt section of the app. | A user should be able to view their invoices/receipts under the given categories and also the option to insert/update/delete them. |
| **Enter the Insemination Date of a Cow** | The ability to click in calve and enter the insemination date of the cow. | A user should be able to insert the date and the system should send notification 7 days prior to the expected due date. |
| **Enter the list of impregnated Cows** | The ability to view list of impregnated cows. | A user should be able to click on the calving date alarm notification and view the list of impregnated cows with the option to enter extra calving information for specific cow. E.g. Mastitis after last pregnancy. |

# Technology overview:

## Android Studio:



I chose Android Studio, the official IDE for Android to develop my mobile application. I was intrigued to work and learn more about mobile application development. It has a user-friendly UI that provided simple steps to get the project connecting to Firebase, Firestore and Git-hub. For the front-end, I used XML files within Android Studio, where I then used the Java standard library in addition to the Android library for the back-end development of the app.

## Database

## Cloud Firestore



Initially, I couldn’t decide what database to use. After ongoing research and numerous changes, I settled on Cloud Firestore. This is the latest addition to Firebase and Google Cloud Platform products. Firestore is a cloud-based NoSQL database, with advanced features from the original Firebase Realtime Database. It offers live synchronization, a server-less solution and offline support for Android development along with more reliable and efficient features then other databases. In comparison to Firebase Realtime Database, Firestore provides richer and faster queries, whilst being more scalable than the Realtime database. Google has proposed that this will eventually replace the Firebase Realtime Database.

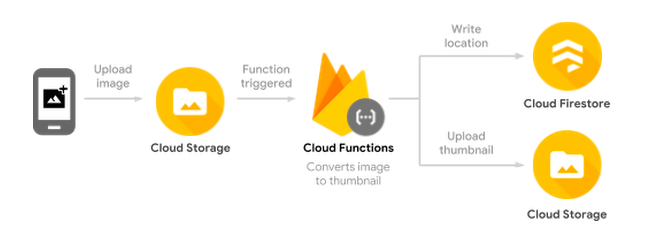
Other Technologies

## Firebase Cloud Messaging



I used the Firebase Cloud Messaging (FCM) platform to send push notifications to users, which will appear in the user's device system tray. Users are reminded of expected calving dates a week in advance and they are also given the option to send their location via a notification to other app users. Personally, FCM has been proven to be more reliable than other solutions for sending notifications, as after the application is restarted FCM ensures that the notification is still processed whereas other solutions were unpredictable.

## Cloud Storage and Cloud Functions



Firebase Cloud Storage allowed me to store and manage various photo content generated by the users of my app. I used Firebase Cloud Functions within the app to process uploaded animal photos and invoices/ receipts. The code is triggered once data is added to or updated in the database or when files are added to Cloud Storage. Cloud Functions optimizes the size and resolution of uploaded photos.

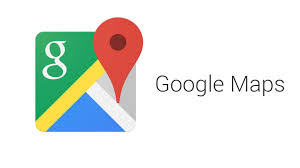
## MP Android Chart

This powerful library allowed me to incorporate a pie chart in order to display results from the calculated Beef Emissions. MP Android Chart provides a visual and pleasant way to display data in different formats e.g. decimal or percentages.

## Open Weather API

I incorporated functionality where current weather data was received with the use of Open Weather API. This API returns data such as temperature, wind speed, sunrise/ sunset etc. from the given location in JSON format. I then converted these findings and displayed them back to the user.

## Google Maps API

I used the Google Maps API to retrieve the current location of the user’s device. The current location is then displayed using a marker on the map. I used the map to zoom into the user's location to give them a clear visual of their location. I also implemented an option for the users to send their location to other app users if they were looking for help, through push notifications.

## Google Places API

After using Google Maps API successfully, I went on to extend the functionality of this by adding a “Find Places Nearby” option. Google provides an API called Google Places that allows users to search for a specific address and nearby resources. I used this API to allow users find nearest vets, mechanics and hospitals with the press of a button.

## Glide

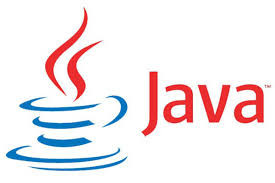
After demoing and receiving feedback on my app, it was suggested that I should allow users to upload their own personal images into the app. For example, the user can take their own picture of an animal and save it as the animal’s profile picture.

In order to achieve this, I used Glide, a powerful image loading library for Android. Glide came with many features such memory/disk cache, transformations, transitions and image resizing, cropping and rotation. It was suggested that I should use Picasso to do this, however, I found that Glide was faster and used a smaller amount of memory cache in comparison to Picasso’s.

Canva

Another piece of feedback I received from family members was to incorporate an overall theme into the design aspect of my project. I chose to use Canva a graphic design platform to achieve this and extend the simplistic design approach in Android Studio. Canva offers a wide range of design templates where one can create design easily.

I was able to create an overall template for my app which allowed me to select photos and incorporate a branding color which in return reflected farming. I designed the File a Farm Logo where I used different symbols to illustrate different farming aspects of the app. I incorporated this design with a circular image to create the splash screen for the app. I believe that branding and the overall esthetic is the cornerstone of a good marketing strategy in the technology industry.

Java

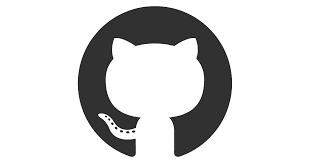
The backend of my app was written in Java. This is my primary language which I started to learn from the offset. I felt most comfortable using Java in order to advance the complexities within my app and is commonly used in Android development.

JSON

JavaScript Object Notation (JSON) is an open standard file format that is used for interchanging. I used JSON as it allowed me to transmit data between my application and various web services.

# Version Control

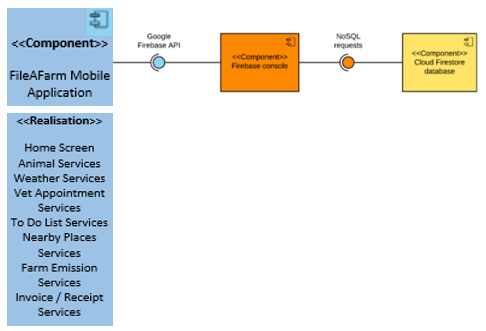
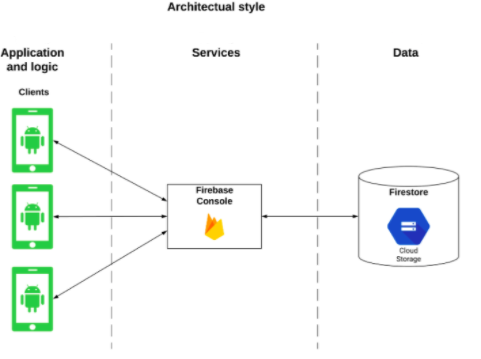
## GitHub



Throughout the development of my app, I used GitHub’s free code hosting platform for version control. GitHub’s version control is of the highest standard. As this project is ever growing, the use of version control is essential. It allowed iterations of new code to be developed without the fear of breaking something as it kept track of the version history of code throughout.

GitHub enabled me to keep track of any changes I made, while saving my code in a secure repository. It also ensured that I had constant access to previous versions in a situation where new versions were causing issues. As GitHub is cloud-based, it can be accessed easily from multiple locations which was very useful in the development of my app. The Version Control service (VCS) provided by Android Studio, is a powerful yet easy option to integrate GitHub with Android Studio. The features provided by VCS form an easy and comfortable way to manage and track code.

# System Architecture



# Database Model

The database I chose was Firestore which is a non-relational database. Therefore, the traditional concept of a relational database model does not apply. However, each table has a common attribute to a specific user which is the user’s id. This is how objects specific to the user are correctly identified and retrieved in the queries created in Android Studio. In this way the layout of the database is relational in nature. I constructed the below diagram to detail the properties of each Firestore collection and how they relate back to the specific user.



# Implementation – UI Design and Functionality Reference Guide

|  |  |
| --- | --- |
| *Splash Screen – Splash Activity:* | |
| A picture containing grass, man, green, front  Description automatically generated | Upon entering the app, the user is presented with the corresponding splash screen. This gives the user their first initial welcome to the File a Farm App. |

|  |  |
| --- | --- |
| *User Login - Login Activity:* | |
|  | The user logs in with their registered email address and password.  This is an essential function of the app.  If the user hasn’t already created an account, they are given the option to Sign Up which directs the User to the Register Activity.  Once the user is logged in successfully, the Main Screen is launched. |

|  |  |
| --- | --- |
| *User Registration - Register Activity:* | |
| A screenshot of a cell phone  Description automatically generated | User can register an account with a valid email address, password and their herd number.  Once the user presses the Sign-Up button, their account is authenticated using the Firebase email and password-based authentication services.  The user’s herd number must be unique, or else a warning will be signified.  Once the user has successfully registered an account they are automatically logged in and presented with the main screen.  If the user already has an account, they can be redirected to the Login page. |

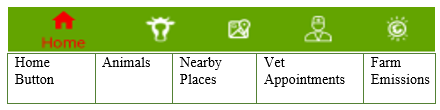
A screenshot of a cell phone

Description automatically generatedQuerying the database to check if herd id number already exists:

|  |  |
| --- | --- |
| Home Screen – Main Activity: | |
|  | Once the user has successfully logged in, the home screen is presented to them. This screen consists of the multiple activities/ services available within File A Farm.  It gives access to the following screens:   * Animals * Weather * Vets * To Do List * Nearby Places * Farm Emissions * Invoice or Receipts   These activities will be discussed below.  This activity consists of a navigation drawer which can be accessed by swiping from the left edge of the screen or by the 3-line menu button available on the top left side of the screen. |

**Bottom Navigation**

Each activity contains a bottom navigation menu which gives users easy access to the 5 essential functions of the app.

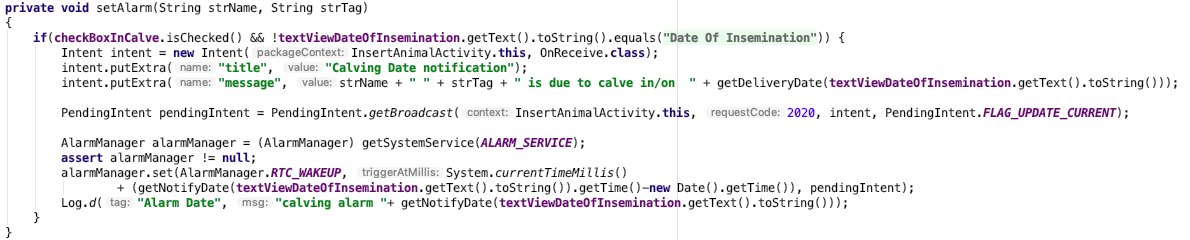


|  |  |
| --- | --- |
| Side Navigation – Main Activity: | |
|  | This is accessible through the Main Activity.  The logged in user’s email address is displayed under the File a Farm logo. This offers quick navigations through the different activities and also allows the users you to logout of the app. |

|  |  |
| --- | --- |
| Animals – Animal Activity: | |
|  | A list of animals that are registered to the current user is displayed here. The animal’s tag number, name, date of birth and breed are displayed in the card views with the option to update the animal's information.  The Scan Barcode option at the top of the screen, allows the user to make a quick search by scanning the animals ear tag/ passport. This retrieves the information of a specific animal.  The plus button allows the user to insert a new animal into their herd list. On Click listeners are on the animal’s view card along with an edit icon which allows the user to update the animal’s info accordingly. |

|  |  |
| --- | --- |
| Insert Animal – Insert Animal Activity: | |
| 1. *If “in Calve” is ticked*   Inserting image...   1. *If “in Calve” is unticked and gender is “Female”.*   A screenshot of a cell phone  Description automatically generated | Here the user can insert an animal into their herd list. The user can insert an image for the animal's profile, or a default animal icon will be set.  The user must insert the following details in order to successfully insert the animal into the database:   * Tag Number (unique) * Animal Name * Date of Birth * Gender * Dam – Mother * Calving Difficulty (1-5) * Select whether the animal’s mother was inseminated by Artificial Insemination (AI) or by the Stock Bull. * Sire – Father * If the user has selected a female animal the option to flag whether the animal is in calve on point of insertion. * If the “in calve” checkbox is ticked the user will have to select an insemination date, which will estimate an expected delivery date and set an alarm to remind the user of the due date 7 days before expected arrival date. *(Image 1)*. * If the “in calve” checkbox is left unticked the Date of Insemination and Expected Delivery fields will remain invisible. *(Image 2).* |

|  |  |
| --- | --- |
| Calving Expectation Alarm Date Implementation: | |
| *Calving Date Notification Received* | A notification alert is sent to the farmer’s phone when a cow’s calving date is near. The gestation period for a cow takes on average 283 days. I created an algorithm to calculate the delivery date of the notification while taking the gestation period into account.  When the notification is clicked a list of cows that are expecting to calve is displayed. If the user clicks on a specific cow from the list an alert dialog appears allowing the user to enter further calving information such as previous birthing difficulties, fertility, calving diet etc. |

Below is a snippet of the set alarm code implementation:

|  |  |
| --- | --- |
| Scan Barcode Implementation: | |
|  | The system is designed to scan the barcodes off the animal tag or off the animal’s passport card also known as the “Blue Card”. (Refer to the images below).  The barcode scanner returns a value in numerical format which is then passed in to fetch the data of the related animal which was scanned. The result is shown in a pop-up dialog. This offers quick retrieval of an animal’s information.  The user must grant permission before using the camera in order to scan the barcodes.  A screenshot of a cell phone  Description automatically generated |

|  |  |
| --- | --- |
| *Passport – “Blue Card”* | *Animal Tag* |

A screenshot of a social media post

Description automatically generated As seen in the picture below, the result of barcode scanner is received, and the corresponding animal is retrieved.

|  |  |
| --- | --- |
| Weather – Weather Activity: | |
|  | As the majority of the app’s users spend a considerable amount of time outdoors, I decided it was suitable to incorporate a latest weather feature into the app.  The weather is based on the user's geolocation. It provides the current weather temperature, min and max temp, time of sunrise and sunset, wind speed and measures of pressure and humidity. The location and time in which the weather was updated is also displayed with a quick description of the weather.  I used Open Weather Maps API to implement this functionality. This involved making a HTTP request to get the result from the API which is in JSON format and then converting it into String format to be displayed. |

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

|  |  |
| --- | --- |
| Nearby Places using Current Location: | |
| *Push Notification Received* | The map fragment shows the current location of the user's device. The user has the option to type in a keyword into the “Search here” text box to search for a specific location. I also incorporated three shortcut buttons which searches for nearest vets, mechanics and hospitals with the press of the button.  When search results are returned, markers are drawn on the map signifying matches. When you press on the marker of a location two image buttons appear on bottom right, when they’re clicked it will give you directions to the selected destination.  Users can send their location to other app users if they were looking for help by simply pressing send location. Firebase Cloud Messaging services sends a push notification with the current user's location details to the other users. |

|  |  |
| --- | --- |
| To Do List - To Do List Activity: | |
|  | To Do List function where users can add checklist items and tick them off as they are complete.  As farmers could have a list of things to do on a daily basis, I wanted to incorporate a service where the users can see a clear outline of their completed and uncompleted tasks which helps them to manage their tasks making sure each task gets ticked off.  The date and time of when the task is visible, and it sets an order to the list. Once a task is ticked off it moves down the list. I created click listeners on each task item allowing users to add, update and delete tasks as the wish. |

|  |  |
| --- | --- |
| Vet Appointments - Vet Activity: | |
|  | Users can enter their upcoming appointments for vets. Once the user fills out the mandatory fields the appointment can be saved.  I incorporated a date picker which allows the user to pick the appointment date.  This service allows farmers to keep track of their vet appointments while being able to correspond the appointments to the vet bills.  I created click listeners on each appointment allowing users to add, update and delete appointments as the wish. |

|  |  |
| --- | --- |
| Farm Emissions – Emissions Activity: | |
|  | It is immediately apparent when one looks at the media that there is a growing concern in regard to the amount of greenhouse gas emissions produced by the agriculture sector.  Given this newfound and somewhat unfair pressure on farmers I wanted to develop a relatively easy way for farmers to calculate the emissions their herd produces.  Through extensive research I found providing a 100% accurate calculation near impossible to implement solely in a mobile app. Speaking with numerous agricultural lecturers between Ireland and the US a variety of equipment would be needed by farmers to gauge emissions with pinpoint precision.  At this point I decided the best approach would be to use a calculation that would provide a rough estimate of herd emissions. There were two separate calculations to determine an average for a dairy herd and a beef herd. These are outlined below. |

|  |  |
| --- | --- |
| Calculate Beef Emissions Activity: | |
|  | Here the user can calculate the average Beef Emissions produced on a farm.  In order to calculate this the user must enter the average weight of a beef cow on their farm and the average weight of a beef bull in kilograms.  *I found the calculation below in my research:*[[1]](#footnote-2)  “The average CF in the entire beef cattle production system was 25.43 kg of CO2 kg-1 of live weight of marketed cattle”  To take into account everything that it takes to get a beef animal to sale:  Each cow weight in kg x 25.43kg  Example: Farmer has 4 cows and 4 bulls. We take the average of each gender and perform the following calculation e.g. average cow is 800 kg and an average bull is 1000kg. |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Average Dairy Emissions Results* | |  |  |  | | --- | --- | --- | |  | No. Of Gender X Weight KG X CO2 KG-1 | = Amount of CO2 Produced | | Cows | 4 X 800 X 25.43 | = 81,376 KG | | Bulls | 4 X 1000 X 25.43 | = 101,720 Kg |   Total = 183,096 KG of CO2 produced by 4 cows and 4 bulls.  183,096 KG = Average Emissions produced by the herd— Taking into account the entire maintenance of getting a cow or bull ready for market. |

The calculation uses both database data and user input data; this is then passed into a method which renders a pie chart that displays Beef Emissions according to gender of the animal.

A screenshot of a social media post

Description automatically generated

|  |  |
| --- | --- |
| Calculate Dairy Emissions Activity: | |
| *Average Dairy Emissions Results* | Here the user can calculate the average Dairy Emissions produced on a farm.  In order to calculate this, the user must enter the number of milk-producing cows they have and the overall average milk yield per cow per annum in litres.  *I found the calculation below in my research:[[2]](#footnote-3)*  “The average carbon footprint for [...] milk production was 1,309g of carbon dioxide equivalents per litre (g CO2e/l).”  For example, I have 250 dairy cows who each produce an average yield of 5000 litres per annum.  The calculation is outlined below:   |  |  | | --- | --- | | No. Of Cows X Average milk yield L p.a. X (CO2e/l /1000) | = Average Amount of CO2 produced by no. of dairy cows per annum | | 250 X 5000 X (1309 / 1000) | = 1,636,250 KG |   1,636,250 KG = mean amount of CO2 produced per dairy herd per annum, therefore if the herd consistently produces milk for 5 years it will have produced on average 8,181,250 KG of CO2 in this period. |

|  |  |
| --- | --- |
| Recommendations - Ways to Reduce Emissions Activity: | |
| *Reduce Beef Emissions* A screenshot of a cell phone  Description automatically generated *Reduce Dairy Emissions* | Following the display of the emission results the user has the option to view ways to reduce emissions based on which calculate activity they chose. i.e. Calculating Beef Emissions will show you ways how to reduce emissions in regard to Beef farming.  Within the Ways to Reduce Emission activity I used a JSON file  A screenshot of a cell phone  Description automatically generated  @SerializedName annotation is used for connecting the object variables with its counterparts in JSON.  Depending on whether the user has performed a beef calculation or dairy calculation the app will retrieve a corresponding recommendations PDF through a URL from the Teagasc website.  A link is also provided to the official Teagasc website for the farmer to peruse if they wish. |

### ***External Links from the Way to Reduce Emissions Activity***

|  |
| --- |
| 1. Beef Emissions PDF 2. Dairy Emissions PDF 3. Teagasc Website |
| Inserting image... |

|  |  |
| --- | --- |
| Invoice/ Receipts – Invoice/ Receipts Activity: | |
|  | In the Invoices and Receipts Activity, I store the invoices and receipts on [Cloud Storage](https://firebase.google.com/docs/storage/) for easy storage and retrieval. The expenses structured data, such as the type of expense, the category it goes under, the amount, date and who registered the invoice/ receipt is stored in my Cloud Firestore database, which is integrated seamlessly with Cloud Storage through the use of Cloud Functions.  Unlike adding an animal’s profile picture which is optional as a default image will be set if no picture was selected, the invoice/ receipt image is a mandatory upload. |

# **Test Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Test Scenario | Expected Result | Test Result: Pass/Fail |
| Launch the App | Splash Screen appears. | Splash Screen is displayed and opens on the registration/ menu page. | PASS |
| Register New User | User attempts to register account with unique email address and herd id | User account is created, and data is added to the database | PASS |
| Register New User | User attempts to register account with existing email address | User account is not created, and user is warned that email address already exists | PASS |
| Register New User | User attempts to register account with existing herd id | User account is not created, and user is warned that herd id already exists | PASS |
| Register New User | Attempt to leave an empty field when creating an account | User receives error message. | PASS |
| Register New User | Attempt to create an account with a non- formatted email or password less than 7 characters | User receives error message. | PASS |
| Login | Ability of a user to be able to login into their account. | User is logged in successfully. | PASS |
| Login | Attempt to login to their account while leaving a field empty. | User receives error message. | PASS |
| Login | Attempt to login to account with invalid username or password. | User receives error message. | PASS |
| Navigation Menu | Attempt to open navigation drawer by sliding left in the main menu. | Navigation Drawer menu is shown. | PASS |
| Insert animal | Attempt to add an animal to their herd by filling in all appropriate fields. | Animal is added to the database and added to the list of existing animals within user herd. | PASS |
| Insert animal | Attempt to insert animal’s tag number that is already in the database. | A new animal is not added to the database and user is warned that the tag number already exists. | PASS |
| Insert animal | Attempt to insert an animal with a missing mandatory field. | A new animal is not added to the database and user is prompted to fill in mandatory fields. | PASS |
| Insert animal | Attempt to insert calving date for a male animal. | When user selects male animal calving options disappear and no calving data is added to the database. | PASS |
| Remove an animal | Delete an animal associated to the user’s account. | Animal removed from user view and database. | PASS |
| View all animals | Select animal’s action in main menu. | Only animals associated to the current user is displayed. | PASS |
| Search Animal | Scan Animal’s passport or tag. | Animal’s record is returned. | PASS |
| Update Animal | Update animal fields on animal record. | Newly updated information populates the animal record and database. | PASS |
| Update Animal | Attempt to update an animal with a missing mandatory field. | The animal is not updated in the database and the user is prompted to fill in mandatory fields. | PASS |
| Update Animal | Attempt to update calving information for a male. | When user selects male animal, calving options disappear and no calving data is updated in the database. | PASS |
| Calving Date Notification Alarm | Receive expected delivery date for a specific animal. | Notification is received on the user's phone. | PASS |
| Calving Date Notification Alarm | Click on the notification. | List of animals that are expecting to calve are returned. | PASS |
| Calving Date Notification Alarm | Enter calving information for an animal. | Click on animal from the expecting to calve list and add calving information or past calving difficulties. | PASS |
| View the weather | View the current weather forecast. | User’s current weather is shown with data for that location visible. | PASS |
| View Vet Appointments | Select Vet Appointment in the main menu. | List of the user’s vet appointments are shown. | PASS |
| Add Vet Appointment | Attempt to add an appointment to their list by filling in all appropriate fields. | Appointment is added to the database and added to the list of existing appointments for that user. | PASS |
| Add Vet Appointment | Attempt to insert an appointment with a missing mandatory field. | A new appointment is not added to the database and user is prompted to fill in mandatory fields. | PASS |
| Remove Vet Appointment | Delete an appointment associated to the user’s account. | Appointment removed from the users view and database. | PASS |
| Update Vet Appointment | Update appointment fields on appointment record. | Newly updated information populates the appointment record and database. | PASS |
| View Nearby Places | Select Nearby Places in the main menu. | Map is shown with current location. | PASS |
| View Nearby Places | User selects hospitals, mechanics or vets' button. | Locations corresponding to the selected button are shown with a marker. | PASS |
| View Nearby Places | User enters search keyword in the search bar. | Location matching the keyword is shown on the map with a marker. | PASS |
| Send Location | User presses send location button. | User’s subscribing to the app receives the location of the transmitting user. | PASS |
| Receive Location | User opens location notification sent by another user. | Notification of the transmitting user location is received. | PASS |
| View the To Do List | View the to-do list items. | List of the user’s checklist items are shown | PASS |
| Add item in the To-Do List | User adds an item into their to-do list. | Item adds into the to-do list. | PASS |
| Edit item in the To Do List. | User edits an item that has been created. | Item is updated with new information. | PASS |
| Calculate Farm Emissions | Select Emissions in the main menu. | Calculate Emissions option is shown. | PASS |
| Calculate Beef Emissions | Select Beef Emissions from the Farm Emissions screen. | Beef Emissions calculations window appears. | PASS |
| Calculate Beef Emissions | User presses calculate emissions without entering any data. | User is prompted to enter data into at least one field. | PASS |
| Calculate Beef Emissions | User enters data into the fields and selects calculate emissions. | Calculate result window is shown with emission results. | PASS |
| Beef Emissions Results | User is directed to the Beef Emissions results. | Results are shown and the pie chart is rendered. | PASS |
| Beef Emissions Results | Recommendations to reduce Beef Emissions. | Bullet point recommendations for beef farmers are shown with a PDF link and external website link. | PASS |
| Beef Emissions Results | User clicks on PDF link. | Teagasc Recommendations PDF is loaded for Beef farmers. | PASS |
| Beef Emissions Results | User clicks on Teagasc website link. | User is directed to the Teagasc website. | PASS |
| Calculate Dairy Emissions | Select Dairy Emissions from the Farm Emissions screen. | Dairy Emissions calculations window appears. | PASS |
| Calculate Dairy Emissions | User presses calculate emissions without entering any data. | User is prompted to enter both data fields. | PASS |
| Calculate Dairy Emissions | User enters data into the fields and selects calculate emissions. | Calculate result window is shown with dairy emission results. | PASS |
| Dairy Emissions Results | User is directed to the Dairy Emissions results. | Results are shown and a link for recommendations. | PASS |
| Dairy Emissions Results | Recommendations to reduce Dairy Emissions. | Bullet point recommendations for dairy farmers are shown with a PDF link and external website link. | PASS |
| Dairy Emissions Results | User clicks on PDF link. | Teagasc Recommendations PDF is loaded for Dairy Farmers. | PASS |
| Dairy Emissions Results | User clicks on Teagasc website link. | User is directed to the Teagasc website. | PASS |
| View Invoice/ Receipts | Select Invoice/ Receipts in the main menu. | Invoice/ Receipts are shown.. | PASS |
| Add Invoice/ Receipt | Attempt to add invoice/ receipt to their list by filling in all appropriate fields. | Invoice/ Receipt is added to the database and added to the list of existing invoice/ receipts for that user. | PASS |
| Add Invoice/ Receipt | Attempt to insert an invoice/ receipt with a missing mandatory field. | The invoice is not added to the database and the user is prompted to fill in mandatory fields. | PASS |
| Remove an Invoice/ Receipt | Delete an invoice/ receipt associated to the user’s account. | Invoice/ Receipt is removed from the users view and database. | PASS |
| Logout | Attempt to log out of their user’s account. | Log out successfully and the Login page is shown. | PASS |

# Implementation Issues

I faced a number of hurdles during the implementation of the File a Farm app. There was a number of design consideration and one of the initial decisions I found difficult was choosing which database service to pick. There are a number of technologies out there each with their pro’s and con’s. I choose Firestore as it came with a lot of functionality that suited the requirements of my app. As it was my first time to use a NoSQL database, there was a learning curve in how to implement an appropriate model and interact with the database for querying and data manipulation.

Inserting images was a challenging issue I faced. I tried various libraries with a lot of trial and error, but I then choose Glide as it was most suitable to my app. Sending notifications took a considerable time to get working correctly. However, once I got my head around the concept it eventually worked out.

Another major issue was the amount of concrete research available for creating a farm emissions calculator. I spoke to numerous researchers in this field and I was somewhat shocked at how much equipment would be needed to accurately track agriculture emissions. All the relevant research eventually pointed back to the Teagasc website. After talks with my lecturers, I decided the best approach was to create an average emissions calculator based on dairy and beef formulas discovered during my research. I also offered a link to the external Teagasc website which can be accessed after the average emissions are calculated and the recommendations on how to reduce emissions based on the herd type are shown.

Unfortunately, I didn’t get to implement all the features I wished to complete. If anything, this has given me great insight into feature estimation and factoring in the unknowns during the planning stage.

Needless to say, I have learned many lessons while developing my first full scale application. Firstly, I would use a planning tool such as JIRA to accurately track sprints as opposed to a word document. This would have given be a more detail view of the progress of the project, issues still outstanding and my achievements. I would also invest more time in planning out my project and ensuring that the majority of the research is completed beforehand, as research in this app consumed some development time.

# Conclusion

To conclude, the overall experience of final year was challenging but extremely rewarding. I grew a lot in confidence and technical ability which will stand to me when I graduate. I learnt how to deal with multiple project blockers and inevitable difficulties one of which was the COVID-19 pandemic. The closure of the college was distressing as supports were no longer available immediately during this time. However, I persevered, and I am happy with the result.

The initial start of the project was quite daunting with the range of technologies available. With no previous experience in developing a fully functional app, I had a considerable learning curve in front of me. I overcame this and now I have understanding and appreciation of developers in the work force who face these challenges day in and day out. Android and iOS users take for granted the amount of work and passion that goes into developing an app.

From a business perspective, I feel that I have filled a gap in the market for a simple, easy to use farm management system. I’ve demonstrated the app to local farmers in my area and the feedback has been very positive. Given this positivity, I am going to continue to implement new features and improve upon the app.

Overall, I really enjoyed creating File a Farm. I believe the final year project was a necessary steppingstone to ready me for the professional working environment which awaits me.

# References

* Retrieved 27 March 2020, from <https://www.teagasc.ie/media/website/publications/2019/Bord-Bia-Dairy-Carbon-Navigator-LR5.pdf>
* Retrieved 27 March 2020, from <https://www.teagasc.ie/media/website/publications/2019/Bord-Bia-Beef-Carbon-Navigator-LR4.pdf>
* Retrieved 10 February 2020, from https://www.canva.com/
* bumptech/glide. (2020). Retrieved 27 April 2020, from https://github.com/bumptech/glide
* Cloud Storage | Firebase. Retrieved 27 February 2020, from <https://firebase.google.com/docs/storage>
* Firebase. Retrieved 18 March 2020, from <https://firebase.google.com/>
* Latest Farming News - Farming in Ireland - Farm Ireland - Agriland. (2020). Retrieved 27 April 2020, from <https://www.agriland.ie/>
* PhilJay/MPAndroidChart. Retrieved 12 April 2020, from <https://github.com/PhilJay/MPAndroidChart>
* Serverless in action: building a simple backend with Cloud Firestore and Cloud Functions. Retrieved 27 April 2020, from <https://cloud.google.com/blog/products/application-development/serverless-in-action-building-a-simple-backend-with-cloud-firestore-and-cloud-functions>
* Teagasc | Agriculture and Food Development Authority. Retrieved 27 March 2020, from <https://www.teagasc.ie/>
* Weather API - OpenWeatherMap. (2012). Retrieved 27 April 2020, from <https://openweathermap.org/api>

# Footnotes

1. Bieńkowski, J., Dąbrowicz, R., Holka, M., & Jankowiak, J. (2018). Carbon Footprint of Beef Cattle in a Conventional Production System: a Case Study of a Large-Area Farming Enterprise in the Wielkopolska Region. *Zeszyty Naukowe SGGW W Warszawie - Problemy Rolnictwa Światowego*, *18(33)*(3), 23-35. doi: 10.22630/prs.2018.18.3.63, from <https://www.researchgate.net/publication/330694603_Carbon_Footprint_of_Beef_Cattle_in_a_Conventional_Production_System_a_Case_Study_of_a_Large-Area_Farming_Enterprise_in_the_Wielkopolska_Region>
2. Greenhouse gas emissions on British dairy farms DairyCo carbon footprinting study: Year one. (2012). Retrieved 6 February 2020, from <https://dairy.ahdb.org.uk/media/623464/greenhouse_gas_emissions_on_british_dairy_farms.pdf>

1. <https://www.researchgate.net/publication/330694603_Carbon_Footprint_of_Beef_Cattle_in_a_CConventional_Production_System_a_Case_Study_of_a_Large-Area_Farming_Enterprise_in_the_Wielkopolska_Region> [↑](#footnote-ref-2)
2. <https://dairy.ahdb.org.uk/media/623464/greenhouse_gas_emissions_on_british_dairy_farms.pdf> [↑](#footnote-ref-3)