INSTITUTE of TECHNOLOGY

CARLOW

Institiúid Teicneolaíochta Cheatharlach

Name : Damien Doran

Student Number : C00221791

Project title: Teagasc app

Document : Design document

Date : 2020/2021

Table of Contents

Contents

1.0 Introduction 3

2.0 Project Planner 4

3.0 Project Plan 4

4.0 Teagasc Application Visualised 5

.... 6

5.0 System Architecture 7

6.0 Database 8

7.0 User Interfaces 9

- 7.1 ConductAssessment 10
- 7.2 Import / Export Nitrates 13
- 7.3 Storage 15
- 7.4 Farm Records 16
- 7.5 Update LSU (Livestock unit) 16

8.0 Database Design (there is a django generated model in diagram folder) 18

.... 19

Figure 11.1 ER Diagram 19

9.0 Database Tables 19

- 9.1 Farmer Model 19
- 9.2 Grassland Model 20
- 9.3 Importation and Exportation Models 20
- 9.4 Monthly Livestock Numbers 21
- 9.5 Slurry Storage 21
- 9.6 Farmer Livestock 21

10.0 Technologies 22

- 10.1Django 22
- 10.2 MySQL 22
- 10.3 Python 22

11.0 References 23

Table of Figures

Figure 1: Project Planner.......... 4
Figure 2: System Architecture.......... 6
Figure 3 System Architecture.......... 6
Figure 4: System Architecture.......... 7
Figure 5: System Architecture{1}.......... 8
Figure 6: Monthly Livestock Numbers.......... 9
Figure 7: Login Screen.......... 9
Figure 8: Home Screen.......... 10
Figure 9: Conduct Assessment 1.......... 11
Figure 10: Conduct assessment 2.......... 12

1.0 Introduction

This project aims to create a web application where Teagasc advisors can perform swift nitrate management assessments of farmers. The project will develop a user friendly UI and store all data securely in a database. The project will consist of the main web application and a mysql database. Records will be maintained of each assessment and the web application will be designed with security in mind. The main goal is to deliver simple concise reports for farmers upon completion of accurate assessment.

2.0 Project Planner

Figure 1.1 displays the projected timelines of development.

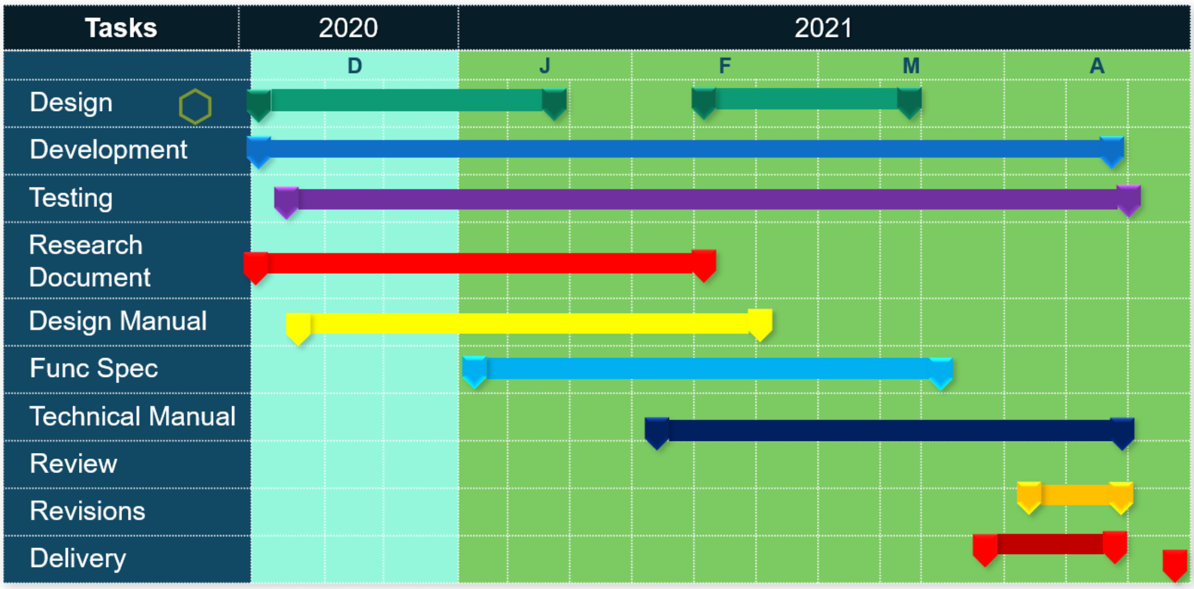


Figure 1: Project Planner

3.0 Project Plan

This project was requested by Robert Sherriff from Teagasc offices Gorey, and as such was treated as a real life project. The first meeting was late September where the project was discussed and documentation regarding calculations were received. The plan was to schedule 3 meetings throughout the college year for feedback and discussion. The first meeting was arranged for November where discussion of how the application would look and feel was discussed.

In January a second meeting was scheduled where a prototype was demoed and unfortunately did not meet the expectations. Following calculations from another application meant the design was beginning to mimic the current software. This was deemed unsatisfactory by Robert who made the request for simpler design requiring less input if possible and to carry out a Grassland Assessment returning a grassland stocking rate and a whole farm stocking rate. Robert also requested an implementation of a livestock unit calculator, making the application unique. The third meeting was set for March just before project submission.

The application needed complete refactoring and a new design with new features implemented. This meant more research and breaking down calculations to the smallest possible form to minimise work for an advisor.

The project was finally demoed on April 22nd, with Roberts commitments to work and college work the meeting scheduled for March was postponed. At the meeting incredible feedback was received, with the application due to be further improved. On 24/04/21 the application was deployed. Teagasc offices in Gorey accessed the application on the 27/04/21 with very positive feedback. On the 27/04/21 the application was used by the Enniscorthy offices again resulting in very positive feedback. The application will be continuing to be developed with new functionality to be demoed to the Department Of Agriculture at the request of Robert Sherriff.

4.0 Teagasc Application Visualised

This is a visualisation of the Teagasc Advisors Application created using the django-extensions module. This returned a very large image and was difficult to fit into the document in a readable format. The full image can be viewed here clearly. https://drive.google.com/file/d/1BYO5r_Mc3sUVOcMgeGIBk2CrH2tGeIpZ/view?usp=sharing

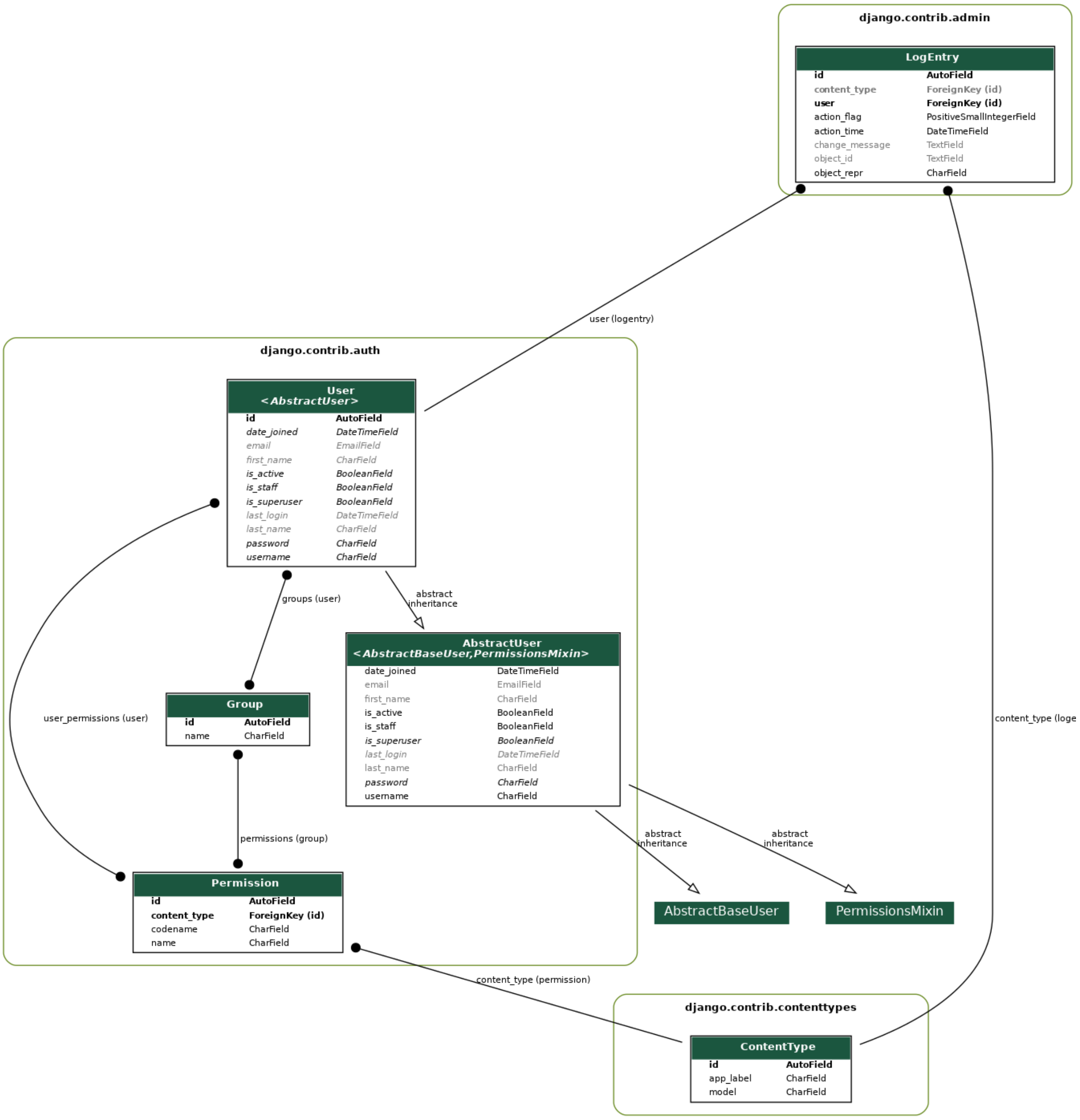


Figure 2: System Architecture

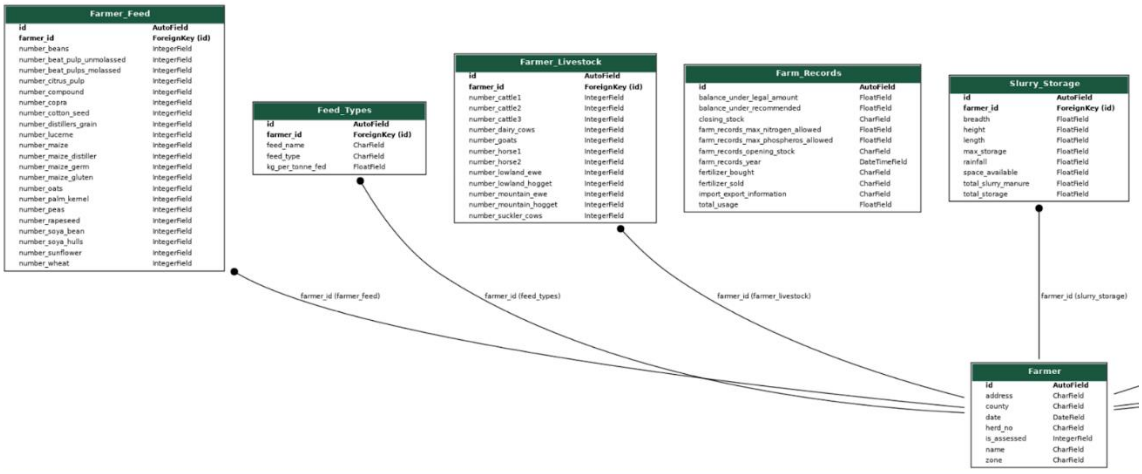
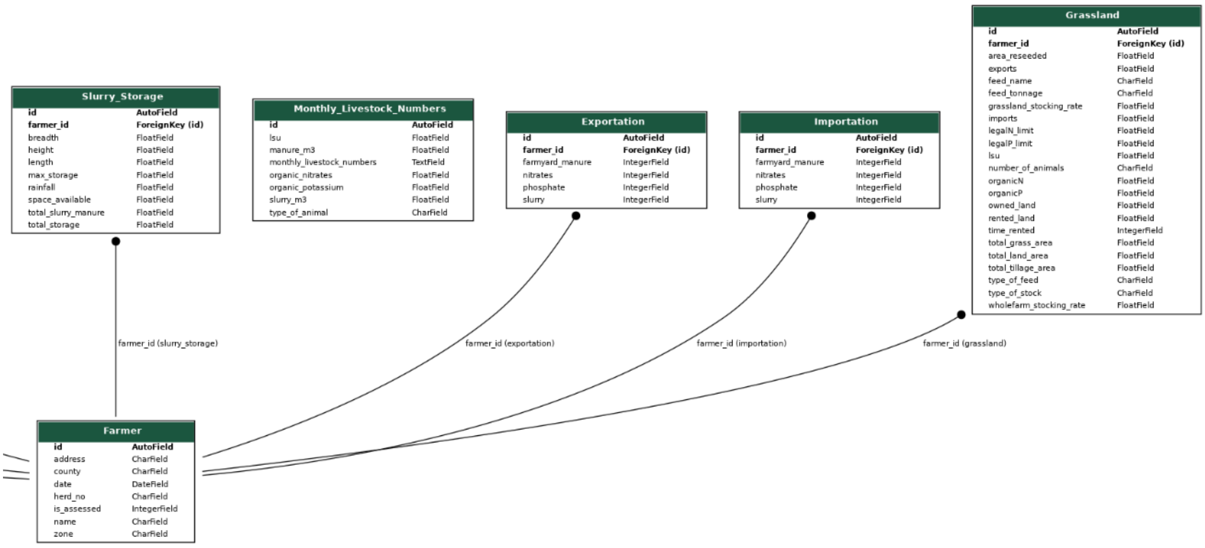

Farmer_Feed
id AutoField
farmer_id ForeignKey (id)
number_beans IntegerField
number_beat_pulp_unmolassed IntegerField
number_beat_pulps_molassed IntegerField
number_citrus_pulp IntegerField
number_compound IntegerField
number_copra IntegerField
number_cotton_seed IntegerField
number_distillers_grain IntegerField
number_lucerne IntegerField
number_maize IntegerField
number_maize_distiller IntegerField
number_maize_germ IntegerField
number_maize_gluten IntegerField
number_oats IntegerField
number_palm_kernel IntegerField
number_peas IntegerField
number_rapeseed IntegerField
number_soya_bean IntegerField
number_soya_hulls IntegerField
number_sunflower IntegerField
number_wheat IntegerField
Feed_Types
id AutoField
farmer_id ForeignKey (id)
feed_name CharField
feed_type CharField
kg_per_tonne_fed FloatField
Farmer_Livestock
id AutoField
farmer_id ForeignKey (id)
number_cattle1 IntegerField
number_cattle2 IntegerField
number_cattle3 IntegerField
number_dairy_cows IntegerField
number_goats IntegerField
number_horse1 IntegerField
number_horse2 IntegerField
number_lowland_ewe IntegerField
number_lowland_hogget IntegerField
number_mountain_ewe IntegerField
number_mountain_hogget IntegerField
number_suckler_cows IntegerField
Farm_Records
id AutoField
balance_under_legal_amount FloatField
balance_under_recommended FloatField
closing_stock CharField
farm_records_max_nitrogen_allowed FloatField
farm_records_max_phospheros_allowed FloatField
farm_records_opening_stock CharField
farm_records_year DateTimeField
fertilizer_bought CharField
fertilizer_sold CharField
import_export_information CharField
total_usage FloatField
Slurry_Storage
id AutoField
farmer_id ForeignKey (id)
breadth FloatField
height FloatField
length FloatField
max_storage FloatField
rainfall FloatField
space_available FloatField
total_slurry_manure FloatField
total_storage FloatField
farmer_id (farmer_feed)
farmer_id (feed_types)
farmer_id (farmer_livestock)
farmer_id (slurry_storage)
Farmer
id AutoField
address CharField
county CharField
date DateField
herd_no CharField
is_assessed IntegerField
name CharField
zone CharField

Figure 3 System Architecture



Slurry_Storage	
id	**AutoField**
farmer_id	**ForeignKey (id)**
breadth	FloatField
height	FloatField
length	FloatField
max_storage	FloatField
rainfall	FloatField
space_available	FloatField
total_slurry_manure	FloatField
total_storage	FloatField

Monthly_Livestock_Numbers	
id	**AutoField**
lsu	FloatField
manure_m3	FloatField
monthly_livestock_numbers	TextField
organic_nitrates	FloatField
organic_potassium	FloatField
slurry_m3	FloatField
type_of_animal	CharField

Exportation	
id	**AutoField**
farmer_id	**ForeignKey (id)**
farmyard_manure	IntegerField
nitrates	IntegerField
phosphate	IntegerField
slurry	IntegerField

Importation	
id	**AutoField**
farmer_id	**ForeignKey (id)**
farmyard_manure	IntegerField
nitrates	IntegerField
phosphate	IntegerField
slurry	IntegerField

Grassland	
id	**AutoField**
farmer_id	**ForeignKey (id)**
area_reseeded	FloatField
exports	FloatField
feed_name	CharField
feed_tonnage	CharField
grassland_stocking_rate	FloatField
imports	FloatField
legalN_limit	FloatField
legalP_limit	FloatField
lsu	FloatField
number_of_animals	CharField
organicN	FloatField
organicP	FloatField
owned_land	FloatField
rented_land	FloatField
time_rented	IntegerField
total_grass_area	FloatField
total_land_area	FloatField
total_tillage_area	FloatField
type_of_feed	CharField
type_of_stock	CharField
wholefarm_stocking_rate	FloatField

Farmer	
id	**AutoField**
address	CharField
county	CharField
date	DateField
herd_no	CharField
is_assessed	IntegerField
name	CharField
zone	CharField

Figure 4: System Architecture

5.0 System Architecture

The project consists of a web application built with Django/Python with a MySQL database. The system will be designed for fast assessments with easy to read reports. Python was chosen with its popularity in mind, most developers will enjoy working with the emphasized readability and simpler syntax.
The system will be used by Advisors to assess Farmers nitrate levels.

Django was the chosen framework as the all batteries included meant that there is most likely a library which would speed up development and provide better functionality.

The MySQL database will allow for easy storage and retrieval of the data and works best with the scenario of having lots of small tables.

The Django/Python web application will be used mainly for the purpose of completing assessments based on farmer data provided.

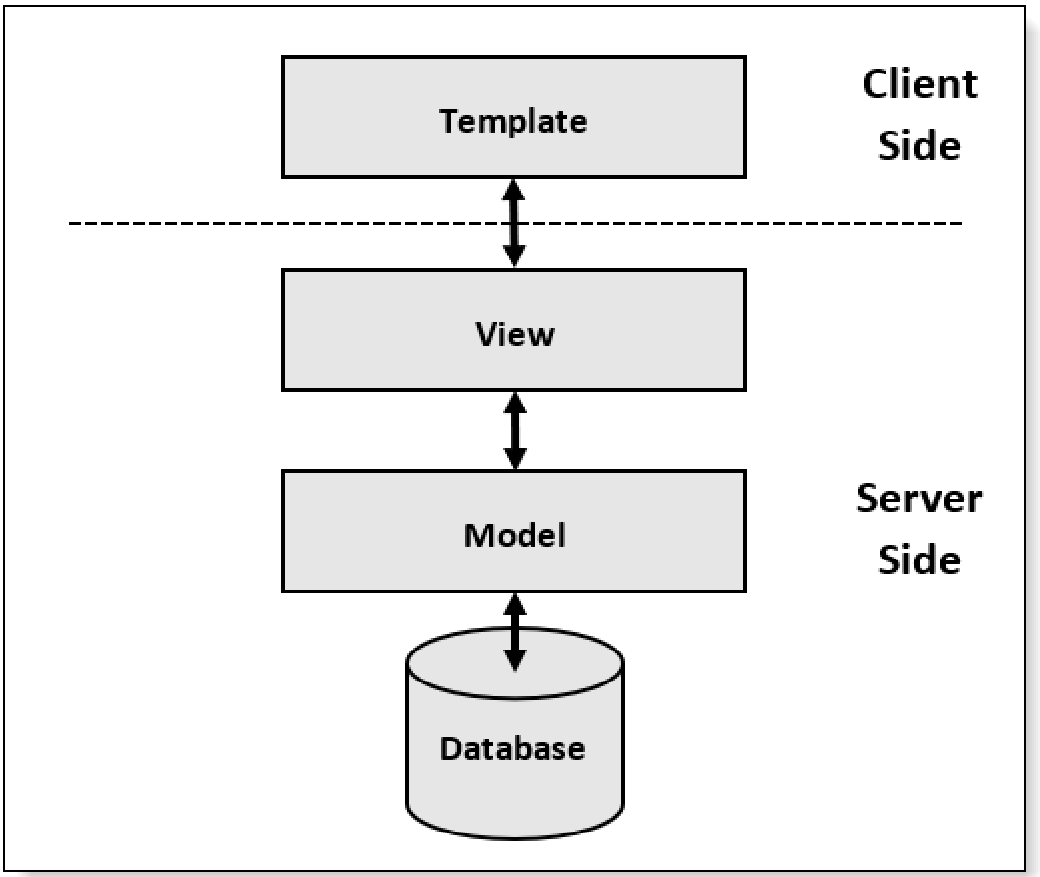
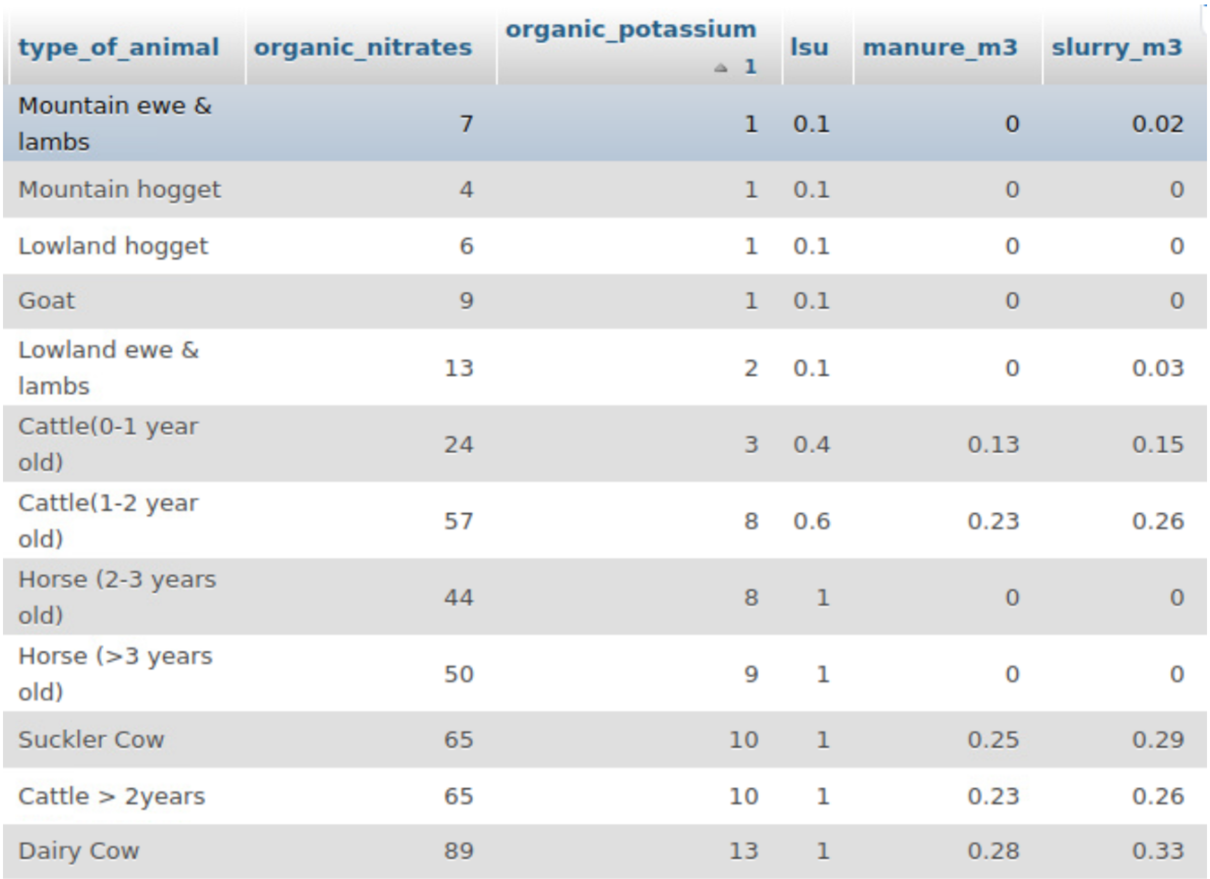


Figure 5: System Architecture{1}

The Application will feature a number of views (screens) with which an Advisor can interact with, and a models section specifying the type of each specified input similar to a class. The way the system works with Django is by separating the logic from the actual aesthetics of the application development becomes easier. The Template section will be what the Advisor will see, the view is where the code logic of that template is stored. The models section specifies the tables and types of data to be stored in the database.

6.0 Database

The MySQL Database will store all the input data from the advisor. This data will include sensitive information about the farmer, relative to the Assessment. The Database will also contain some of the figures required for calculating stocking rates and amount of farm produce.
This information is stored in the Monthly Livestock Numbers table.



type_of_animal	organic_nitrates	organic_potassium ▲ 1	lsu	manure_m3	slurry_m3
Mountain ewe & lambs	7	1	0.1	0	0.02
Mountain hogget	4	1	0.1	0	0
Lowland hogget	6	1	0.1	0	0
Goat	9	1	0.1	0	0
Lowland ewe & lambs	13	2	0.1	0	0.03
Cattle(0-1 year old)	24	3	0.4	0.13	0.15
Cattle(1-2 year old)	57	8	0.6	0.23	0.26
Horse (2-3 years old)	44	8	1	0	0
Horse (>3 years old)	50	9	1	0	0
Suckler Cow	65	10	1	0.25	0.29
Cattle > 2years	65	10	1	0.23	0.26
Dairy Cow	89	13	1	0.28	0.33

Figure 6: Monthly Livestock Numbers

7.0 User Interfaces

This section will illustrate the pages available to the Advisors through the application. These screens will feature a short explanation of what's on the screen and what is happening in the background. The first screen available to an Advisor is the login screen.

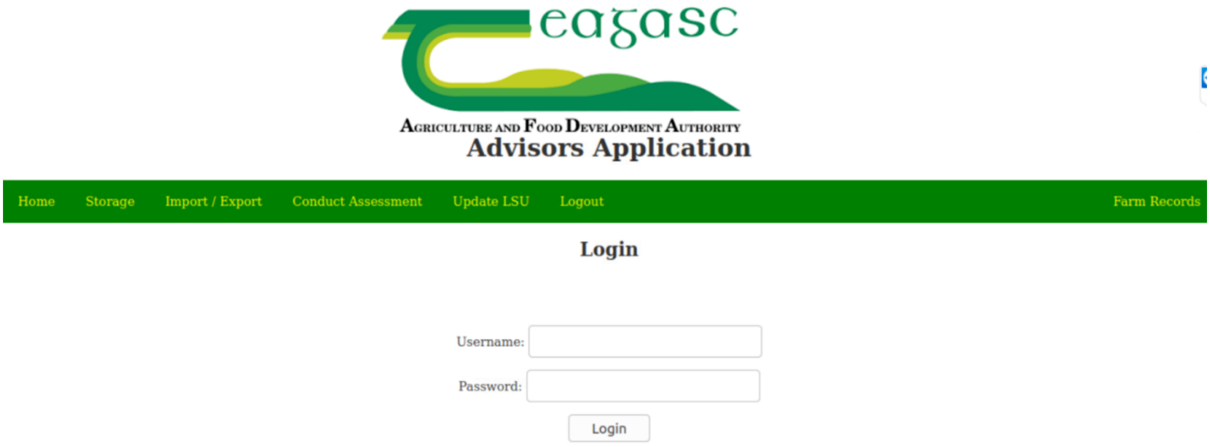
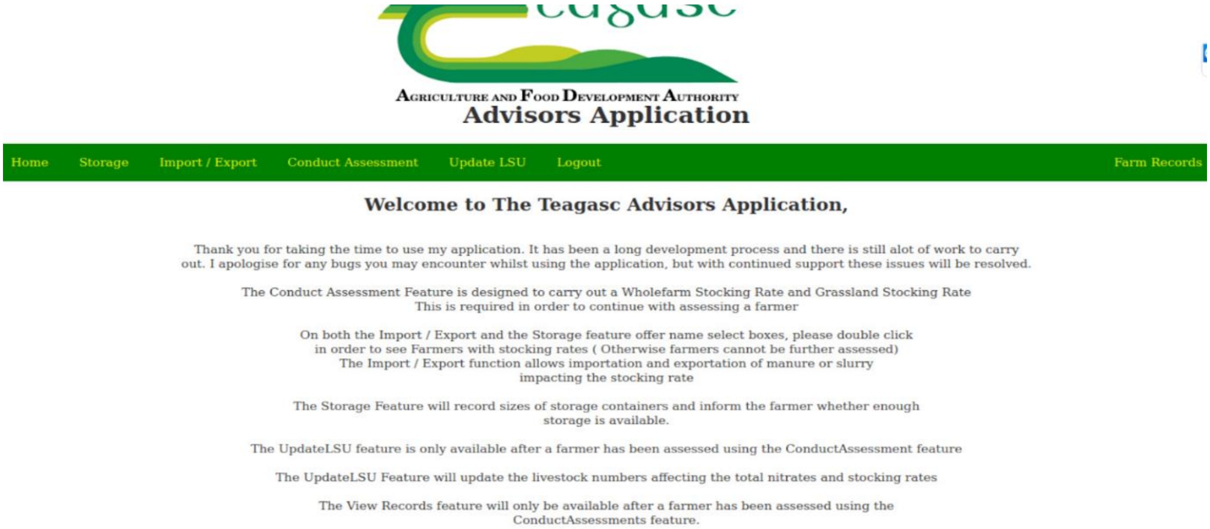


Figure 7: Login Screen

Upon successful login an Advisor will be redirected to the home screen, in the case of a failed login the Advisor will remain at the login page.



Welcome to The Teagasc Advisors Application,

Thank you for taking the time to use my application. It has been a long development process and there is still alot of work to carry out. I apologise for any bugs you may encounter whilst using the application, but with continued support these issues will be resolved.

The Conduct Assessment Feature is designed to carry out a Wholefarm Stocking Rate and Grassland Stocking Rate
This is required in order to continue with assessing a farmer

On both the Import / Export and the Storage feature offer name select boxes, please double click
in order to see Farmers with stocking rates (Otherwise farmers cannot be further assessed)
The Import / Export function allows importation and exportation of manure or slurry
impacting the stocking rate

The Storage Feature will record sizes of storage containers and inform the farmer whether enough storage is available.

The UpdateLSU feature is only available after a farmer has been assessed using the ConductAssessment feature

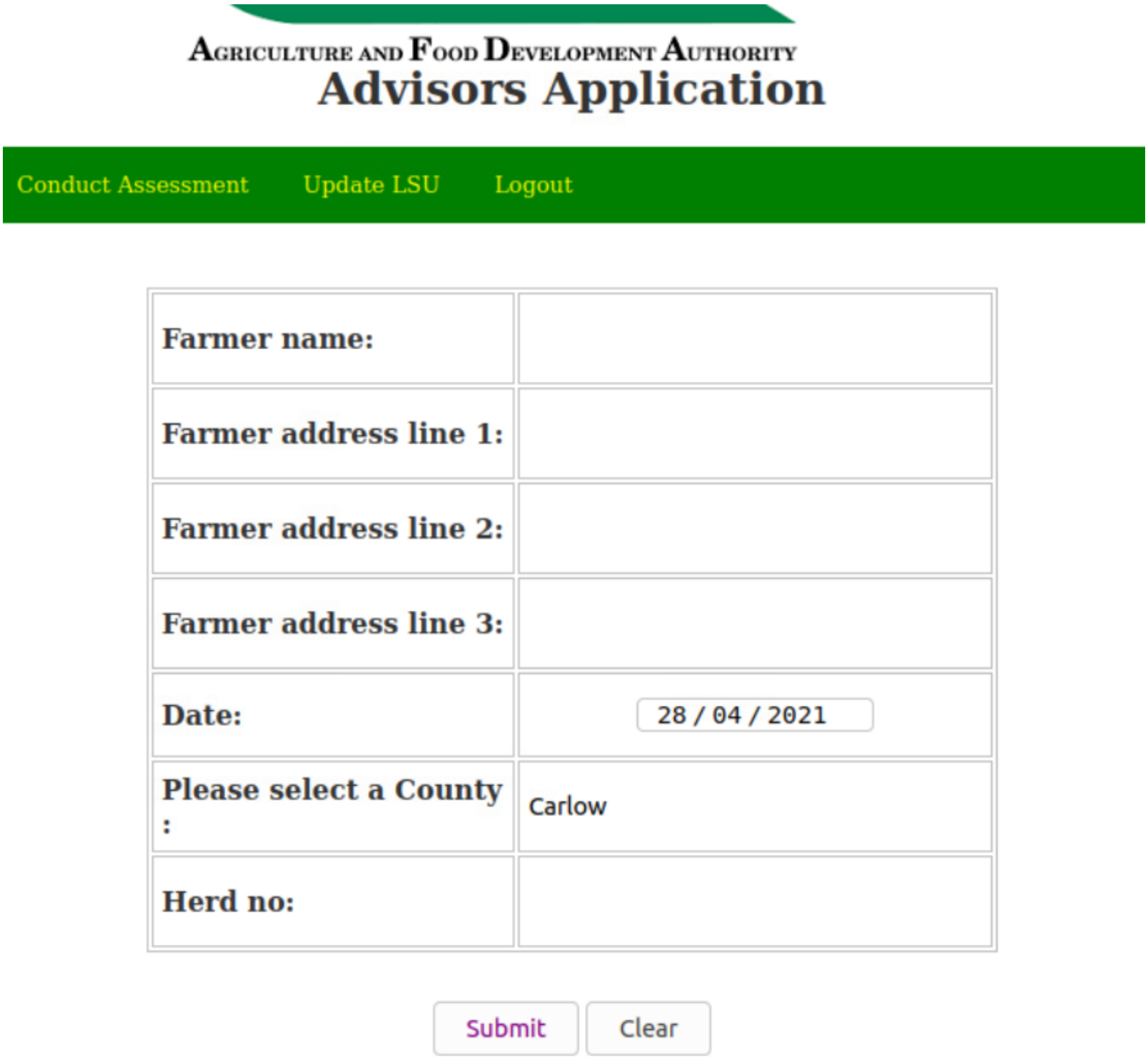
The UpdateLSU Feature will update the livestock numbers affecting the total nitrates and stocking rates

The View Records feature will only be available after a farmer has been assessed using the ConductAssessments feature.

Figure 8: Home Screen

7.1 ConductAssessment

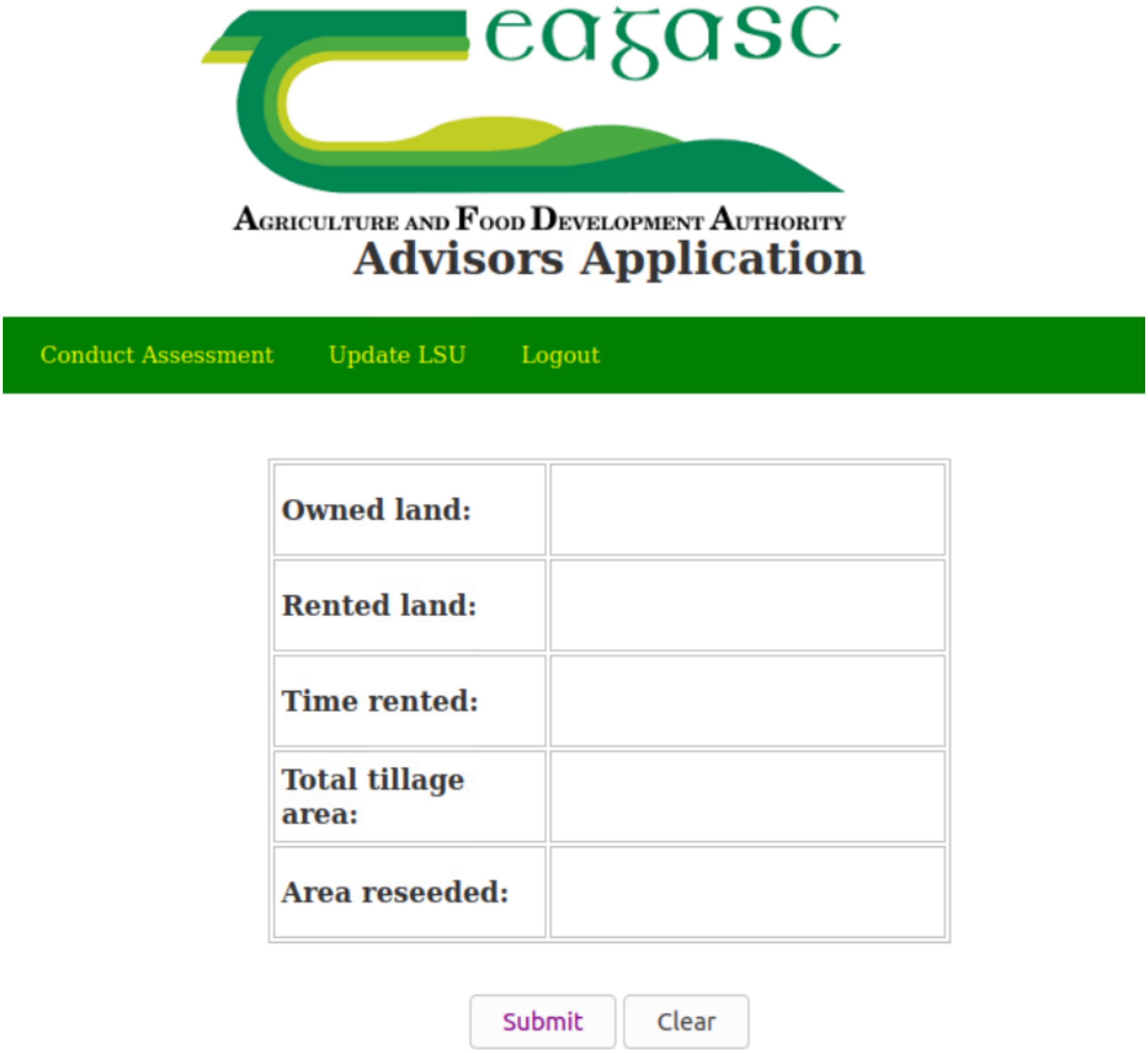
From the home screen an Advisor can access the features listed in the navigation bar. The application is designed to assess farmers nitrates levels and on farm produce. In order to be able to accurately assess the farmer must possess both a grassland stocking rate and a whole farm stocking rate. For this to happen the first feature an Advisor should use is the ConductGrassland feature which consists of 3 forms and a report page.



Farmer name:	
Farmer address line 1:	
Farmer address line 2:	
Farmer address line 3:	
Date:	28 / 04 / 2021
Please select a County :	Carlow
Herd no:	

Figure 9: Conduct Assessment 1

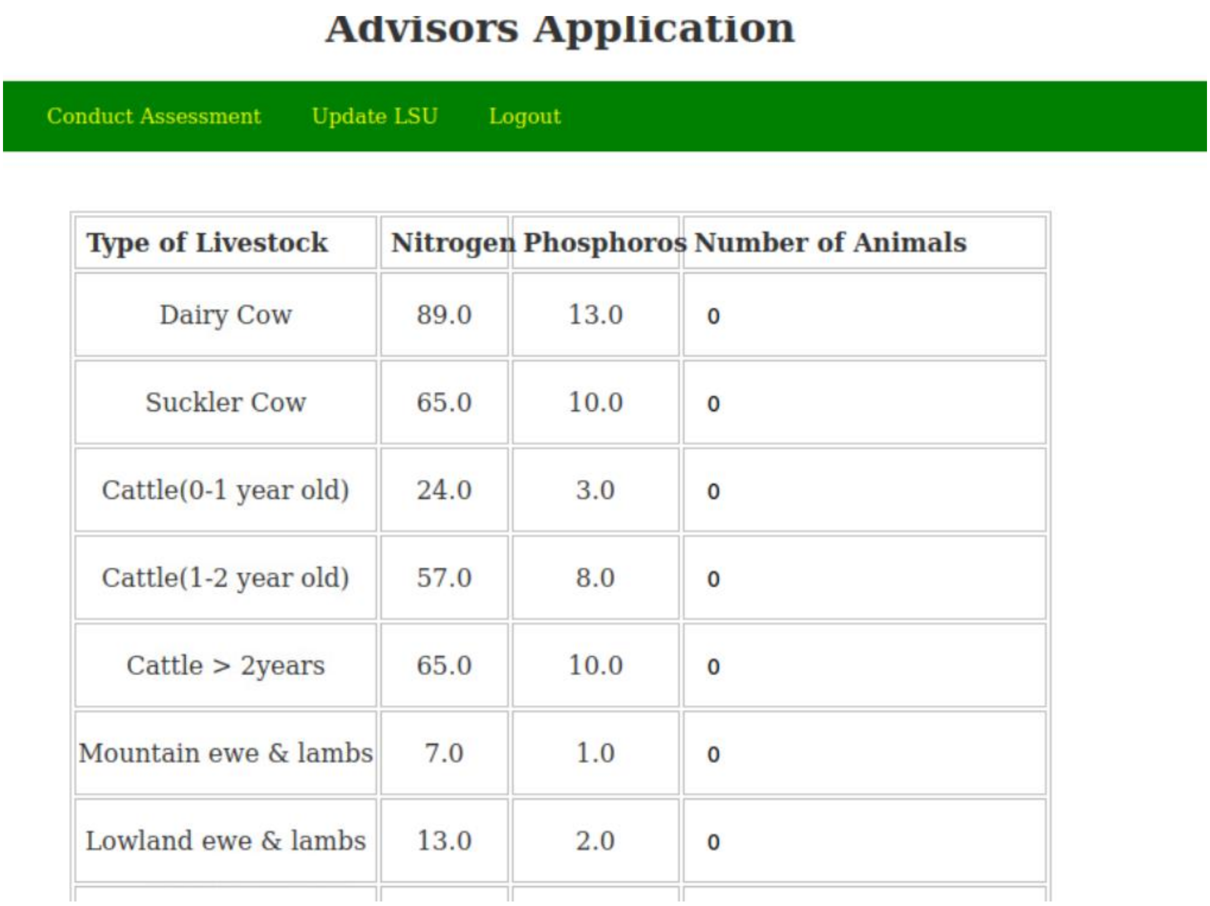
The ConductAssessment first page will record the farmers personal information, the county option is a multiple choice field where an Advisor can select the relevant county. The county will determine the storage period and predicted rainfall, this will affect later calculations. The herd number should be unique to each farmer. If the Advisor enters invalid information the application will highlight the invalid field in a red border which will remain till the issue is resolved by the Advisor. On completion the Advisor will then fill out the second page of the form.



Owned land:	
Rented land:	
Time rented:	
Total tillage area:	
Area reseeded:	

Figure 10: Conduct assessment 2

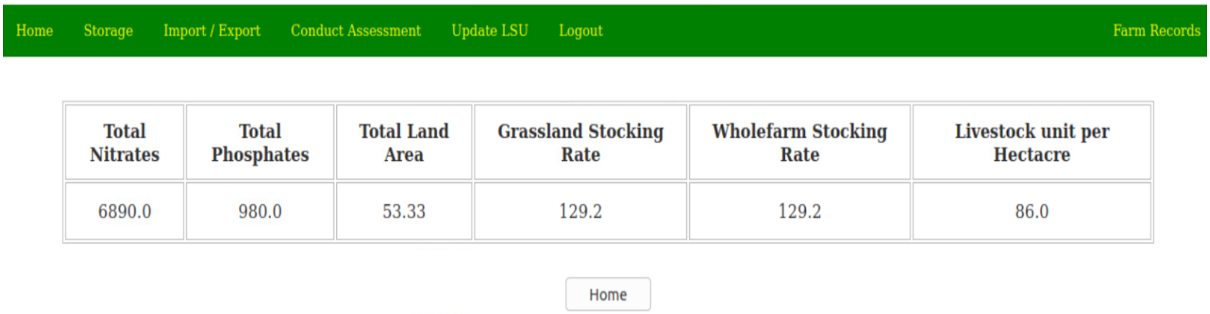
ConductAssessment2 will record the land information of the farmer, these values are extremely important in calculating the whole farm stocking rate, record5 and grass stocking rate. The next form will record the livestock information from the farmer, with the amount of land recorded the Livestock Unit per Hectare can also be calculated.



Type of Livestock	Nitrogen	Phosphoros	Number of Animals
Dairy Cow	89.0	13.0	0
Suckler Cow	65.0	10.0	0
Cattle(0-1 year old)	24.0	3.0	0
Cattle(1-2 year old)	57.0	8.0	0
Cattle > 2years	65.0	10.0	0
Mountain ewe & lambs	7.0	1.0	0
Lowland ewe & lambs	13.0	2.0	0

figure 6.3 ConductAssessment 3

After successfully recording this information a report will be displayed to the Advisor.

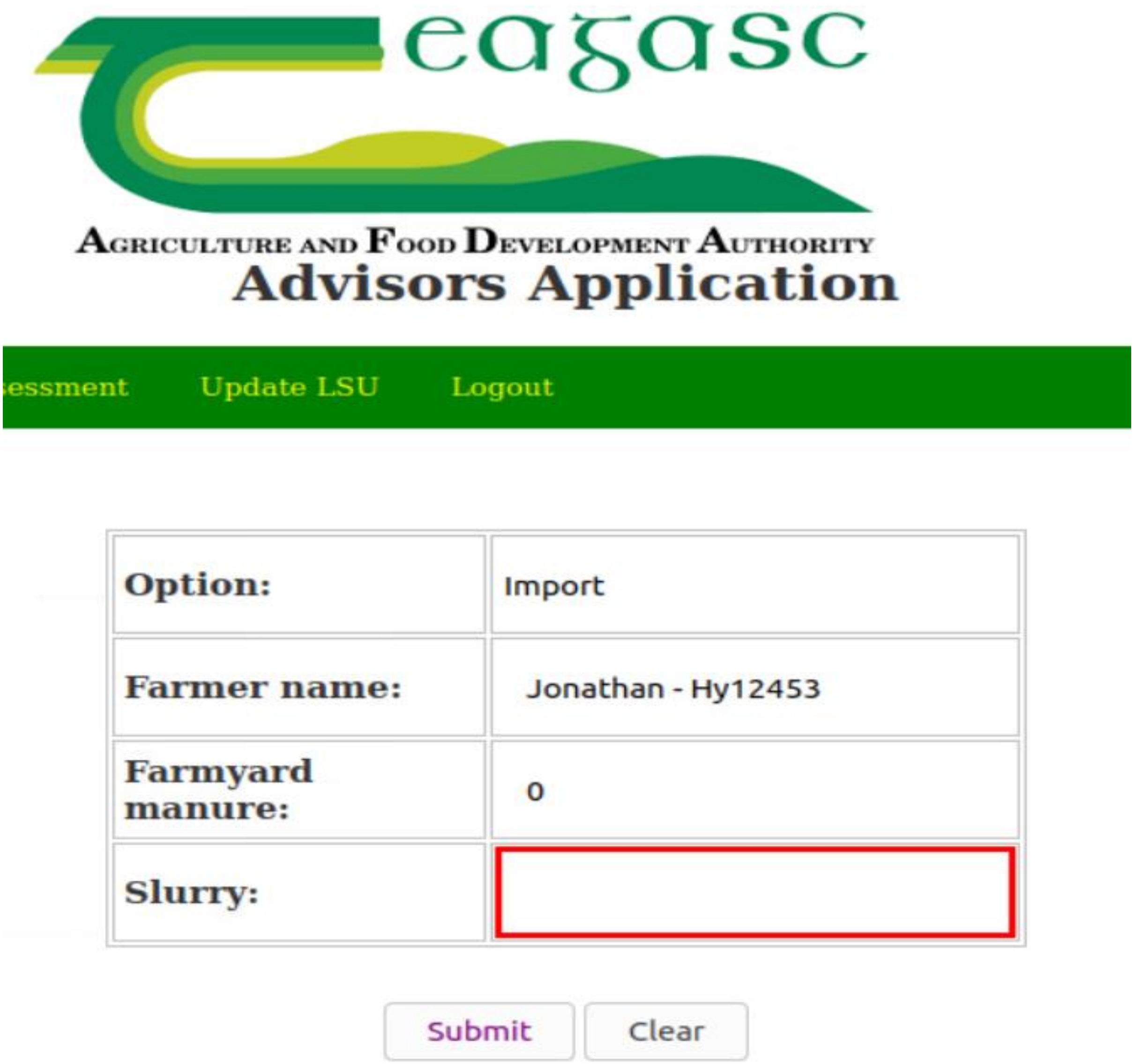


Total Nitrates	Total Phosphates	Total Land Area	Grassland Stocking Rate	Wholefarm Stocking Rate	Livestock unit per Hectacre
6890.0	980.0	53.33	129.2	129.2	86.0

figure 6.4 ConductAssessmentReport

7.2 Import / Export Nitrates

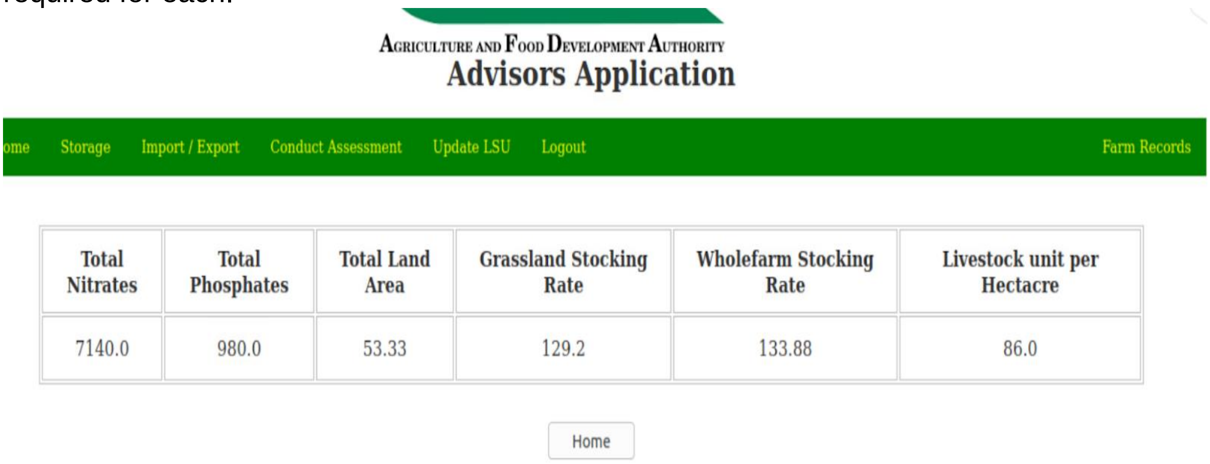
The Import Export feature is a small form consisting of 4 fields, import or export, farmer name, manure and slurry. A farmer will import or export depending on their stocking rate. Some farmers' stocking rate will reach a point far beyond a legal limit, in this scenario an export would be required to reduce the stocking rate. With further development the application will be more informative with reports whilst also offering advice on maintaining a healthy whole farm stocking rate. In the below image the field with no data is invalid and the application highlights this.



Option:	Import
Farmer name:	Jonathan - Hy12453
Farmyard manure:	0
Slurry:	

figure 7.1 Import Export

After completing this page a report will be generated depending on whether an import or export occurred and depending on the content, manure or slurry, different calculations are required for each.

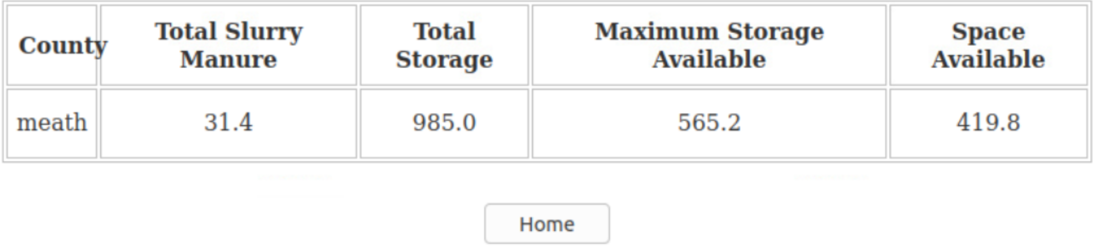


Total Nitrates	Total Phosphates	Total Land Area	Grassland Stocking Rate	Wholefarm Stocking Rate	Livestock unit per Hectacre
7140.0	980.0	53.33	129.2	133.88	86.0

figure 7.2 Import Export Report

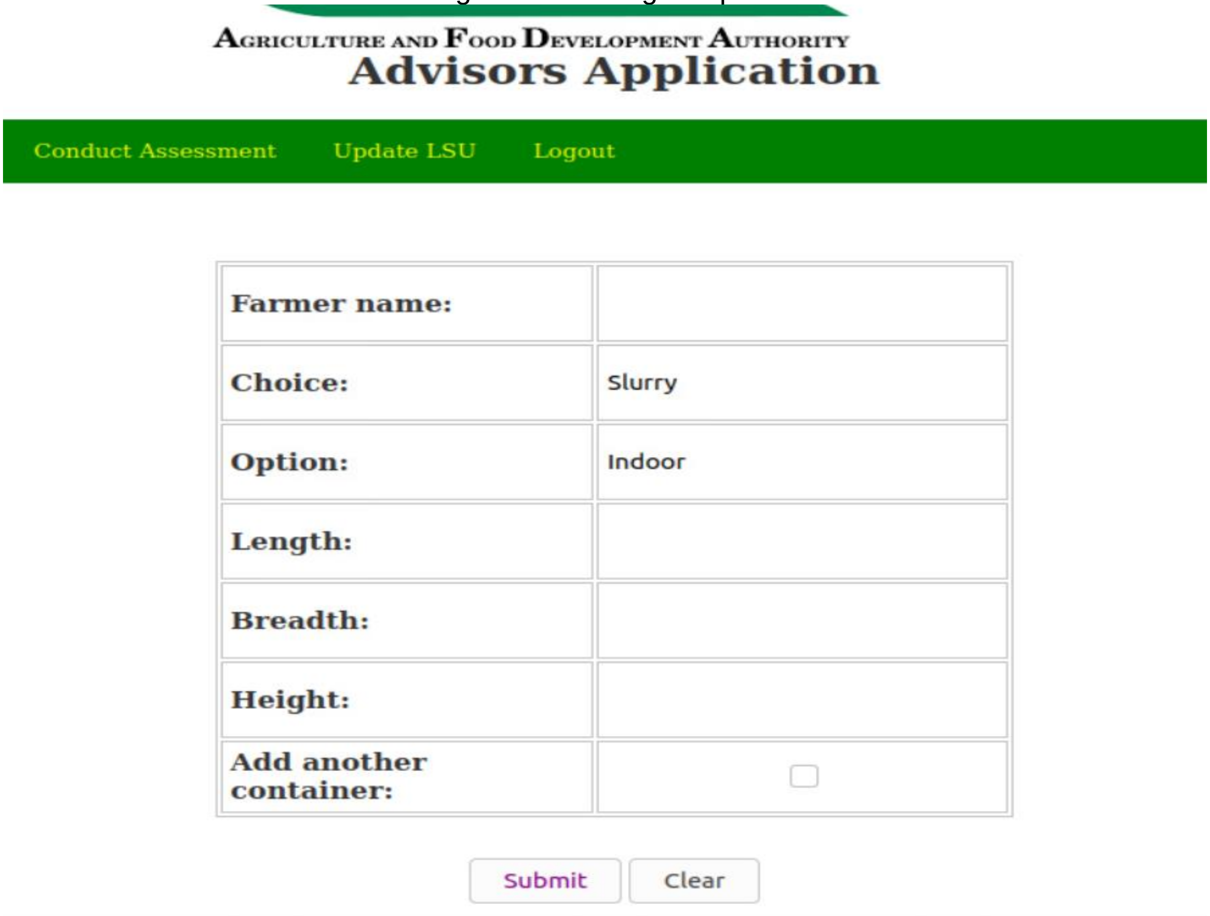
7.3 Storage

The storage feature will be used to record the storage facilities owned by the farmer. These storage facilities vary, and calculating whether a farmer is legally within their limits on a farm depends on the dimensions, type and location of the tank. Each different county has a different storage period and a predicted rainfall. If the tank is outdoors then the rainfall must be taken into consideration as this will affect the contents and the space available in the storage facilities. This feature is responsible for making all these calculations and displaying a clear report at the end readable by the farmer. This form also offers the option of adding another container, which if left unchecked will return the report. In this demo there is no need to add another container.



County	Total Slurry Manure	Total Storage	Maximum Storage Available	Space Available
meath	31.4	985.0	565.2	419.8

figure 8.1 Storage Report

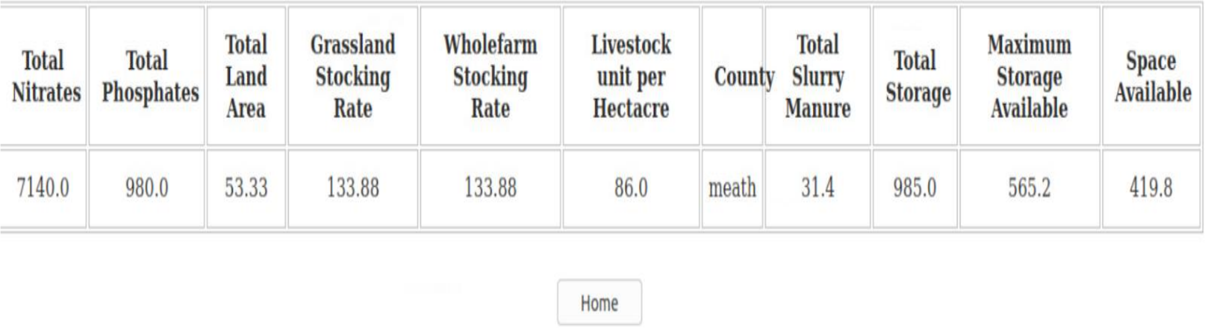


Farmer name:	
Choice:	Slurry
Option:	Indoor
Length:	
Breadth:	
Height:	
Add another container:	☐

figure 8.2 Add Storage Container

7.4 Farm Records

This is a one page report which displays the reports from the stocking rate calculations and the storage information recorded.

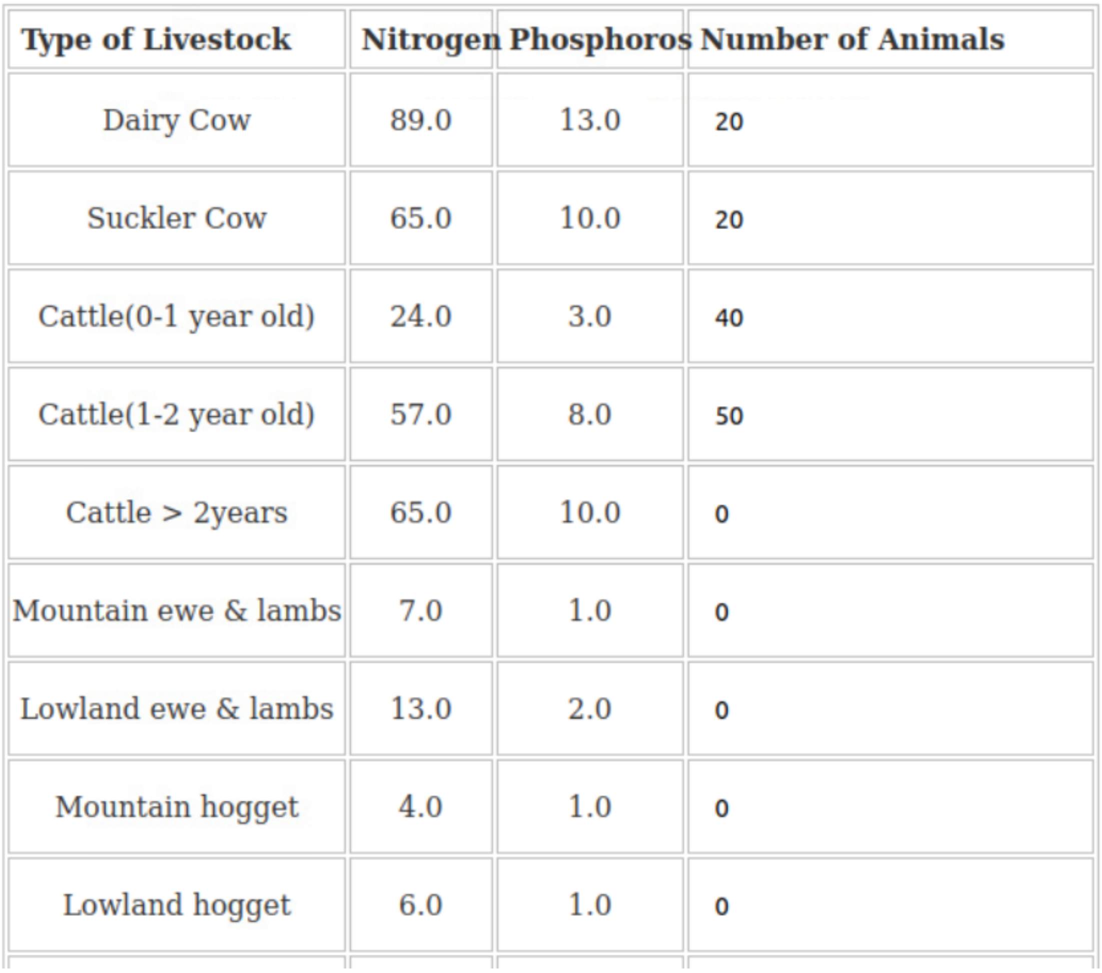


Total Nitrates	Total Phosphates	Total Land Area	Grassland Stocking Rate	Wholefarm Stocking Rate	Livestock unit per Hectacre	County	Total Slurry Manure	Total Storage	Maximum Storage Available	Space Available
7140.0	980.0	53.33	133.88	133.88	86.0	meath	31.4	985.0	565.2	419.8

figure 9.1 Farm Records Report

7.5 Update LSU (Livestock unit)

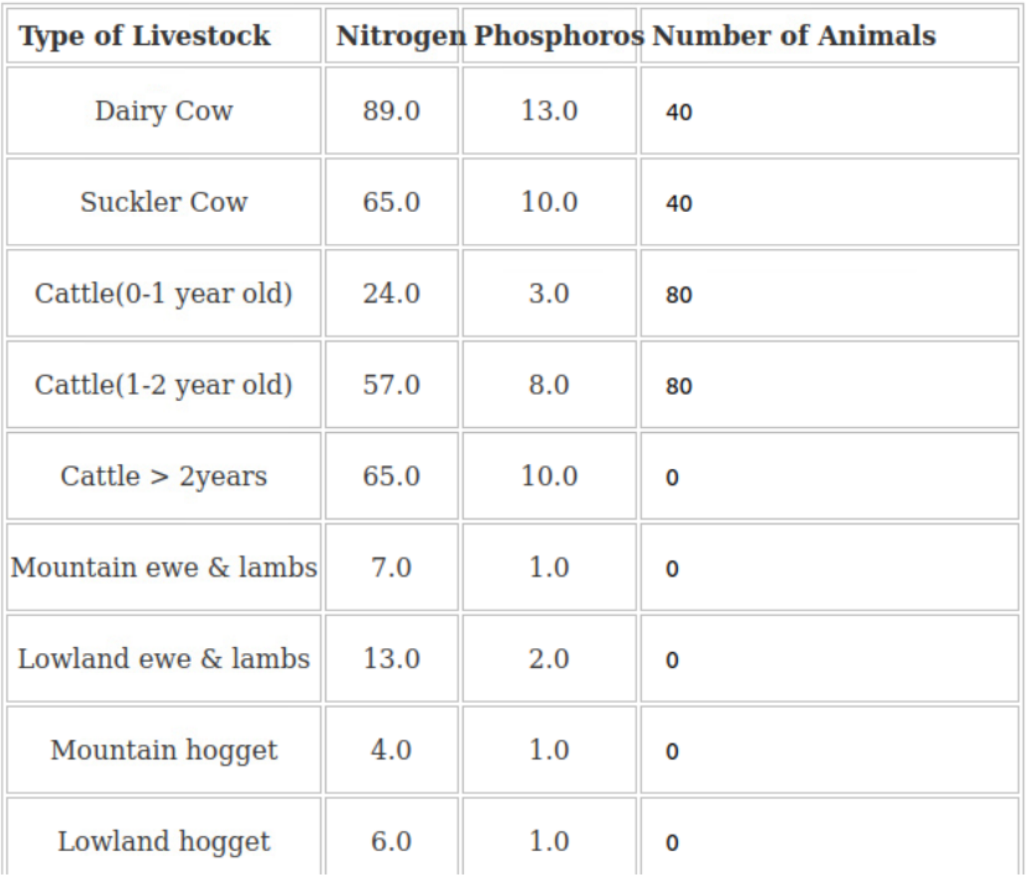
The Update LSU feature includes the livestock table from the conductAssessment feature with the values being replaced with the current farmers quantity of animals. If these numbers increase or reduce the Livestock unit per hectare value will change. Livestock unit represents the amount of excrement produced by the animal. This value is necessary for determining if a farmer holds too many animals in too small an area and to ensure remaining within the legal limits.



Type of Livestock	Nitrogen	Phosphoros	Number of Animals
Dairy Cow	89.0	13.0	20
Suckler Cow	65.0	10.0	20
Cattle(0-1 year old)	24.0	3.0	40
Cattle(1-2 year old)	57.0	8.0	50
Cattle > 2years	65.0	10.0	0
Mountain ewe & lambs	7.0	1.0	0
Lowland ewe & lambs	13.0	2.0	0
Mountain hogget	4.0	1.0	0
Lowland hogget	6.0	1.0	0

figure 10.1 Update Livestock Unit

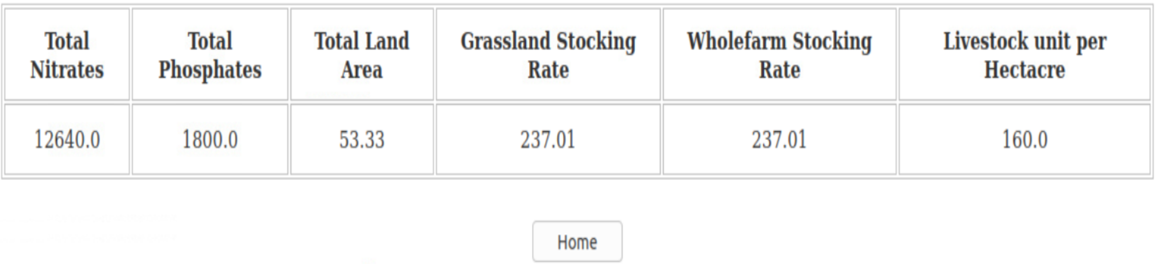
On the previous page the Livestock unit per hectare is 86 and the total nitrates is 7140, after updating this table with new figures increasing the number of Dairy Cows, Suckler Cows and other cattle brackets these figures should increase.



Type of Livestock	Nitrogen	Phosphoros	Number of Animals
Dairy Cow	89.0	13.0	40
Suckler Cow	65.0	10.0	40
Cattle(0-1 year old)	24.0	3.0	80
Cattle(1-2 year old)	57.0	8.0	80
Cattle > 2years	65.0	10.0	0
Mountain ewe & lambs	7.0	1.0	0
Lowland ewe & lambs	13.0	2.0	0
Mountain hogget	4.0	1.0	0
Lowland hogget	6.0	1.0	0

figure 10.2 Update Livestock Unit (increased figures)

And the report to verify success, increased units per hectare and increased nitrates.



Total Nitrates	Total Phosphates	Total Land Area	Grassland Stocking Rate	Wholefarm Stocking Rate	Livestock unit per Hectacre
12640.0	1800.0	53.33	237.01	237.01	160.0

figure 10.3 Update Livestock Unit (increased figures)

8.0 Database Design (there is a django generated model in diagram folder)

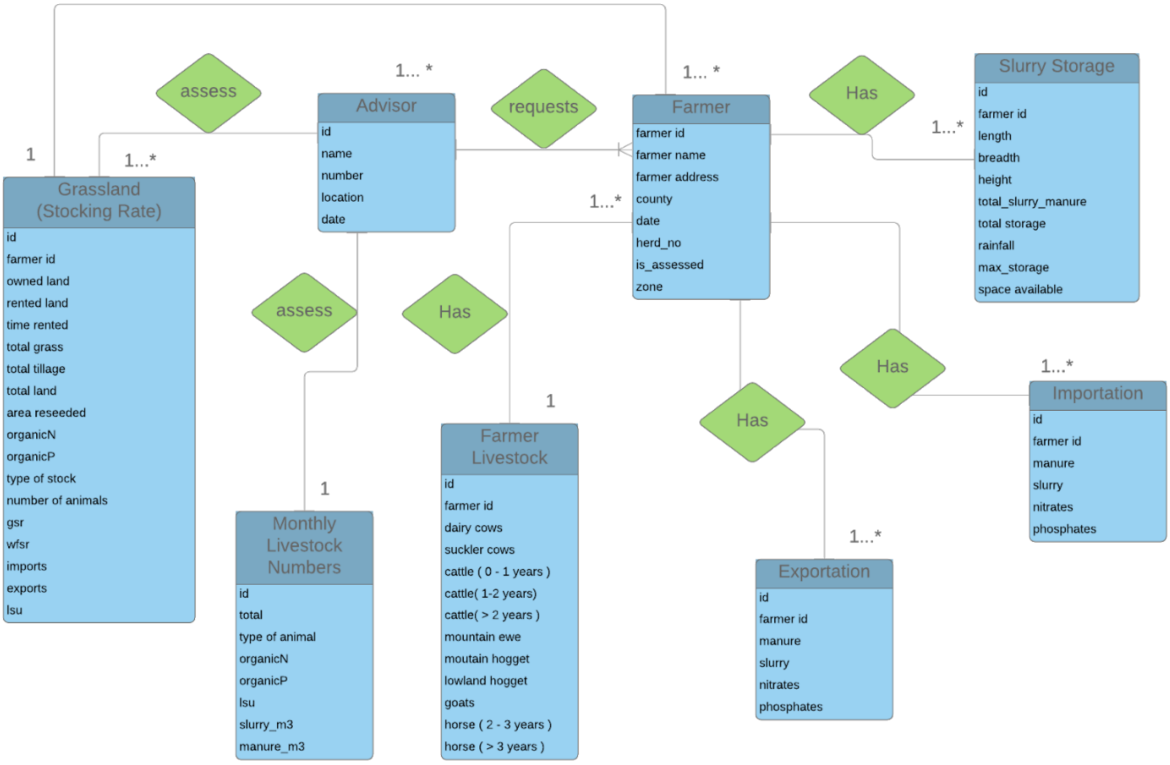


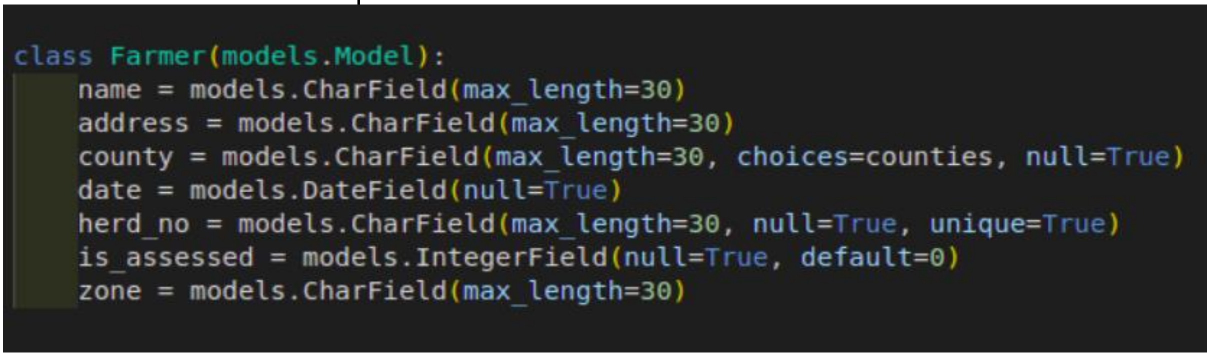
Figure 11.1 ER Diagram

9.0 Database Tables

As explained in the System Architecture this application was built using Django and to keep the logic separate from the visual side of the application Django uses a MVC controller. Django models are the tools used to work with data and databases, these tools minimise complications expected with handling communications between the database and the users. This next section will display screenshots of the models in the code base, and how the database will look.

9.1 Farmer Model

This model will contain the personal information associated with the Farmer.

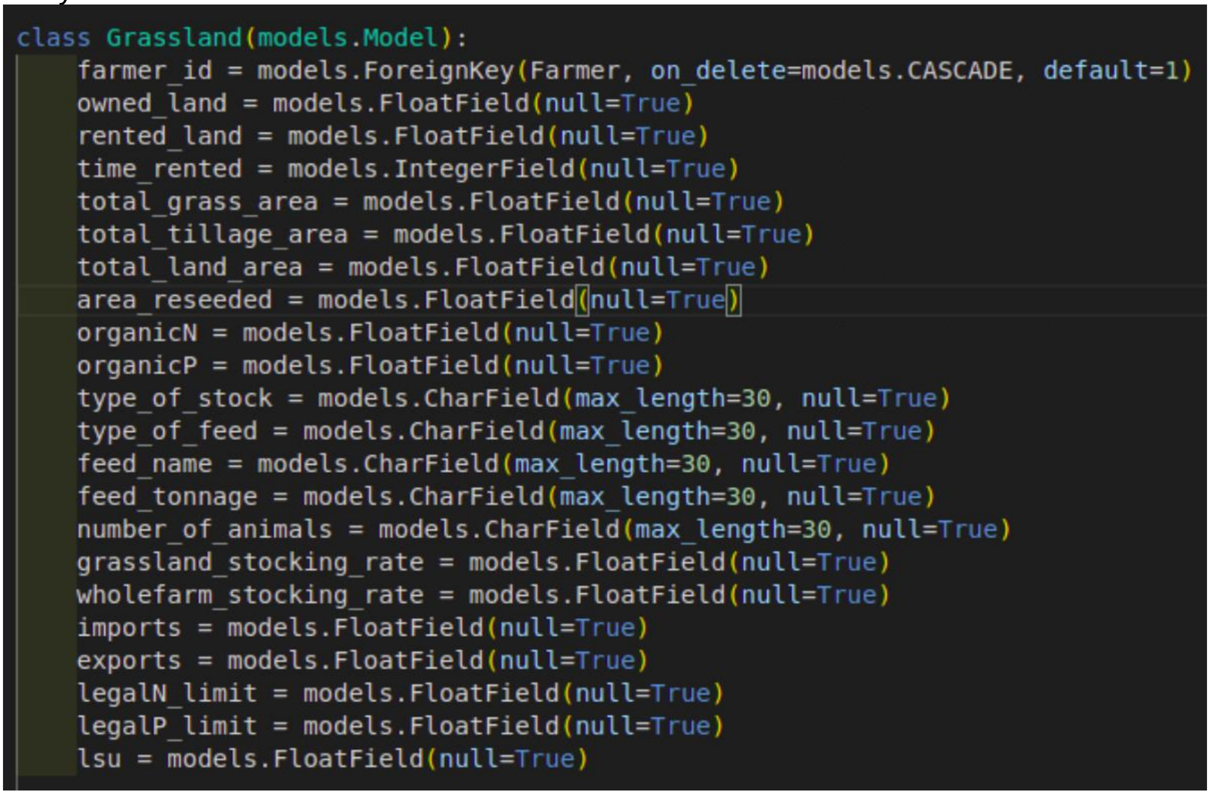


```
class Farmer(models.Model):
    name = models.CharField(max_length=30)
    address = models.CharField(max_length=30)
    county = models.CharField(max_length=30, choices=counties, null=True)
    date = models.DateField(null=True)
    herd_no = models.CharField(max_length=30, null=True, unique=True)
    is_assessed = models.IntegerField(null=True, default=0)
    zone = models.CharField(max_length=30)
```

Figure 12.1 Farmer Model

9.2 Grassland Model

This model will contain the information relating to the land, please understand this model will be changing, currently there are too many attributes associated which will be separated into smaller tables but due to the application moving in a new direction this refactoring has been delayed.

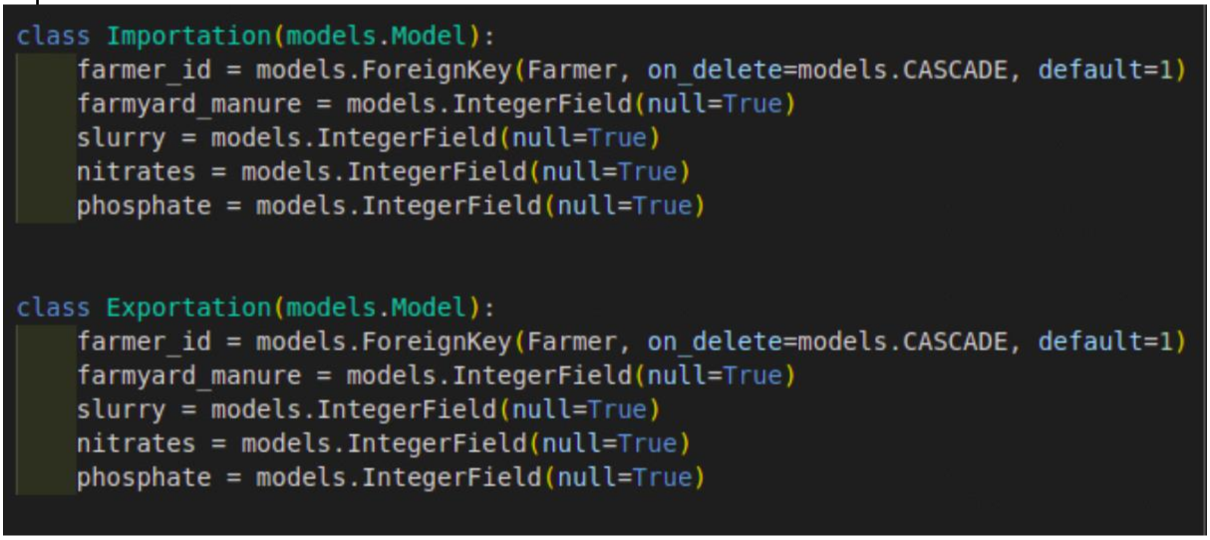


```
class Grassland(models.Model):
    farmer_id = models.ForeignKey(Farmer, on_delete=models.CASCADE, default=1)
    owned_land = models.FloatField(null=True)
    rented_land = models.FloatField(null=True)
    time_rented = models.IntegerField(null=True)
    total_grass_area = models.FloatField(null=True)
    total_tillage_area = models.FloatField(null=True)
    total_land_area = models.FloatField(null=True)
    area_reseeded = models.FloatField(null=True)
    organicN = models.FloatField(null=True)
    organicP = models.FloatField(null=True)
    type_of_stock = models.CharField(max_length=30, null=True)
    type_of_feed = models.CharField(max_length=30, null=True)
    feed_name = models.CharField(max_length=30, null=True)
    feed_tonnage = models.CharField(max_length=30, null=True)
    number_of_animals = models.CharField(max_length=30, null=True)
    grassland_stocking_rate = models.FloatField(null=True)
    wholefarm_stocking_rate = models.FloatField(null=True)
    imports = models.FloatField(null=True)
    exports = models.FloatField(null=True)
    legalN_limit = models.FloatField(null=True)
    legalP_limit = models.FloatField(null=True)
    lsu = models.FloatField(null=True)
```

Figure 12.2 Grassland Model

9.3 Importation and Exportation Models

These models are similar, containing the same attributes depending on whether an import or export dictates which of these models will be used.



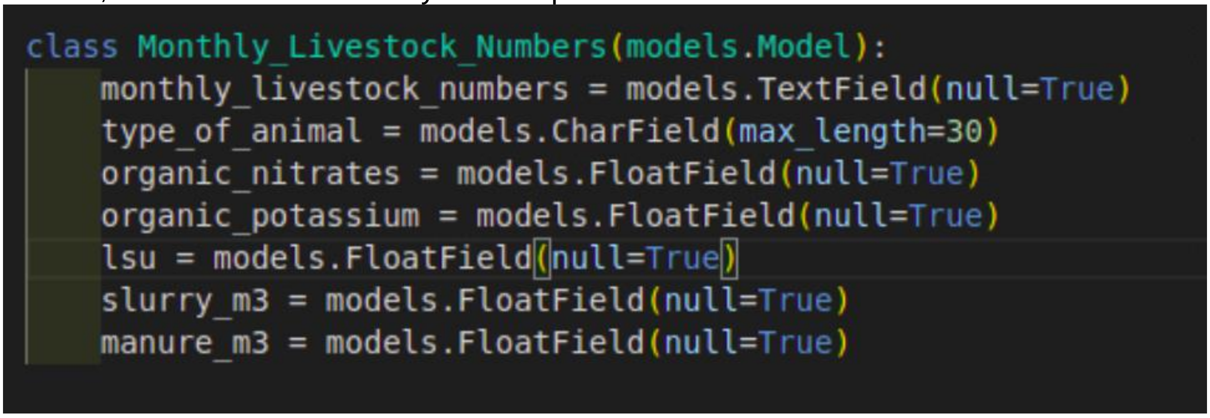
```
class Importation(models.Model):
    farmer_id = models.ForeignKey(Farmer, on_delete=models.CASCADE, default=1)
    farmyard_manure = models.IntegerField(null=True)
    slurry = models.IntegerField(null=True)
    nitrates = models.IntegerField(null=True)
    phosphate = models.IntegerField(null=True)


class Exportation(models.Model):
    farmer_id = models.ForeignKey(Farmer, on_delete=models.CASCADE, default=1)
    farmyard_manure = models.IntegerField(null=True)
    slurry = models.IntegerField(null=True)
    nitrates = models.IntegerField(null=True)
    phosphate = models.IntegerField(null=True)
```

Figure 13.1 Import Export Models

9.4 Monthly Livestock Numbers

This model is extremely important as this contains important information in order to calculate nitrates, livestock units and slurry/manure produced on farms.

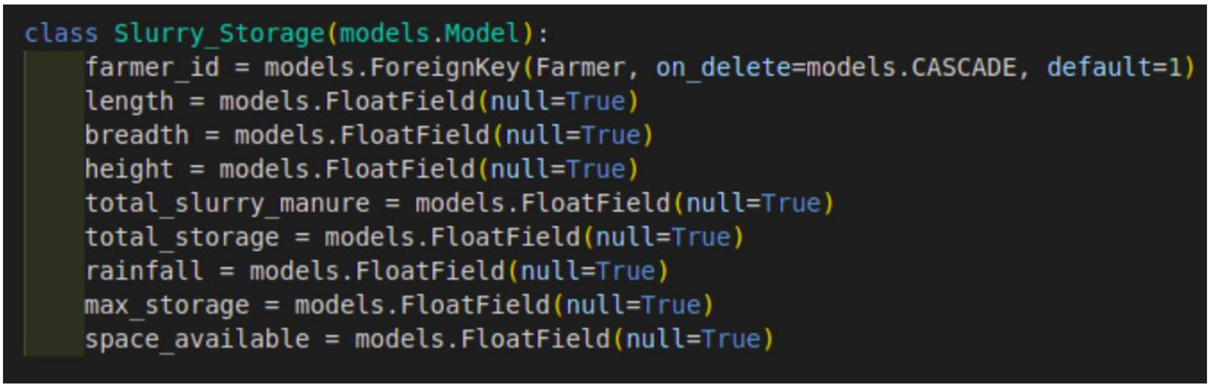


```
class Monthly_Livestock_Numbers(models.Model):
    monthly_livestock_numbers = models.TextField(null=True)
    type_of_animal = models.CharField(max_length=30)
    organic_nitrates = models.FloatField(null=True)
    organic_potassium = models.FloatField(null=True)
    lsu = models.FloatField(null=True)
    slurry_m3 = models.FloatField(null=True)
    manure_m3 = models.FloatField(null=True)
```

Figure 14.1 Monthly Livestock Unit Model

9.5 Slurry Storage

This model will hold the information regarding the measurements of each storage container, the legal maximum amount of storage. These figures are explained in the Add Storage Container section above.

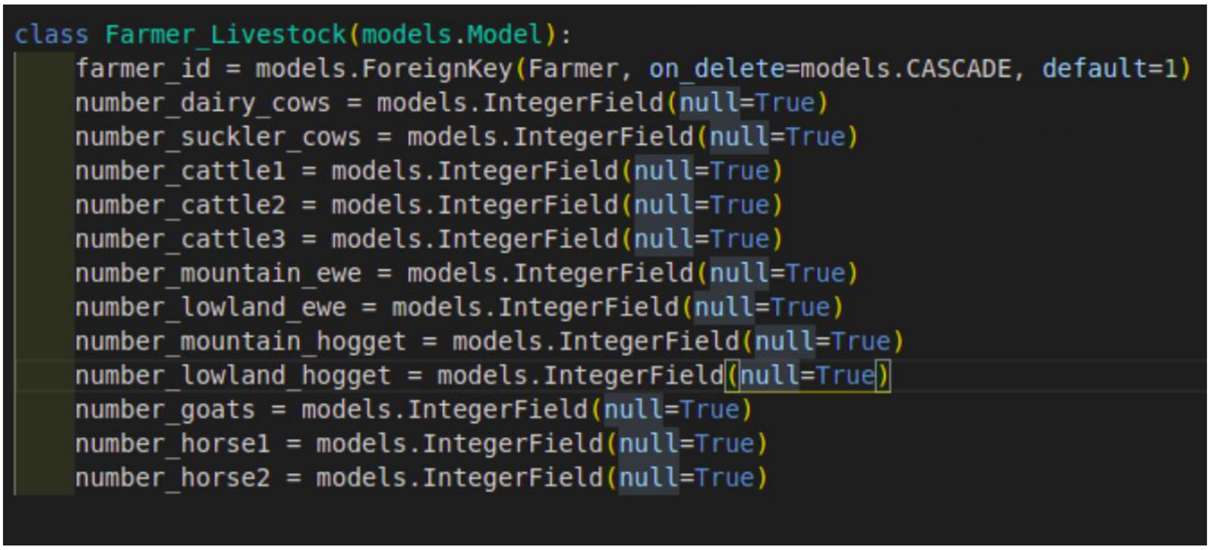


```
class Slurry_Storage(models.Model):
    farmer_id = models.ForeignKey(Farmer, on_delete=models.CASCADE, default=1)
    length = models.FloatField(null=True)
    breadth = models.FloatField(null=True)
    height = models.FloatField(null=True)
    total_slurry_manure = models.FloatField(null=True)
    total_storage = models.FloatField(null=True)
    rainfall = models.FloatField(null=True)
    max_storage = models.FloatField(null=True)
    space_available = models.FloatField(null=True)
```

Figure 15.1 Monthly Livestock Unit Model

9.6 Farmer Livestock

This model will hold the numbers of each breed of livestock owned by a farmer. These table results will be used when calculating the Total Nitrates, produced on farm and livestock units.



```
class Farmer_Livestock(models.Model):
    farmer_id = models.ForeignKey(Farmer, on_delete=models.CASCADE, default=1)
    number_dairy_cows = models.IntegerField(null=True)
    number_suckler_cows = models.IntegerField(null=True)
    number_cattle1 = models.IntegerField(null=True)
    number_cattle2 = models.IntegerField(null=True)
    number_cattle3 = models.IntegerField(null=True)
    number_mountain_ewe = models.IntegerField(null=True)
    number_lowland_ewe = models.IntegerField(null=True)
    number_mountain_hogget = models.IntegerField(null=True)
    number_lowland_hogget = models.IntegerField(null=True)
    number_goats = models.IntegerField(null=True)
    number_horse1 = models.IntegerField(null=True)
    number_horse2 = models.IntegerField(null=True)
```

Figure 15.1 Monthly Livestock Unit Model

10.0 Technologies

10.1Django

Django is a full batteries included framework and was chosen due this. Django offers a very secure framework and due to this application being designed for live use this was a huge benefit. Django offers a (MVC) framework which allows developers to change the look of the application without affecting the logic. This is necessary as there was no specification for the design and aesthetics of this project so they will undoubtedly change. Django also does not restrict the type of database used with the application with support for MySQL, PostgreSQL, SQLite and Oracle. The model aspect of django architecture made it easier to build each class and correctly assign attributes whilst all remaining separate from other functionalities. Django also contains several applications which can be plugged into the system for extra functionality with little work. [2]

10.2 MySQL

The database chosen for this application is MySQL as it works well with large numbers of tables and offers comprehensive application development meaning it was easy to incorporate into the application. MySQL offers high availability, tremendously fast data insert capability and a high performance query engine. With this project's goal of assessing farmers sensitive information will be stored, MySQL's strong data protection offering safe security features will offer peace of mind making it a great candidate for this project.

10.3 Python

Python's popularity, support, versatility and its easy syntax made it a viable option for this project as studies and research take focus and having an easier language to work with simplified development. This application is designed to be deployed as a webapp, something Python excels at. The simplicity and less restrictions of the language made the development process much easier, as research, studying, experimenting with lambdas and generators, Python is a deep language with a progressive learning curve and combined with Django this application was possible. The more research performed the less code needed to produce working pages, debugging was easier as Python tolerated mistakes, when there was a small issue in the code base things didn't break completely in fact tracking errors consumed less and less time. Python's frameworks such as Django allowed much easier development of a very complicated intricate project.

11.0 References

[1] View, D., 2021. *Django’s Structure – A Heretic’s Eye View - Python Django*. [online] The Django Book. Available at: https://djangobook.com/mdj2-django-structure/ [Accessed 29 April 2021].

[2] Djangoproject.com. 2021. *The Web framework for perfectionists with deadlines | Django*. [online] Available at: https://www.djangoproject.com/ [Accessed 29 April 2021].