**BACKEND DOCUMENTATION**

For the Livestock Management Software

What is this software for?

This software is for goat herders to manage their finances and livestock of goats using this software. It is built for those who want to manage their finances relating to the livestock they own all at one place with an intuitive UI and a fast response time.

Tools used to build this software:

Python – Used for both frontend and backend.

SQLite – For Database.

**DOCUMENTATION**

What is this documentation for?

To hopefully help anyone reading this code to understand the mess of a code that I have written and for myself in the future if I ever return to update the software.

What happens when the software is started for the first time?

Once the software starts, and the backend script is called upon, a database file is created in the working directory if the database file is not already present and initializes all the tables required.

The tables initialized are: WRITE ABOUT THE COLUMN NAMES AND EXPLAIN THEM TODO

* MasterTable – To hold all the basic information about the goats. This will be queried in several functions to retrieve the information about the selected goat and output the desired result.
* WeightTable – To hold all the weight changes about a particular goat.
* KidsIdTable – To hold all the data about a particular goat.
* LivestockNetworth – To hold all the financial information about the breed of a goat.
* Labour – To hold all the financial information about the laborers employed.
* Feed – To hold all the financial information about the amount spent on feed for the animals.
* HealthExpense – To hold all the information about the amount spent on maintaining the health of the goat(s).
* Misc – To hold all the financial information about Miscellaneous accidental information.

What happens next?

A connection to the database is established (variable name *c*) using python’s inbuilt library – sqlite3. A cursor variable with the name *conn* is also established.

What is the weird *if* statement that checks for the first of April?

That condition runs when it is the beginning of a new financial year. It erases all the data in the tables Labour, Feed, HealthExpense, Misc, LiveStockNetworth and all the dead or sold goats from the MasterTable.

Following that, the LivestockNetworth table is updated using the *updateLivestockNetworth* function.

What is the *updateLivestockNetworth* function?

It takes in one argument named *values*. It selects all the breed and category from the LivestockNetworth table.

According to the category(age + gender) of the goat, the LivestockNetworth table is updated. If a category or breed is not already present, it is created as a new record in the table, else the existing values are updated.

This function is called at the beginning of the financial year(1st of April) to keep all the tables in sync.

***CLASS DATABASE****:*

\_\_init\_\_ - Initializer method

* No purpose for the initializer.
* Input – No args
* Output – No output

getGoatRecords – Class method

* Returns all the records present in the MasterTable.
* Input – No args
* Output – List of list of records

getNumberOfRecords – Class method

* Returns the number of records present in the MasterTable.
* Input – No args
* Output – int

getColumnNames – Class method

* Returns the name of the columns present in the MasterTable.
* Input – No args
* Output – List of str

insertGoatRecord – Class method

* If no mother ID is present for the goat, (i.e.) the goat was bought or it’s mother is unknown, it gets inserted as a new record with all the values provided to it.
  + Inserts into the columns – goat\_no, breed, date\_of\_birth, gender, pregnant, weight, v1, v2, v3, v4, v5, v6. The vaccination columns are set to the system’s date and time for ease of calculation for vaccination dates.
* Else
  + In the MasterTable a new record is created and these specific columns are written into – goat\_no, breed, date\_of\_birth, gender, pregnant, weight.
  + In the KidsTable a new record is created and the following columns are written into – mother\_id, kid\_id, gender.
  + The number of kids in the MasterTable are updated for the mother\_id provided.
* The mortality of the new record is updated to *Alive*.
* A separate WeightTable for the goat record inserted is created with the name of the WeightTable being initialised as *(WeightTable + goat\_id)*.
* The *LivestockNetworth* method is called to update the LivestockNetworth table.
* Input – Dictionary containing the values TODO
* Output – Consoles the string “Inserted successfully”

updateLiveStockNetworth – Class method

* It takes in one argument named *values*. It selects all the breed and category from the LivestockNetworth table.
* According to the category(age + gender) of the goat, the LivestockNetworth table is updated. If a category or breed is not already present, it is created as a new record in the table, else the existing values are updated.
* Input – Dictionary containing values TODO
* Output – No output

getMaleKidId – Class method

* Returns the male Kid ID from KidsTable for the given mother ID.
* Input – Int mother\_id
* Output – List of all the male kids\_id present for the given mother ID

getFemaleKidId – Class method

* Returns the female Kid ID from KidsTable for the given mother ID.
* Input – Int (mother\_id)
* Output – List of int (all the male kids\_id present for the given mother ID)

getMotherId – Class method

* Returns the mother ID from KidsTable for the given goat ID.
* Input – int (goat\_no)
* Output – Int (mother\_id)

UPDATE WINDOW

updateGoatRecord – Class method

* Updates the following columns in the MasterTable for the goat ID provided
  + mortality, breed, date\_of\_birth
* Also updates the following columns based on whether it has been changed
  + sold\_date, sold\_rate
* If the weight is updated,
  + The weight in the MasterTable for the corresponding goat is updated.
  + A new record in the WeightTable corresponding to the goat is inserted.
* If the goat is marked as pregnant by the user, the following columns in the MasterTable are updated
  + The pregnant column is updated
  + The date of delivery is calculated and set to 150 days in the future from the current system date.
* Input – dict goatValues, bool isSold, bool isWeightUpdated
* Ouptut – Prints “Updated Successfully” in the console

updateVaccination – Class Method

* Updates the vaccination date of any one of the six vaccinations by using the system date for the goat ID passed.
* Input – int vacc\_no (Vaccination Number), int goat\_id
* Output – No output

ALERTS WINDOW

getGoatsToBeVaccinated – Class Method

* Returns all the goats that are to be vaccinated within 30 days from current system date, sorted in ascending order of dates
* It selects all the vaccination dates that are after the current system date from MasterTable.
* All the queried records are looped over and 30 days are added to the dates.
* This list of dates are sorted in ascending order and returned.
* Input – int vacc\_no
* Output – list of lists containing the goat ID and the vaccination date (in str datatype)

getVaccinationDates – Class Method

* Returns a list of vaccination dates of all sizx vaccinations for the given goat ID.
* Input – int goat\_no
* Output – list of dates (in str datatype)

getDeliveryDates – Class Method

* Returns a list of goats that have a delivery date in the MasterTable.
* Input – No args
* Output – list of lists containing the goat ID and the date of delivery

getBreedReadyGoats – Class Method

* Returns a list of goat ID and breed which satisfy the following conditions:
  + The goat is adult (has lived over 365 days).
  + If 90 days has passed since the goat’s delivery date or the goat is marked as not pregnant.
  + The gender of the goat is female.
* Input – No args
* Output – list of lists containing goat ID and breed

VIEW GOAT WINDOW

getKidsTableData – Class Method

* Returns a list of Kids ID for the given mother goat ID.
* Input – int goat\_no
* Output – list of kids\_id

FINANCE WINDOW

insertLiveStockNetworth – Class Method

* Updates the cost and total\_cost of the LivestockNetworth table.
* It is updating because when a new goat is added to the MasterTable, if the breed is new, a record is added to the LivestockNetworth table. TODO check if this is correct.
* Input – dict networthValues
* Output – No ouput

getTotalLivestockNetworth – Class Method

* Returns the sum of the total cost of all breeds and categories of goat from the LivestockNetworth.
* Input – No args
* Output – int

getLivestockNetworthData – Class Method

* Returns all the records from the LivestockNetworth table.
* Input – None
* Output – list of lists containing the LivestockNetworth table records

LABOUR TABLE

insertLabour – Class Method

* Inserts a record in the Labour table for the following columns:
  + category, salary, count
* Calculates the monthly total by multiplying the number of labourers and the salary of the labourers and the number of days which has been set to 30. That is, calculates the amount of salary per 30 days for all the labourers present in a particular category.
* Input – dict labourValues TODO
* Output – No output