User Documention

CIS 643

Ben Hartman, Yijun Lin, Colden Miller

**Introduction**

The real-world problem that is being solved with our group is helping the *Kansas Artificial Breeding Service Unit* transform their entire paper system into a fully developed software where they will be accessing a database through a user interface to update, delete, and add new records based on bull semen vials. The most important program features are the user interface and the database just mentioned. Going from a full paper system into a user interface is extremely important and brings a lot of pressure onto the group. This is because most individuals do not know how applications and software work so we need to make our interface very user-friendly for them. The database is another important feature because that will be running in the background for everything the software requires us to do. We need it to be very precise and accurate so when the project is fully released, they will not have to worry about making edits inside the database since that is something tricky for non-developers. The database can also be a constraint in our project because users will need to have access to the *Kansas State University* servers in order for the database to be connected to the application. The purpose of this document is to provide insight for what our system is intended to do, how we designed it, and its functionality. The intended audience of this document is for computer scientists who are intrigued by our work and how our system works.

**Project Overview**

The client we have held meetings with for this system is Leon Prather, who was the main employee of the paper system in the office. He will be one of the main users of this system alongside one or two other office workers. The problem we are trying to solve as discussed earlier is being able to transform a full paper system that includes tons of Excel documents tracking bull’s and their individual semen vials that are stored in separate freeze shipment containers, and trying to turn that into a software user interface being able to communicate with a database that has similar fields and keys in relation to the excel sheets. The database uses MySQL and is connected through the university servers and includes queries for deleting, adding, retrieving, and storing bulls, owners and records. Each bull has its own collection date for its semen vial, the can number which represents which freezing tank that vial is in at the breeding unit, the number of vials, its name, and what kind of breed it is. The main features of the user interface are the different windows that provide user input or modifications that communicate with the database servers to provide real-time data configurations. The most important constraints for our design decisions are modifying the paper system to fit their needs but also be able to provide a reasonable interface and communication in the system from program to database without getting overly complicated.

**Development and Target Environments**

The physical environment our project is used in is a Windows Forms application in Visual Studio. The database is in “phpMyAdmin” for the computer science department.

**System Model**

Our system is made up of multiple user-interface windows as shown in Appendices A-B which are interconnected with the database (appendix C). Starting at appendix A, the main window is what the user first sees when running the application. The window object contains objects to the “Record Window” and “Search Window” as those are the main functionalities from that page. The “Search Window” contains a “Search Window Results” object and can pull in a list of search results and search terms that call on the database for a specific query. The “Record Window” contains objects to the “Note Window” and “Additional Info Window”. The record window also links to the “Search Result” class and “Record” class shown in appendix B that are in connection with the database for pulling or adding new records. The last main window is the “Inventory Page” which is similar to the “Record Window” GUI since it communicates with similar classes that pull from the database but just use that data in a different manner to display and easily update certain data functionalities.

Looking at our database diagram, it is fairly simple. Our main tables are ‘Person’ and ‘Animal’ which are to add the owners of the bull and the bull itself. From there, we have a ‘Data’ table which is for the bull’s collection date, can number, etc. to be referenced in the system. For now, we have a ‘Sample’ table for importing fake data for testing.

**User Interaction**

When a user accesses the program, they first see our main page which shows a calendar and two buttons; “Add New Record”, and “Modify Record”. If the user clicks on the “Add New Record” button, it will prompt to open the “Record Window” form which shows a blank document regarding the name of the bull, its can number, code number, breed, owner, date, number of units, etc. Once the user finishes entering in the information, all they need to do is close the window to save it. Once they hit the exit button, the “Additional Info” window appears to enter the type of species, city, state, and country of the bull. If the user clicks the “Modify Record” button, it will go to the “Search” window which allows users to provide a filter on their search, look up the total can capacity, total sum of units, or provide an inventory list which we’ll explain soon. You can filter on the owner, breed, animal name, code, can number, town, and/or state. If the user were to click on the “search” button, it will display all of the data from the database and a user can choose to double-click on a given row of data which will populate the “Record Window” which the data or they can click a row and then click on the “Pull Data” button towards the button right which will open the “Inventory Page” with the provided data that the breeding unit uses for inventory of a bull and the number of units they have left.

**Functional Requirements**

The functional requirements of the system are to add new data, update the record windows, update the inventory pages, and find records with key information in the search results window.

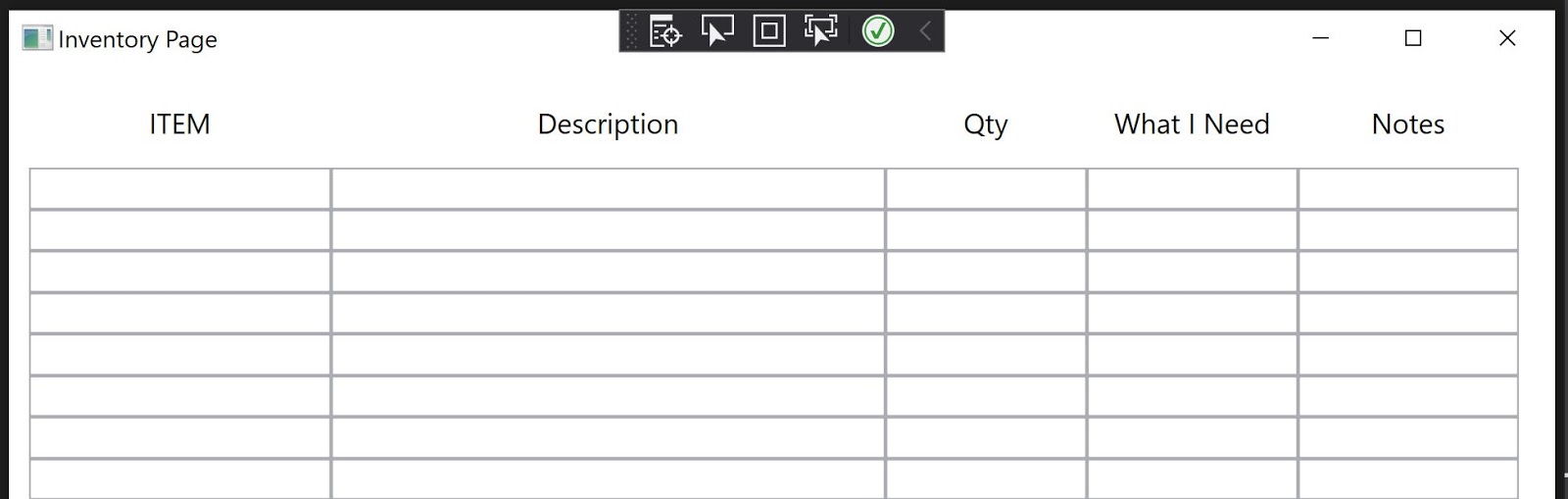
**Nonfunctional Requirements**

The non-functional requirements our system must operate under are being able to connect to the Kansas State servers with no issues, be error-free in its performance, and be able to maintain a high capacity of data and information while being scalable.

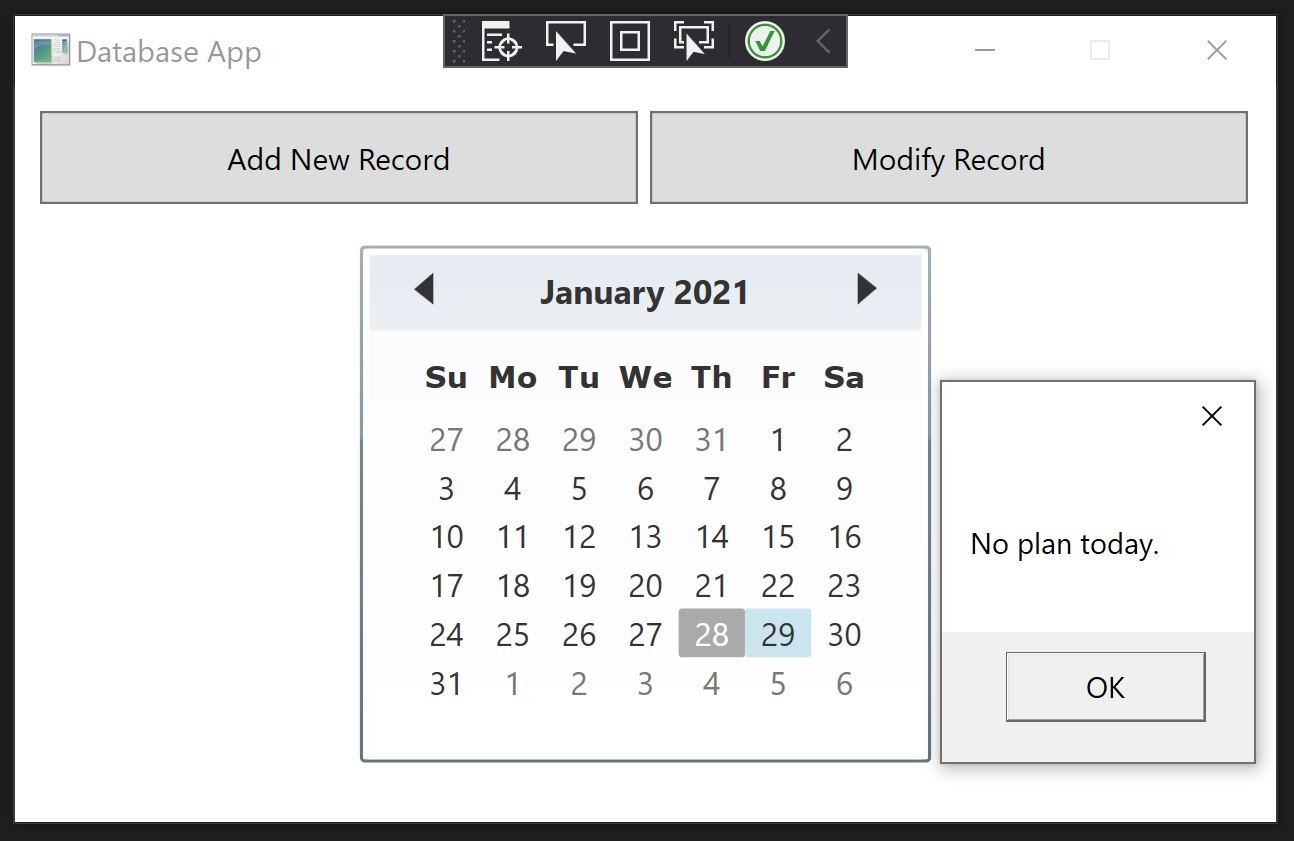
**Semester Goals**

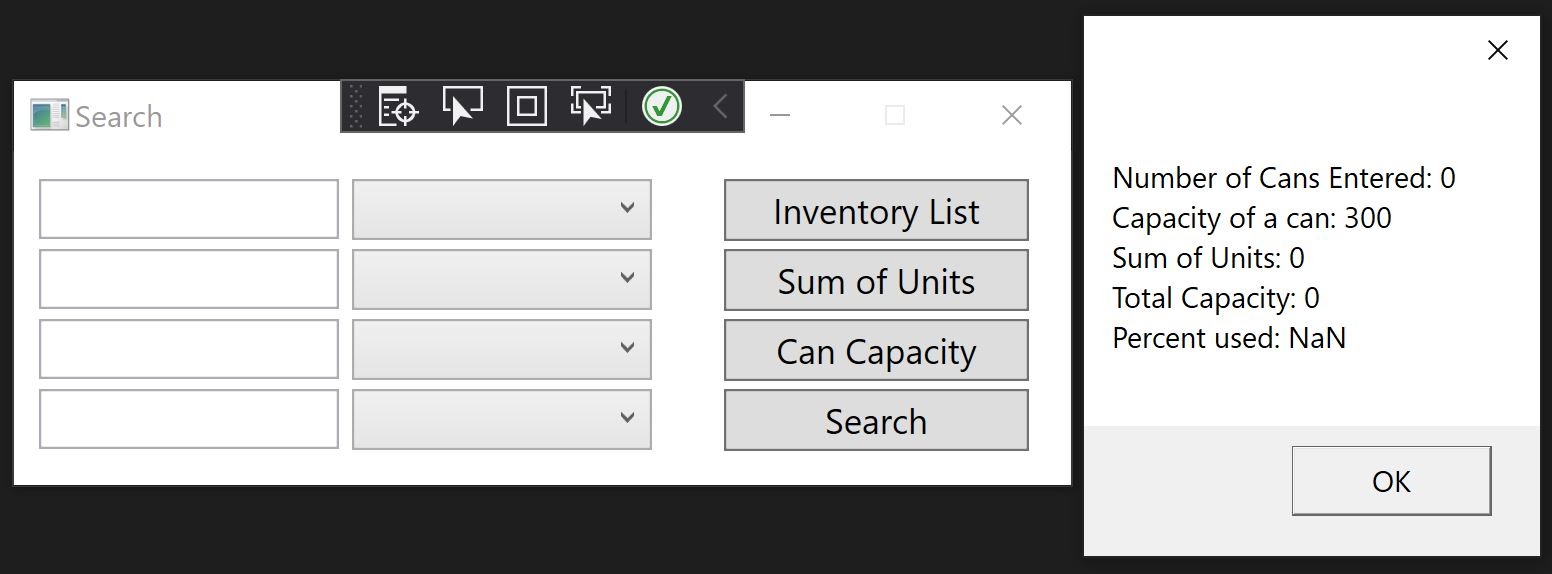
Minimum viable product that meets needs of customer:

Inventory Page that can contain a list of items for shipping purposes. Allowing the user to pick multiple items from different bulls and cans. The program then updates the database based on what the user takes.



Enhanced version that incorporates all desired features:

Clicking on a date should give you a list of everything coming in and going out of the store

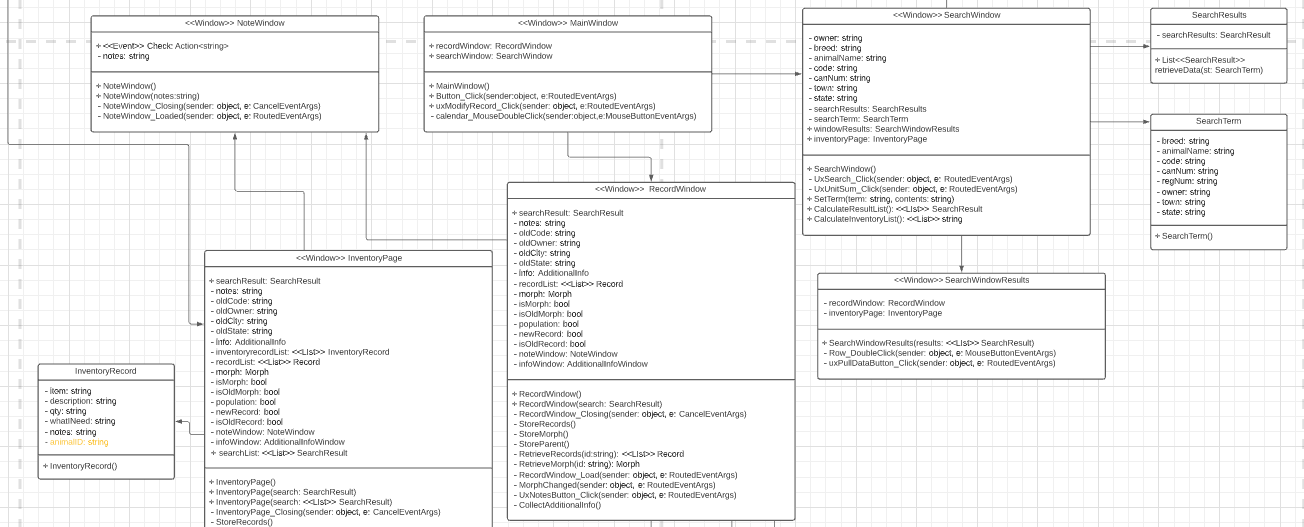
Capacity of the cans. Allows to check the capacity of one or more cans.

Auto-fill function that will fill in the black when entering a new record. This also includes error checking to make sure that there are not two of the same bull in the database.

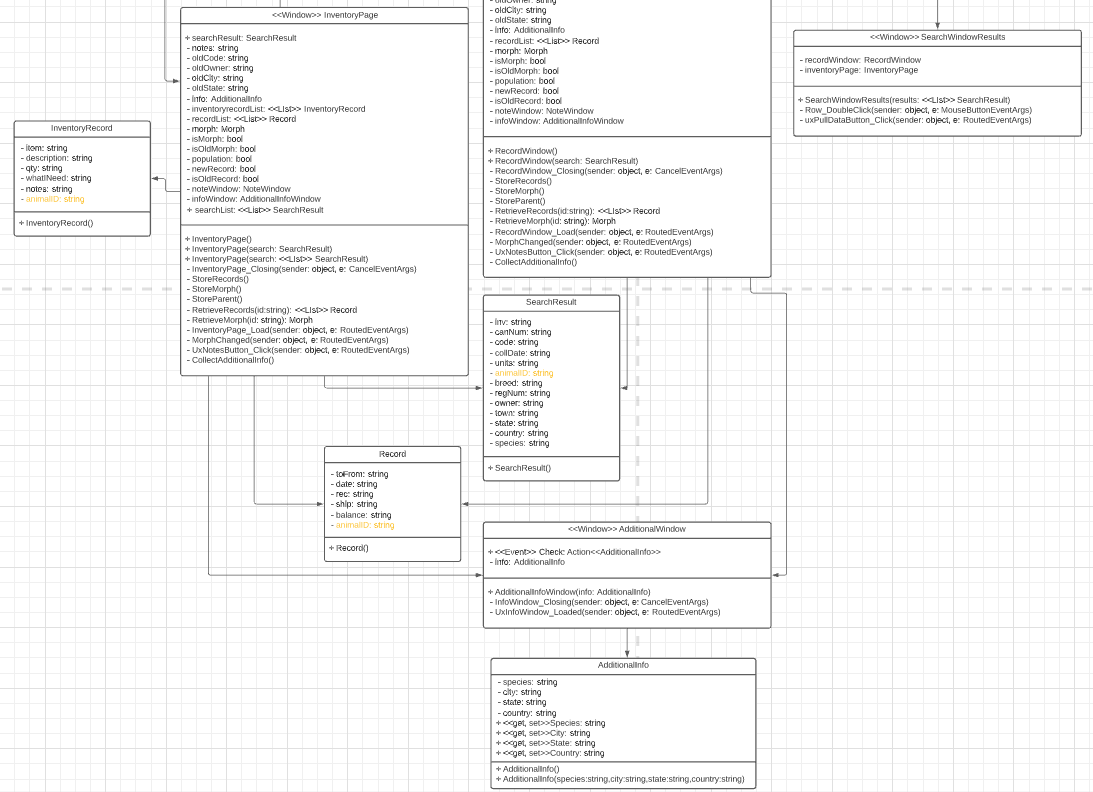
**Appendices**

UML for Class Diagrams:

Appendix A: \*Top half\*



Appendix B: \*Bottom half\*



Appendix C: Diagram for Database:

