**C868 – Software Capstone Project Summary**

**Task 2 – Section C**

Logo

Description automatically generated

|  |  |
| --- | --- |
| **Capstone Proposal Project Name:** | Farm Inventory: Cost Tracker |
| **Student Name:** | Craig Lohrman |

Table of Contents

[**Application Design and Testing** 3](#_Toc131704479)

[**Design Document** 3](#_Toc131704480)

[**Class Diagram** 3](#_Toc131704481)

[**UI Design** 4](#_Toc131704482)

[**Unit Test Plan** 5](#_Toc131704483)

[**Introduction** 5](#_Toc131704484)

[**Purpose** 5](#_Toc131704485)

[**Overview** 5](#_Toc131704486)

[**Test Plan** 5](#_Toc131704487)

[**Items** 5](#_Toc131704488)

[**Features** 6](#_Toc131704489)

[**Deliverables** 6](#_Toc131704490)

[**Tasks** 6](#_Toc131704491)

[**Needs** 7](#_Toc131704492)

[**Pass/Fail Criteria** 7](#_Toc131704493)

[**Specifications** 7](#_Toc131704494)

[**Procedures** 8](#_Toc131704495)

[**Results** 9](#_Toc131704496)

[**Source Code** 9](#_Toc131704497)

# **Application Design and Testing**

# **Design Document**

## **Class Diagram**

The class diagram below is the primary 4 Classes. Java is the application's language, and the diagram shows the model classes using the MVC structure. The classes in the image are the database’s mapping.

The one classes Animal is the main class. You must have an animal to add a visit, medication, or item. The Animal class is a one-to-many relationship, and the other three are many-to-one. There is a Login class that is not shown.



## **UI Design**

Text, letter

Description automatically generated The low-fidelity image is the first draft of the animal screen. With the client’s feedback on the low-fidelity version, a few changes were made when looking at the following image of the high-fidelity. The significant difference was adding the list of all the animals in birth date order, youngest to oldest. Then you can see a Clear button was added, all with how some fields have changed from text boxes to drop-down menus and a Date Picker.

![Graphical user interface

Description automatically generated]() *First Draft of the Low-Fidelity Animal Image*

*The Final Version of the High-Fidelity Animal Image*

# **Unit Test Plan**

## **Introduction**

### **Purpose**

The testing plan that was used was manual testing. This is where you enter data yourself correctly and then incorrectly. You verify that the correct errors show when they are supposed to. The first test was to the add button, so after typing in all the correct data and clicking the Add button, the list should show a new row in the table view. If the erroneous data were entered, it would pop up with an error telling you which fields are incorrect. You click the ok button on the pop-up and correct the listed fields. Once all the fields have the correct data, it will add the row to the table view and the database.

### **Overview**

The first test was done on the Animal screen. I will enter all the correct information, click the add button, and watch to ensure it was added to the database and table view. This being the first test, it did not work and had a runtime error. I had put the variables in the insert command in the wrong order. After correcting the problem, it added the animal to the database and table view. Then click on the new row and the data fill in the fields with the data saved from the database. Now I would do a validation test on the user’s entered data. If a field is blank or the date selected is the future, a pop-up will show and tell the user which fields are missing entered data or if the chosen date is a future date.

## **Test Plan**

### **Items**

Once a screen is coded, you go to the main class and adjust the starting screen to the tested one. This way, it will make it quicker to get the screen testing started. To try the Add button, enter the data for each field, select a date in the past or present, and click the Add button. With all the correct data entered, it should add the new row to the database and the table view. Next, click the Clear button, verify that all the fields are blanked out, and then click on the new row in the table view to confirm that the data is loaded in the correct fields. With the data loaded in the fields, click the Delete button. A pop-up will ask if you are sure you want to delete, then another pop-up telling you that the row was deleted, verifying the row was removed from the database the table view.

Now run through the test again but leave one or more fields blank. Confirm that the pop-up happens and the errors are correct with what is missing or wrong. Also, ensure it did not add anything to the table view or database. Following these same steps as each screen is completed.

### **Features**

The fieldCheck() is used to check if any field is missing and the date selected is not in the future. It first gets the text data from each text box and stores that in the assigned variable. Sets the String error to blank. When the screen is loaded, it creates “dates” as a utilities.Dates to be used throughout the screen. The first step is that all the fields are not blank; if any, add that message to the error String in the pop-up. Then it uses the dates variable to call two methods, the currentDateFinder() and convertDateToCalendar(). The currentDateFinder() gets the current date, and convertDateToCalendar() takes a passed-in date and converts it to a Calendar data type to verify that the selected date is not in the future. If anything is wrong, it is added to the String error and displayed in the pop-up.

### **Deliverables**

When testing, a pop-up should tell you what is missing or wrong. If a field is missing, it displays that “*field name*” is missing, please enter “*field name*.” And if it is that the date is in the future, it adds, “The date selected is in the future, please select another date that is in the past.”

### **Tasks**

Manual testing will follow these steps:

1. Adjust the main file to load straight to the screen that will be tested; this is only to make testing one screen easier instead of having to click through to that screen each time.
2. Enter data in some or none of the fields.
3. Read the pop-up message, and verify it lists what is missing correctly.
4. Select a date that is in the future and verify that the error is on the pop-up.

### **Needs**

Manual testing requires only what is needed to run the application.

### **Pass/Fail Criteria**

A positive or pass test is when the fieldCheck() sends back false, the String variable error is not null, or it sends back true, and the error variable is blank. A negative or fail test is when fieldCheck() sends back false, and error is empty or sends back true, and error is not blank.

## **Specifications**

This is the code for the fieldCheck() for the AnimalController. All the core classes have a fieldCheck() method; for the most part, they are the same, with just a few differences.

![A picture containing table

Description automatically generated]()

## **Procedures**

The code above shows that each textbox is saved in a variable. Then it checks each variable to see if it is empty or not. Empty means it adds the error message to the variable error and moves on to the next check when it is not blank. After that, it creates a variable of a Calendar datatype and saves the date sent back from the currentDateFinder() method of the current local date. Then creates another Calendar variable to store a date. It passes the entered date from the Date Picker field to be converted from a String to a Calendar with the convertDateToCalendar() method. It then looks at the current date and the birthdate variables to verify if the birthdate is in the future.

## **Results**

![Graphical user interface, website

Description automatically generated]()The first image is what happens when no data is entered. The pop-up shows that each field is missing. The second image is what happens when a future date is selected, and some fields are blank.

![Graphical user interface

Description automatically generated]()*No data entered.*

*Future date and some missing fields.*

# **Source Code**

All the source code is in the file named CLohrma\_C868\_Capstone.zip along with javafx-sdk-17.0.1 and mysql-connector-j-8.0.32.jar and generated Javadoc files.