

Table of Contents

1. Introduction.....	3
2. Project Management	4
2.1. Project Schedule.....	4
2.2. Project Backups	5
3. Requirements	6
3.1. UML Diagrams	6
3.1.1. Use Cases Diagrams.....	7
3.1.2. Class Diagrams	13
3.1.3. Sequence Diagrams.....	14
4. Design	15
4.1. Graphical User Interface (GUI)	15
4.2. Code Design.....	20
4.2.1. Code Structure.....	20
4.2.2. Refresh	21
4.2.3. Progress Bar	21
4.3. Database Design.....	22
5. Implementation	23
5.1. Tools	23
5.1.1. Diagramming Tool – Visual Paradigm	23
5.1.2. Integrated Development Environment - NetBeans	23
5.1.3. Database - Derby.....	23
5.1.4. Web Hosting – ServersFree	24
5.2. Languages	24
5.2.1. Main Implementation Language – Java	24
5.2.2. Other Languages – SQL, HTML	24
5.3. External Resources.....	24
5.3.1. JAR Files.....	24
5.4. Difficulties	24
6. Testing.....	26
6.1. White-box Testing	26
6.2. Black-box Testing.....	26
6.2.1. Outside Testers.....	27
6.2.2. Usability Testers	28

6.3.	Bugs we Found Running.....	29
7.	Review of the Requirements	30
8.	Future Developments	31
9.	Abbreviations	32
10.	References.....	33
10.1.	Websites/CD URL's	33
10.2.	Images	33

1. Introduction

This project was undertaken as part of my fourth year project. This project is to create a finance manager that will keep track of the expenses of a farm, and will make reports on such expenses. This finance manager does not deal with taxes or VAT, as its intended goal is not to act as an accounting application. The intending objective of this project is just to keep track of expenses. Future developments will allow for the development of this project to become more of an accounting based application; however this is not the case now as there is was not enough knowledge in accounting throughout development of this project.

The idea for this project came to me as I was trying to think of something that the market was lacking. There are many finance managers out there, but none seem to help with the managing of a farm's finances. This intrigued me as I believe that there could be a potential market for an application to help farmers to keep track of their finances. That is how the project was conceived. Last semester I asked a few local farmers for a few ideas of what they might want to see in a finance manager for a farm. I took their points on board and tried to incorporate them into my project, while also trying to stick to the plans that I had decided on my designs.

This project does not cover every aspect of a farm, as it does not cover every type of farming, nor does it cover every possible expense of a farm. This project deals with the finances that a beef farmer might need to keep track of as the local farmers that I had asked during my research phase were all beef farmers.

2. Project Management

2.1. Project Schedule

In this section I show my project schedule from before I undertook this project as how I planned to go about doing the work. This schedule was followed closely for the most part during the implementation of this project. However, other elements came into play throughout the duration of this project which altered the schedule. At the start of the semester we were informed that we would have an outside tester assigned to test our projects. This changed the need for me to have a full project test every few weeks as I would be able to have an outside tester test my project as a whole, every week. This freed up some time for me to perform more rigorous component tests with each addition of a new feature. I had also run into a few problems as I needed to take a full week off before the Easter break to start a project due before Easter for one of my modules. This did not have as big an effect on the project schedule as it should have had as I was quiet ahead of schedule prior and for the time I saved on what should have been spent on testing, I was quickly able to get everything back on track.

Date	Description	Status
07-01-2013	Design the Graphical User Interface (GUI) and implement it as the base of the skeleton code.	Finished
14-01-2013	Design and implement the database tables.	Finished
21-01-2013	Design the functions in the beef tracker module.	Finished
04-02-2013	Implement the beef tracker module with the Graphical User Interface and test to see how well the GUI can work with and without the beef tracker module.	Finished
18-02-2013	Test how well the skeleton code works with the database and if the right connections can be made.	Finished
25-02-2013	Test how well the skeleton code with the attached module, beef tracker connects to the database.	Finished
04-03-2013	Design and build functions to turn the textual data into graphical output.	Finished
11-03-2013	Design and build function to show projects profit and loss.	Finished
25-03-2013	Test for accurate information.	Finished
01-04-2013	Make test plan to cover all functions and operations for third party testing the application. Allow user space for errors in functions not listed.	Finished
08-04-2013	Fix any bugs found. Test the overall project again. Fix any bugs.	Finished
19-04-2013	Final test of the project. If any bugs, highlight in documentation as current errors. Hand up of documentation and project.	Finished

I meet with my project supervisor on a weekly basis and we discussed where I was at in my implementation phase, what had been done and what was left to finish. These meetings were half an hour long. Within this time, I would show a brief demonstration of the new features and functionality done during the week. I would be given advice on whether I should do something differently or whether something was ok with the project. One of the biggest changes in the design came from the second meeting, where I changed the options panel from buttons, into a collapsible list of options. This had a big knock on effect to the rest of the project as I was able to put all of the options into this list, rather than having these options in the main panels which showed the information from the database. This allowed for a much

neater, easier to find list of options. This however, had quite an impact on the rest of my designs, possibly for the better.

2.2. Project Backups

Throughout the duration of my project, I have kept countless backups of my project. Each with a small text file, to document the stages of the project. In these text files I have noted, what new features were added, what needs to be done, and possible solutions to errors that are causing flaws and or bugs in the project. These backups were created any time I was adding something that was unknown or if I was making a change to something that already worked. This was due to not wanting to waste time and trying to back step to the last known working phase. By making a backup before I added something new to the project, I was able to experiment with the code to learn how these new features worked, as I was able to make changes to past, known working features without having to worry about making too big a mess as I could always revert back to a previous known working backup.

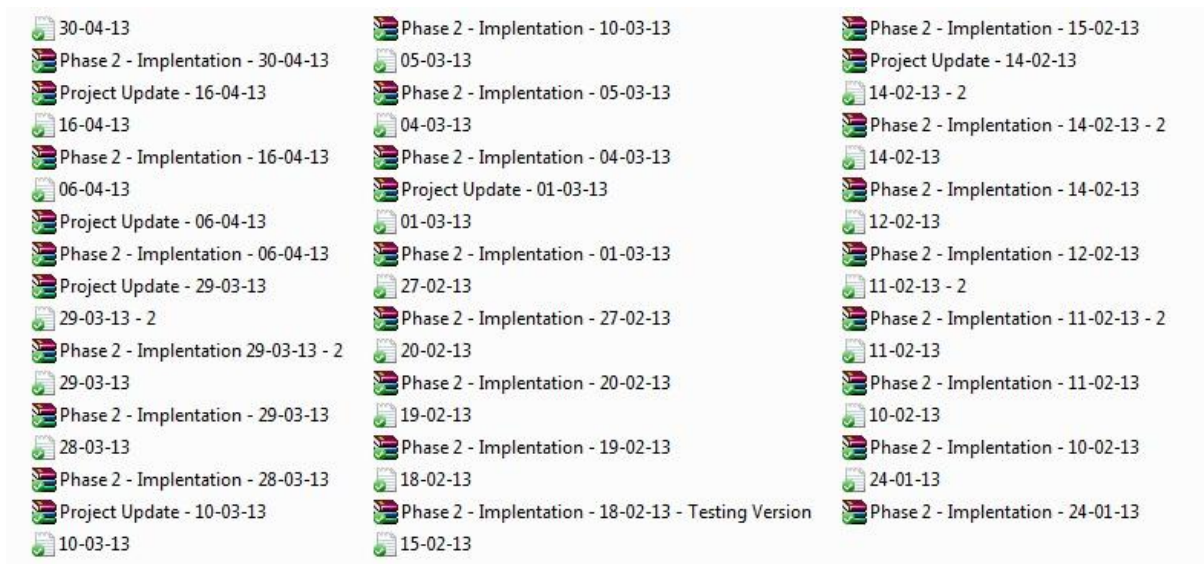


Figure 1

3. Requirements

In this section, I will list out all of the requirements that are needed to complete this project. I will also show Unified Modelling Language (UML) diagrams to show the overall structure of the project. These diagrams will be Use Case diagrams. These requirements will hopefully become concrete steps to help the development of a complete project and not become a dead weight that will cause problems in implementing the solutions to these requirements.

Functional Requirements:

- Core finance tracking features.
- Capture expenses.
- Profit and loss projections
- Create a modular system to allow different components to install to the application.
- Online updates.

Non-functional Requirements:

- Should be reliable.
- Should be efficient with retrieving information.
- It should be easily maintainable.
- The graph screen show not look cluttered.
- The use of colours to help with graphs, spreadsheets and to add eye appeal to the Graphical User Interface (GUI).
- Spreadsheets will only show relevant information to what the user has asked it to return.

These are the requirements of this project which have all been taken into account. As this project was being developed.

3.1. UML Diagrams

UML diagrams act as the blueprint for a project as the diagrams show the requirements, functionality, structure and the workings behind the project. These diagrams help to see what has been done and what has yet to be done. These diagrams however, are only an aid to help developers and primarily the future developers who will maintain the project after it has been released.

There have been many conflicting reviews on how important UML diagrams are to the success of a project, *“So its not a huge waste of time for planning how to initially attack the problems. But likely a waste of time for documenting your code for future programmers. To use a military metaphor, you need a battle plan, but you wouldn't give that plan to historians to use as an after action report.”*¹ This quote shows UML from the point of view that it is a development tool, which can be useful, but not essential to the success of a project.

“it is standardized (by OMG Group), not proprietary, supported by software industry - is explained and described in every aspect by vast amount of publications, resources, textbooks, etc. - it could be customized and extended for specific application domain, software process,

¹ <http://programmers.stackexchange.com/questions/144530/how-important-are-uml-diagrams-for-a-successful-project>

or implementation platform.”² Some of the advantages mentioned about UML in the above quote show why UML is so effective. There are lots of resources behind UML and the fact that it has been standardized makes it easy for anyone with knowledge in UML to read the diagrams shown.

3.1.1. Use Cases Diagrams

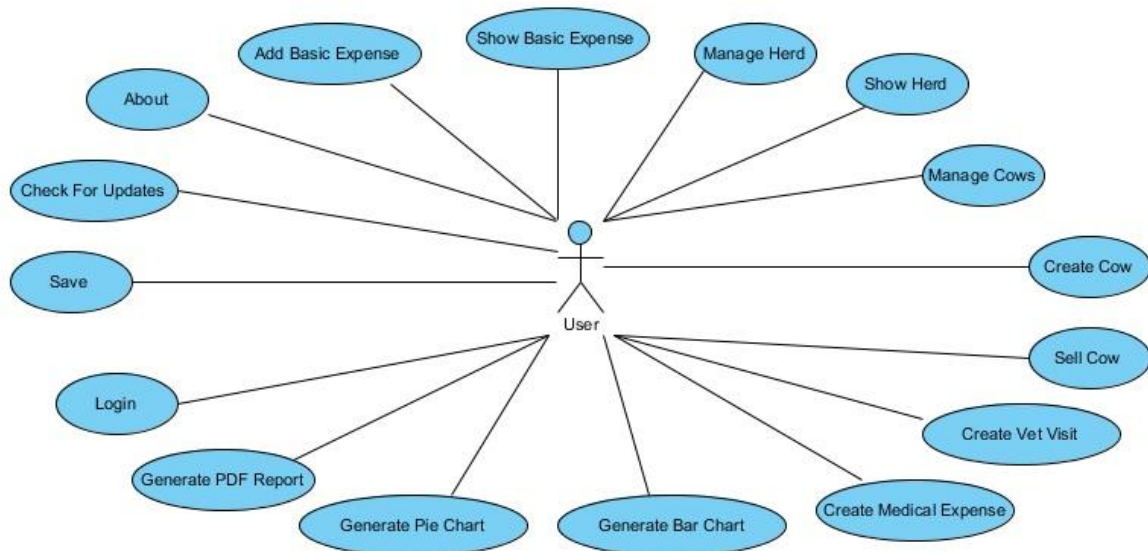


Figure 2

Use Case ID	1
Name	Login
Actors	User
Description	The user logs into the application
Trigger	User runs the program
Preconditions	User wants manage the farm finances.
Normal Flow	1. User is presented with a login prompt 2. User enters username and password 3. User logs in
Post Conditions	User is logged in and can use the application
Alternative Flow	3. User enters wrong username and password

² http://wiki.answers.com/Q/What_are_the_advantage_of_UML

Use Case ID	2
Name	Save
Actors	User
Description	The user saves the current state of the database
Trigger	User clicks save on the menu bar
Preconditions	User has logged into the application
Normal Flow	1. User chooses the 'Save' option under 'File' 2. PDF Reports are generated into their respective folders
Post Conditions	PDF Reports are generated in the report folders
Alternative Flow	There is no alternative flow

Use Case ID	3
Name	Check For Updates
Actors	User
Description	The user can check for updates in internet access is available
Trigger	User clicks check for updates on the menu bar
Preconditions	User has logged into the application Application has access to the internet
Normal Flow	1. User chooses the 'Check for Updates' option under 'Help' 2. Window opens showing the update website
Post Conditions	Window opens showing the update website page
Alternative Flow	2. No website to view

Use Case ID	4
Name	About
Actors	User
Description	The user will receive a prompt showing information about the application
Trigger	User clicks about on the menu bar
Preconditions	User has logged into the application
Normal Flow	1. User chooses the about 'Option' under 'Help' 2. A prompt will be displayed showing information about the application
Post Conditions	A prompt will be displayed showing information about the application
Alternative Flow	There is no alternative flow

Use Case ID	5
Name	Add Basic Expense
Actors	User
Description	The user will add an expense to the basic expense table
Trigger	User clicks on the apply button under add expense
Preconditions	User has logged into the application
Normal Flow	1. User chooses 'Basic Utilities' 2. User chooses 'Add Expense' 3. User clicks 'Apply' 4. The table is updated with new information
Post Conditions	The new information will be displayed in the table
Alternative Flow	3. A prompt is shown displaying which information needs to be entered

Use Case ID	6
Name	Show Basic Expense
Actors	User
Description	The user will view the expenses from the basic expense table
Trigger	User clicks on the apply button under show expense
Preconditions	User has logged into the application
Normal Flow	<ol style="list-style-type: none"> 1. User chooses 'Basic Utilities' 2. User chooses 'Show Expense' 3. User clicks 'Apply' 4. The table is updated with new information
Post Conditions	The new information will be displayed in the table
Alternative Flow	There is no alternative flow

Use Case ID	7
Name	Manage Herd
Actors	User
Description	The user can create and terminate a herd from here
Trigger	User clicks on the apply button under manage herd
Preconditions	User has logged into the application
Normal Flow	<ol style="list-style-type: none"> 1. User chooses 'Beef Farming' 2. User chooses 'Herd Management' 3. User chooses 'Manage Herd' 4. User clicks 'Apply' 5. The table is updated with new information
Post Conditions	The new information will be displayed in the table
Alternative Flow	4. A prompt is shown displaying which information needs to be entered

Use Case ID	8
Name	Show Herd
Actors	User
Description	The user can view all the cattle in the herds
Trigger	User clicks on the show herd option in the option panel
Preconditions	User has logged into the application
Normal Flow	<ol style="list-style-type: none"> 1. User chooses 'Beef Farming' 2. User chooses 'Herd Management' 3. User chooses 'Show Herd' 4. The table is updated with new information
Post Conditions	The new information will be displayed in the table
Alternative Flow	There is no alternative flow

Use Case ID	9
Name	Manage Cows
Actors	User
Description	The user can add cattle to the a herd
Trigger	User clicks on the manage herd option in the option panel
Preconditions	User has logged into the application
Normal Flow	<ol style="list-style-type: none"> 1. User chooses 'Beef Farming' 2. User chooses 'Herd Management' 3. User chooses 'Manage Cows' 4. The table is updated with new information
Post Conditions	The new information will be displayed in the table
Alternative Flow	There is no alternative flow

Use Case ID	10
Name	Create Cow
Actors	User
Description	The user can create an new animal
Trigger	User clicks on the create cow option in the option panel
Preconditions	User has logged into the application
Normal Flow	<ol style="list-style-type: none"> 1. User chooses 'Beef Farming' 2. User chooses 'Cow Management' 3. User chooses 'Create Cow' 4. User chooses 'Create Cow' 5. The table is updated with new information
Post Conditions	The new information will be displayed in the table
Alternative Flow	4. A prompt is shown displaying which information needs to be entered

Use Case ID	11
Name	Sell Cow
Actors	User
Description	The user can sell an animal
Trigger	User clicks on the sell cow option in the option panel
Preconditions	User has logged into the application
Normal Flow	<ol style="list-style-type: none"> 1. User chooses 'Beef Farming' 2. User chooses 'Cow Management' 3. User chooses 'Sell Cow' 4. User chooses 'Apply' 5. The table is updated with new information
Post Conditions	The new information will be displayed in the table
Alternative Flow	4. A prompt is shown displaying which information needs to be entered

Use Case ID	12
Name	Create Vet Visit
Actors	User
Description	The user can create a new vet visit
Trigger	User clicks on the create vet visit option in the option panel
Preconditions	User has logged into the application
Normal Flow	<ol style="list-style-type: none"> 1. User chooses 'Beef Farming' 2. User chooses 'Medical' 3. User chooses 'Create Vet Visit' 4. User chooses 'Apply' 5. The table is updated with new information
Post Conditions	The new information will be displayed in the table
Alternative Flow	4. A prompt is shown displaying which information needs to be entered

Use Case ID	13
Name	Create Medical Expense
Actors	User
Description	The user can create a new medical expense
Trigger	User clicks on the create medical expense option in the option panel
Preconditions	User has logged into the application
Normal Flow	<ol style="list-style-type: none"> 1. User chooses 'Beef Farming' 2. User chooses 'Medical' 3. User chooses 'Create Medical Expense' 4. User chooses 'Apply' 5. The table is updated with new information
Post Conditions	The new information will be displayed in the table
Alternative Flow	4. A prompt is shown displaying which information needs to be entered

Use Case ID	15
Name	Feed Management
Actors	User
Description	The user can create a new feed expense
Trigger	User clicks on the feed management option in the option panel
Preconditions	User has logged into the application
Normal Flow	<ol style="list-style-type: none"> 1. User chooses 'Beef Farming' 2. User chooses 'Feed Management' 3. User chooses 'Apply' 4. The table is updated with new information
Post Conditions	The new information will be displayed in the table
Alternative Flow	3. A prompt is shown displaying which information needs to be entered

Use Case ID	16
Name	Generate Bar Chart
Actors	User
Description	The user can generate a bar chart
Trigger	User clicks on the bar chart option in the option panel User clicks on generate bar chart in the menu bar
Preconditions	User has logged into the application
Normal Flow	<ol style="list-style-type: none"> 1. User chooses 'Reports' 2. User chooses 'Bar Chart' 3. Will display a bar chart of the information <ol style="list-style-type: none"> 1. User chooses 'File' 2. User chooses 'New' 3. User chooses 'Bar Chart Report' 4. Will display a bar chart of the information
Post Conditions	The new information will be displayed in the table
Alternative Flow	There is no alternative flow

Use Case ID	17
Name	Generate Pie Chart
Actors	User
Description	The user can generate a pie chart
Trigger	User clicks on the pie chart option in the option panel User clicks on generate pie chart in the menu bar
Preconditions	User has logged into the application
Normal Flow	<ol style="list-style-type: none"> 1. User chooses 'Reports' 2. User chooses 'Pie Chart' 3. Will display a pie chart of the information <ol style="list-style-type: none"> 1. User chooses 'File' 2. User chooses 'New' 3. User chooses 'Pie Chart Report' 4. Will display a pie chart of the information
Post Conditions	The new information will be displayed in the table
Alternative Flow	There is no alternative flow

Use Case ID	18
Name	Generate PDF Report
Actors	User
Description	The user can generate a pdf report
Trigger	User clicks on the pdf report option in the option panel User clicks on generate pdf report in the menu bar
Preconditions	User has logged into the application
Normal Flow	<ol style="list-style-type: none"> 1. User chooses 'Reports' 2. User chooses 'PDF Report' 3. Will display the generated pdf file and create it in the correct folder <ol style="list-style-type: none"> 1. User chooses 'File' 2. User chooses 'New' 3. User chooses 'PDF Report' 4. Will display a pdf report of the information
Post Conditions	The new information will be displayed in the table
Alternative Flow	There is no alternative flow

3.1.2. Class Diagrams

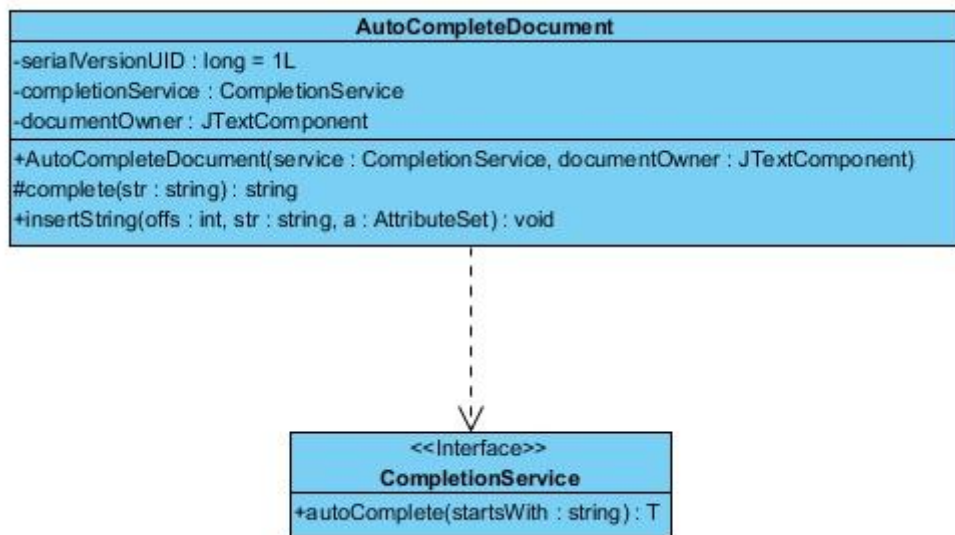


Figure 3

This is a class diagram of one of the external features I added to my project. This is the auto-complete feature which will retrieve information from the database, and compare the information entered into the text field to search for a unique sequence of characters. Once this unique sequence has been found, it will fill in the text field with the full sequence of characters.

Due to the number of class diagrams that are associated with this project, I have only included one. To view the remaining class diagrams, search the CD associated with this documentation.³ This is a Visual Paradigm project and requires Visual Paradigm to open it.

³ Farm Finance Manager/Documentation/Farm Finance Manager.vpp

3.1.3. Sequence Diagrams

Sequence diagrams show the action flow of a process.

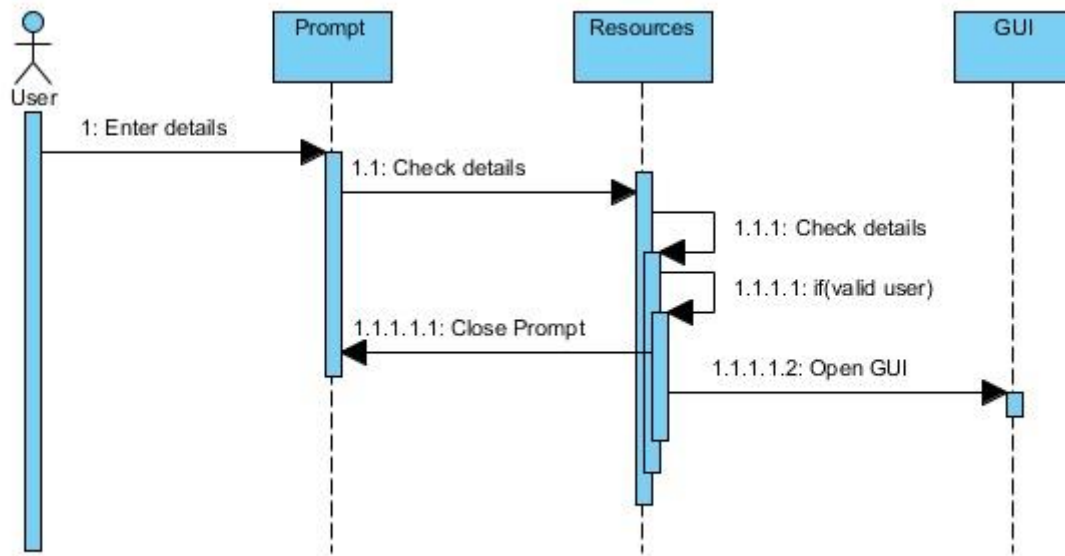


Figure 4

This sequence diagram shows the flow of the login action.

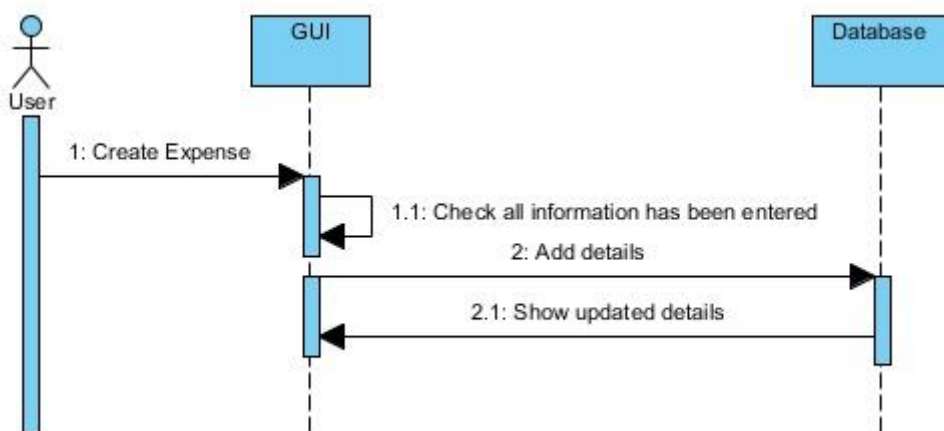


Figure 5

This sequence diagram show the flow of the add expense action.

4. Design

There were multiple aspects that were taken into consideration during the design process of this project, both in design of the overall Graphical User Interface (GUI) and the design of the code. I had looked at a few different types of finance managers that were already out on the market. I had done this research prior to try and work out the different types of designs and layouts used for their GUI. I also tried to find any commonalities between them. I took some of these designs and made my own improvements to them to suit the needs of this project. A few of these improvements are shown below. I also show some elements of code that show the workings of some features.

4.1. Graphical User Interface (GUI)

I have used many different elements to try and make my GUI navigable using easy mapping lists. The type of list I used is called a JList. This list was very useful for what was needed. My first draft of the option pane was to use buttons, however, this was immediately changed to something that looked more appealing and more importantly, easy to use.

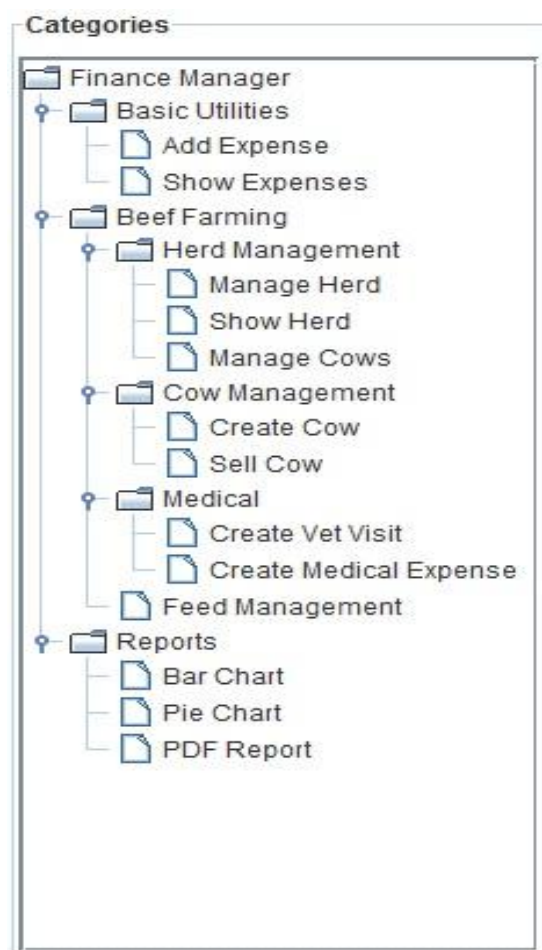


Figure 6

This JList allows for a much better view of the different functions that can be performed. This list is also collapsible so it can allow for options to be hidden. Each option has been grouped into their respective categories depending on what their functionality is.

The screenshot shows the 'Farm Finance Manager' application window. On the left is a 'Categories' sidebar with a tree view containing 'Finance Manager', 'Basic Utilities' (selected), 'Add Expense', 'Show Expenses', 'Beef Farming', and 'Reports'. The main area is the 'Module Panel' containing a table with the following data:

Type	Expense	Month	Year
Fertilizer Info			
	345.0	January	2013
	345.0	February	2013
	567.0	March	2013
Gas Info			
	567.0	January	2013
	50.0	February	2013
	50.0	March	2013
Labour Info			
	460.0	January	2013
	40.0	February	2013
	125.0	March	2013
Lighting Info			
	1072.0	January	2013
	799.0	February	2013
Machinery Info			
	247.0	January	2013
	350.0	February	2013
	240.0	March	2013
Mortgage Info			
	468.0	January	2013
	12.0	February	2013
	345.0	March	2013
Rent Info			
	363.0	January	2013
	12.0	February	2013
	34.0	March	2013
Slurry Info			

Figure 7

This is the GUI as a whole. This particular view shows the Basics Overview. This is not a comparison of the first colour chosen as the title bar colour, red, and the final chosen colour, yellow/orange. The table itself has undergone a few changes as well. The colours of the rows took quite a bit of time to work out what best suited the visual aesthetic of the application. I felt the grey and white left enough of a difference to allow for easy line reading. The yellow/orange title bar was chosen as it allowed for easier reading of the text inside. I originally had this as a red bar but the black text was not strong enough or bold enough to be read by someone who might not have perfect 20/20 vision. The current colour allows for enough of a contrast to make the text stand out more clearly.

This particular table of information had caused test reports due to the table not reloading the new information if something had been changed or added. These errors had me at a loss as I was unable to think of a way to reload this information without putting a button into the panel which I felt would take away from the layout to have just one simple button. The answer came to me later as I had to use a refresh method for certain panels in the beef farming module which resulted in me being able to use this same method in every class. (Ref 4.2.2) This method allowed me to attach a refresh function to the JList itself so anytime a selection was made; it would refresh that panel with the new information.

This screen shot shows one of the features that have been added to this application. This is the calendar button to the left of the 'Apply' button. The next image shows this button once it has been clicked.

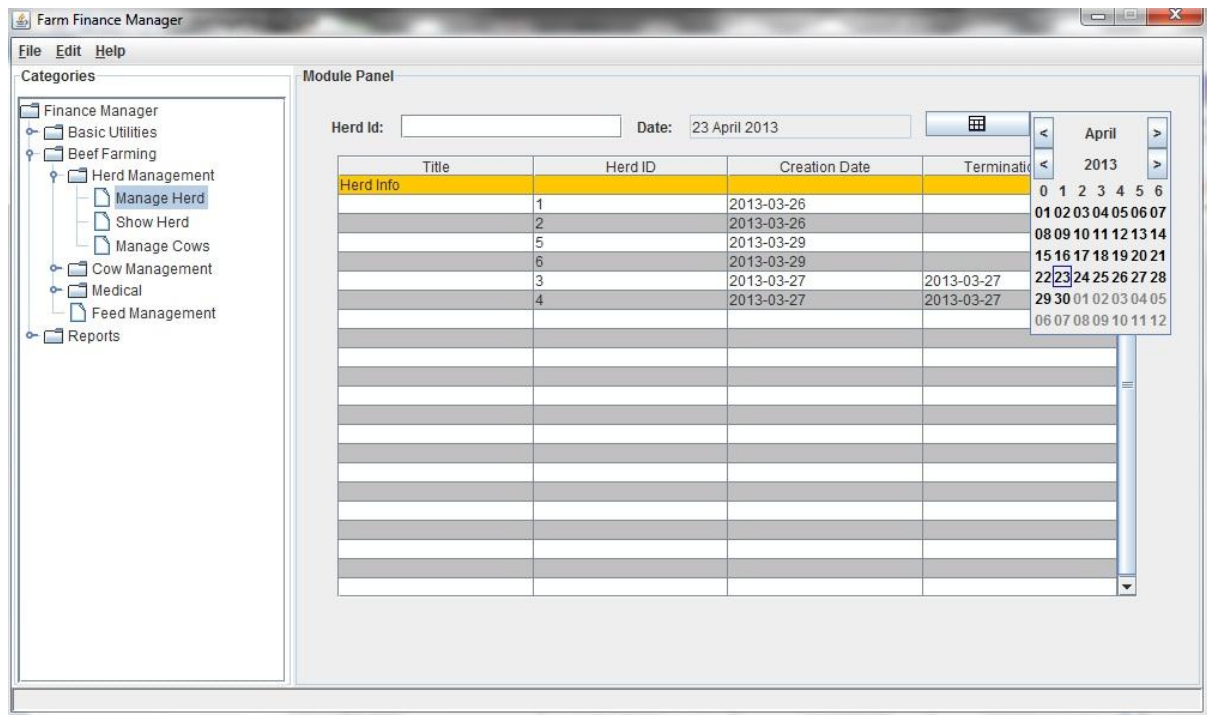


Figure 10

Once this button has been clicked, you can see a calendar appear which will allow the user to select a date. They can change the view to show different months or years as well. Also in this image you can see that the grey text field above will only show information chosen from the calendar button. This is to keep formatting correct once the date is to be added to the database. The user is not allowed to type information into this text field.

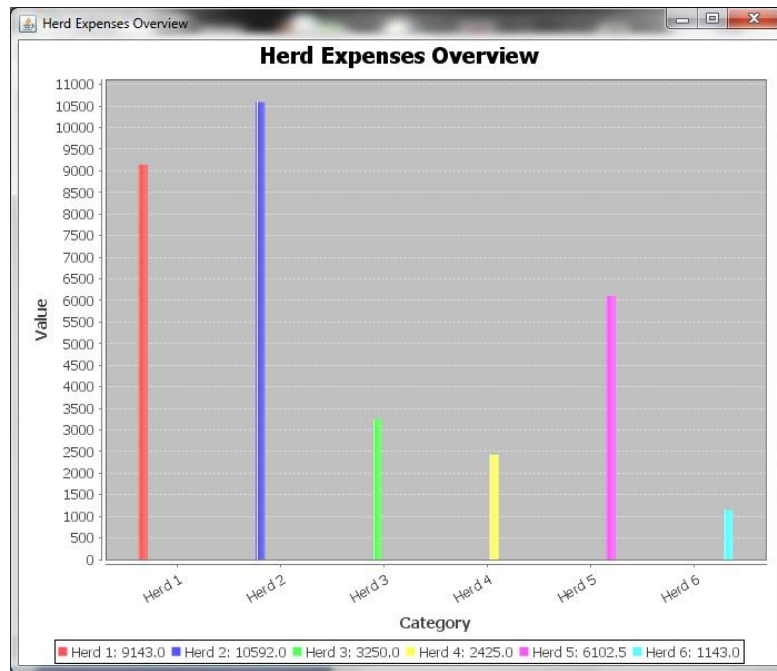


Figure 11

This is an example of the bar chart that is produced from the information in the database. This particular example shows the herd expenses. This shows all expenses across all herds, both active and inactive, to allow for comparison between the herds.

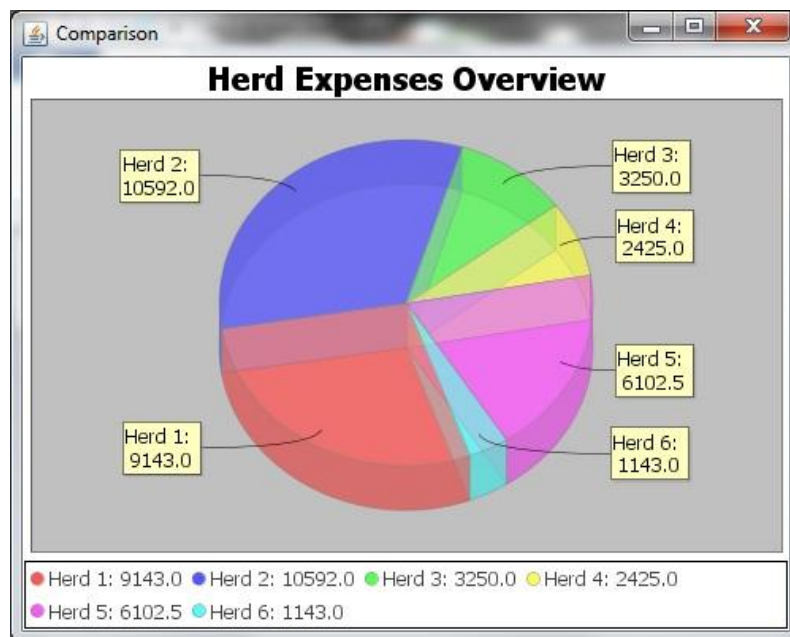


Figure 12

This is an example of the pie chart that is produced from the information in the database. This example shows the herd expenses. This chart shows the same information as the bar chart above in Figure 11. This pie chart was added to allow for a different view in which to view the information without having to look at it in table form or in the bar chart form. More chart styles will be added during future development.

4.2. Code Design

The structure of my code was meant to allow for easy maintenance as I tried to keep everything relevant to that particular module under that particular package. There are multiple packages in the project, each package having a range of classes in them. Some packages may only have one class and others may have upwards of twenty or more.

4.2.1. Code Structure

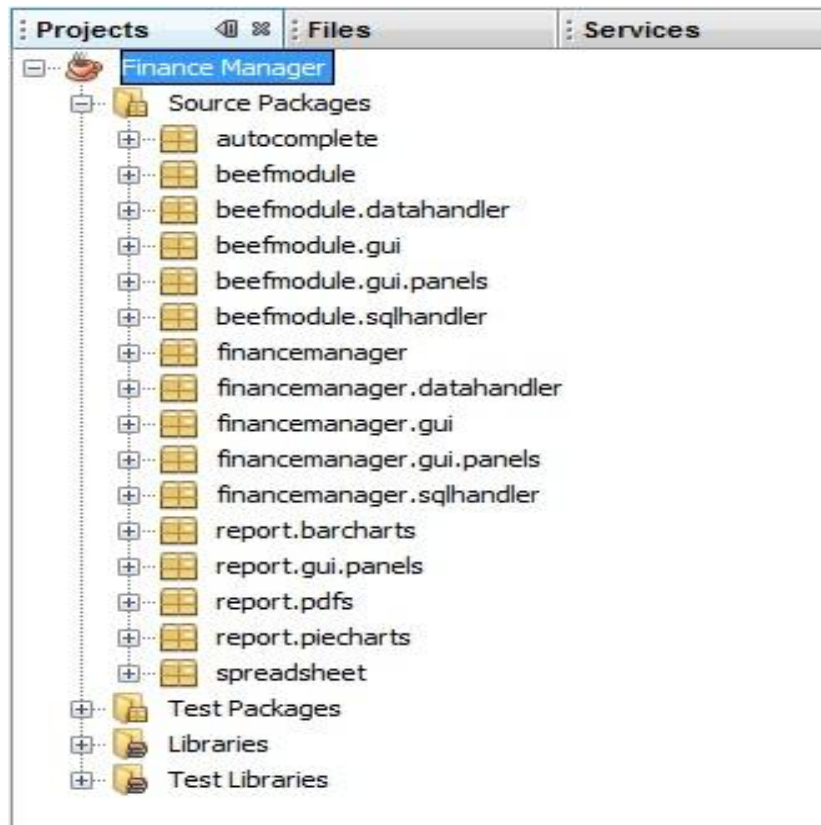


Figure 13

These packages in the image above are shown to be divided up into more specific packages to further group classes to more specific paths. I felt that this would make finding code to change or add to, or even the creation of new classes will be made easier by keeping these package names to the point and specific to what the classes within do. As can be seen from the above image, most package names between the modules share the same names. This is because I tried to keep a standard naming convention throughout the project.

4.2.2. Refresh

```
public void refresh() {  
    selectAll();  
    ss.updateAllBasicData(ss, fData, gData, lData, liData, maData, mData, rData, sData, wData);  
    gui.panel.revalidate();  
}
```

Figure 14

This refresh method was used in all display classes to refresh the information whether or not a change had occurred. This method was useful as it allowed for the program to call it from anywhere and refresh the information. This method has a structure take is the same in every class. The first command would call the database to get a fresh update on the information. The second command would call for the update of the table. This is the only command where a change would occur in the name and parameters depending on which refresh would be called. The third command would call for a revalidation of the GUI panel to repaint the information.

4.2.3. Progress Bar

```
class SaveThread implements Runnable {  
    public void run() {  
        int i = 0;  
        int next = 0;  
        gui.progbar.setVisible(true);  
        while(i<=98) { //Progressively increment variable i  
            gui.progbar.setValue(i); //Set value  
            gui.progbar.repaint(); //Refresh graphics  
            i += saveChoice(next);  
            next++;  
        }  
        gui.progbar.setValue(0); //Set value  
        gui.progbar.setVisible(false);  
    }  
}
```

Figure 15

I have added a progress bar to allow the user to know that the application has not frozen when the user saves the information. The above image shows a thread that manages the progress bar. This private class passes an incremented number with each loop to choose the next process which will be selected as the number passes through a switch statement. There were not many tutorials on how to properly set up a progress bar to fill with the correct percent of work done. The way I worked around this was to divide the fourteen processes that are used into one hundred. The closest integer came to be seven, so for each process, I just added seven to the total which allowed for the progress bar to jump a fraction with the completion of each process.

4.3. Database Design

The design of my database had changed throughout the course of implementing this project. Most changes were simply the addition of tables to add more features to the application. Others were the changing of data types.

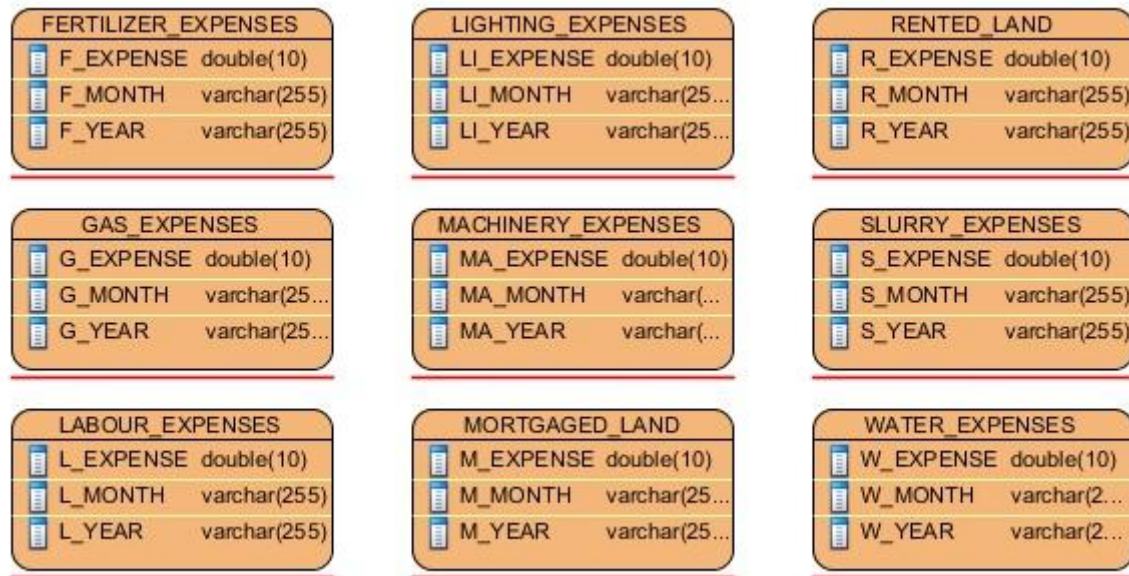


Figure 16

The above image shows the basic database tables. All of these tables are independent in that their only job is to hold information. Nothing connects these tables to any other table as they are just data holding tables.

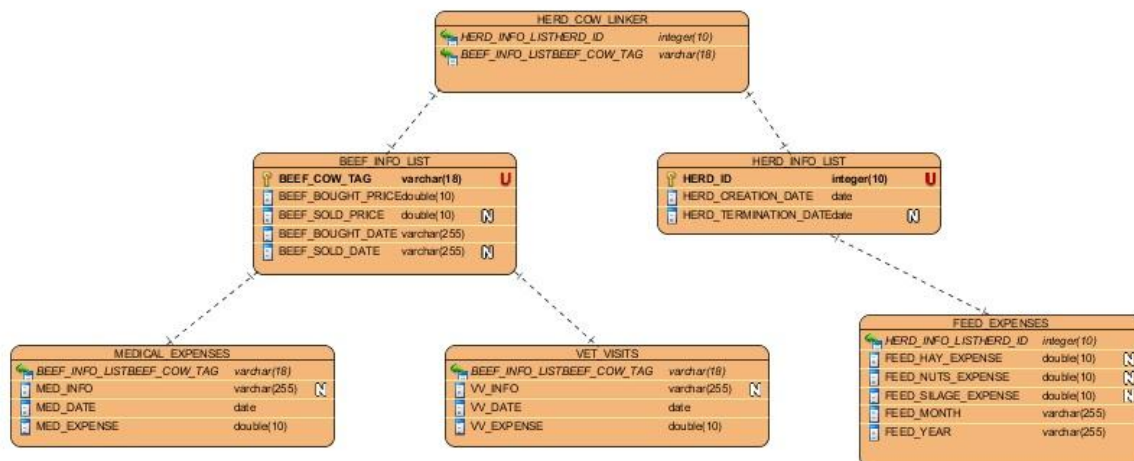


Figure 17

The above image shows the beef database tables. These tables are linked to one another through a hierarchy of tables as can be seen above. Unlike the basic tables in Figure 16, these tables need to be linked as they can have multiple expenses related to one individual entity in another table.

5. Implementation

During the course of implementing this project, I used several different methods to ensure that the project stayed on track and to maintain the overall objective during the duration of this projects timeline. I would regularly test new features that I added to the project and would also send regular project updates to outside testers. These testers helped to point out flaws that I either over looked or did not notice. Many reasons for the regular testing came due to the process I was using to develop code. I used the waterfall process for my implementation of this project as time was tight and I had a lot of ground to cover to make a fully working system.

5.1. Tools

5.1.1. Diagramming Tool – Visual Paradigm

When it comes to documenting a software project, the best way to help both the developer(s) and the future development staff to understand what has been done and what needs to be done is through the use of Universal Modelling Language (UML) diagrams. These models are representations of both proposed and implemented structures. This makes it possible for someone who is not familiar with the workings of the project to look at these UML diagrams and models to gain an understanding of the project. These diagrams act as a map for many things in the project. They can cover part of a system or cover the entire project. These diagrams can also cover sequences of events to show the workings behind the project.

These diagrams are essential for people outside of the project to gain an understanding of the project. As this project goes into future development, it will also be a critical factor if new developers take part in this project as it will allow them to gain an understanding of the project in a short amount of time.

The first tool that I was going to use was an online UML creator called Gliffy⁴. This however, although capable for small projects, is not suited for larger scale projects. This is why I used Visual Paradigm instead, as Visual Paradigm keeps all of the related diagrams to the project in groups, keeping all use case diagrams under one list, etc. I had used Visual Paradigm previously before in two modules during second year.

5.1.2. Integrated Development Environment - NetBeans

NetBeans was the IDE I used to implement my project and it also provided the database environment I used as well. This is a very useful IDE which allowed for an easy development process and allowed me to integrate the database into the project with relative ease. In second year used this IDE to incorporate a database into the project so I had some knowledge as how to do this for my fourth year project. This IDE suited my needs perfectly for this project as it allowed for me to interact with the code and the database from the one source. It also allowed me to test my Structured Query Language (SQL) code before I put it into my java code.

5.1.3. Database - Derby

For my data storage, I used the built in database with NetBeans IDE, Derby, which was to be used as I was using NetBeans to code my project. This made making and maintaining any changes relatively easy as NetBeans provides a GUI to show the database and it tables.

⁴ <http://www.gliffy.com/>

5.1.4. Web Hosting – ServersFree

The website I used for my web pages is a free web hosting site called ServersFree⁵. This was used to allow for the demonstration of online connectivity. This site provides a nice, easy to use control panel which allows for files to be uploaded and put into folders to create different Uniform Resource Locator (URL)'s from.

5.2. Languages

5.2.1. Main Implementation Language – Java

The main language that I have chosen for my project is Java as it has libraries to allow for embedded SQL and data retrieval from the database. It is also possible to add .jar files to add functionality to the project.

5.2.2. Other Languages – SQL, HTML

I have also used other languages for this project. I have used SQL to deal with data manipulation. I have also used HTML to make two web pages which I have uploaded to a free hosting site called ServersFree.

5.3. External Resources

5.3.1. JAR Files

I have used a number of .jar files in my project to allow for some functions to be used in my project. These .jar files were a big help in many aspects of my project. I used some to improve simple functionality like creating a calendar button to make a popup calendar, and others which were necessities for connection to a database.

List of .jar files	Type	Used for
autocomplete.jar	Feature	Fills in text field when a unique string has been found.
derby.jar	Essential	Used to make a connection to the database.
derbyclient.jar	Essential	Used to make a connection to the database.
derbytools.jar	Essential	Used to make a connection to the database.
designgridlayout-1.10.jar	Feature	Used as the layout manager.
itextpdf-5.4.0.jar	Feature	Used to create .pdf files for the finance report.
jcalendarbutton-1.4.5.jar	Feature	Shows calendar for user to select date.
jcommon-1.0.17.jar	Feature	Is a dependency of jfreechart.
jfreechart-1.0.14.jar	Feature	Generates different types of charts for reports.

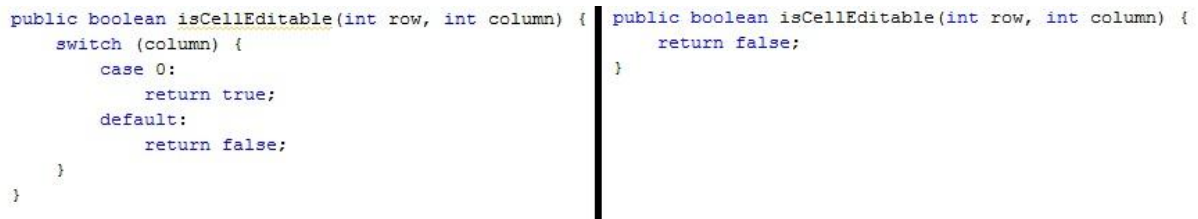
The table above shows a list of the .jar files used and whether the .jar file was used as part of a feature or if it was essential in fulfilling the requirements and the running of the program.

5.4. Difficulties

I had problems with the implementation of many features during this project. One problem being the use of JTables in the project. Even once these difficulties where stamped out, I was once again hindered trying to make the JTable work with the database to update

⁵ <http://www.serversfree.com/>

once a table's cells data has been edited. This caused for the creation of a second class to be called upon, 'Manage_Herds_Spreadsheet.java' which has the same purpose and design as 'Spreadsheet.java'. The only difference between the two is a small change to the code to allow for a column of cells to be editable.



```
public boolean isCellEditable(int row, int column) {  
    switch (column) {  
        case 0:  
            return true;  
        default:  
            return false;  
    }  
}  
  
public boolean isCellEditable(int row, int column) {  
    return false;  
}
```

Figure 18

This comparison in the above image shows the code from both classes. The left being the only change that was needed to get the JTable to do what I wanted it to do. This small change checks to see if the column is equal to the first column, and will set it as editable. By default it will make all other columns editable nature to be set to false.

6. Testing

During the course of the implementation phase, the project has been subject to multiple tests by myself, two outside testers and multiple usability testers. The usability testers chosen for this will be the end user to get relevant feedback back on the project. These are the people who will end up using the finished product. So to get and include the end user's wants and needs seem like it was the most logical choice to select them as the usability testers.

6.1. White-box Testing

*“White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality.”*⁶ White-box testing is testing that I part took in. This is where you examine the code to test that it does what is suppose to do and does not cause ill effects to the overall running of the project.

All of the tests I ran for this project were spontaneous, depending on whether or not a significant change was made. Testing of the system as a whole was left to the black-box testers as I only tested the specific part I was working with to make sure that what I was trying to add to the project worked the way it was suppose to. This type of testing could occur multiple times a day and sometimes may not happen for two or three days. The testing that I undertook was completely irregular.

The tests I ran usually came from adding something new into the project to make sure that all events were supposed to happen the way they were supposed to, with no unexpected results. There were a few additions to the code where it caused failure to the project which had worked previously. One of the most notable was when I was trying to create a table that would allow the user to edit the database by changing the data in the cell of the table. The solution to this was to make a new class with near identical code to allow for this functionality.

There were a few errors which had been known about, which I had put down as a low priority when I sent the project to be tested by outside testers. Some of these errors were the likes of number exception handling when taking in the amounts of the expenses. Another was the resizing of the window, in which it would leave a bevelled border in the old position of the status bar. These have both been resolved since.

6.2. Black-box Testing

*“Black-box testing is a method of software testing that examines the functionality of an application (e.g. what the software does) without peering into its internal structures or workings.”*⁷ Black-box testing allows for a more thorough testing of the project as it is their job to try and break the project.

⁶ http://en.wikipedia.org/wiki/White-box_testing

⁷ http://en.wikipedia.org/wiki/Black-box_testing

6.2.1. Outside Testers

I had two outside testers, Shane Kelliher and David Myślak. They both helped this project by testing the functionality of the updated projects that were sent to them. Their main job was to point out minor flaws to potentially fatal flaws in the project as I only tested the project in small areas at a time. These small areas were the areas I was currently working on. Many errors had been found, and thanks to this, serious problems had been stamped out. One of these problems was a design flaw; once the window was resized, it would leave a bar running across the old position of where the status bar was previously.

The Problem Report Form

Problem Report:	#1
Reporter:	David Myslak
Date:	24/02/2013
Program:	Farm Finance Manager
Release:	18/02/2013
Version:	0.0.1
Configuration:	-
Report type:	2 1 - Coding error 2 - Design issue 3 - Suggestion 4 - Documentation 5 - Query
Severity:	3 1 - Fatal 2 - Serious 3 - Minor
Attachments:	Yes
Description:	Screenshots (1a, 1b)
Problem summary:	GUI problem - covered button, duplicated bottom title field
Reproducible:	Yes
Reproduction steps:	Open "Beef Farming" view, maximize window.
Suggested fix:	On 1st screenshot I presented the place that buttons could be placed.

This problem report is from David Myślak⁸. The accompanying image shows the error.

⁸ Farm Finance Manager/Test Reports/From David/1.docx

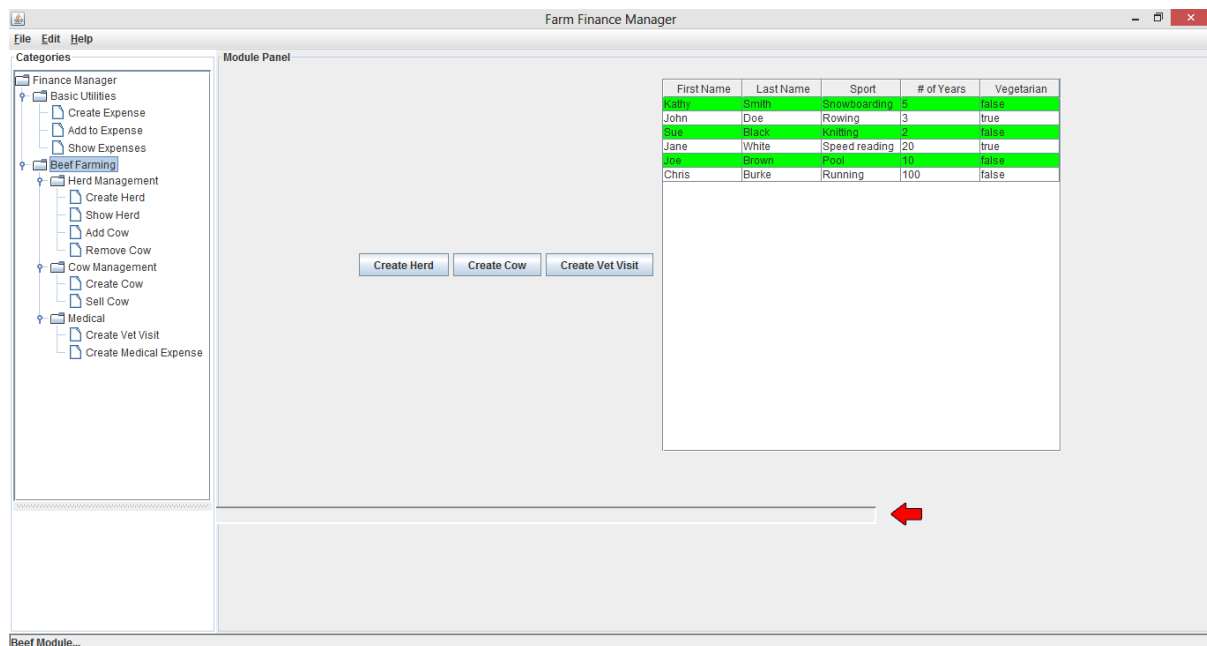


Figure 19

As can be seen in this above image, a bar is shown to be left in the old position of the status bar. This hollow bar splitting the window would remain for the duration of the programs lifespan. This was the first error pointed out by David, and although it did not cause a fatal flaw to the running of the project, or even a serious one, it was a serious design flaw.

6.2.2. Usability Testers

During my testing phase, I went and asked some potential end users to use my application and to point out anything that they thought could be improved, what could be taken out and what could be added in. These potential end users are farmers, and they are the same farmers that I had visited last semester to ask about what they might want to incorporate into a farm finance manager. There were a few things that they would have liked to expand on and add to the project, since they have seen a working version of the project.

The most popular suggestion was to change the dates in the 'Add Expense' option in the 'Basic Utilities' group. The suggestion was to change the date to the same method I had used in the 'Beef Farming' group where I used a calendar button to let the user choose the day, month and year all in the one go.

A change linked to the above was to make an option of two different views in the 'Show Expense' option in the 'Basic Utilities' group. The first view was to show each individual expense in a list under the type chosen, similar to how it is now, except, only to show every expense with the date the expense was added. The second view is to the one that is currently in place as they like the fact that it groups the expenses in a month together.

The above two suggestions were also suggested for the 'Feed Management' option in the 'Beef Farming' group. Like the problems above, they suggested the date to be changed to the calendar button, and for the two different types of views to show different types of information depending on what was chosen.

A suggested addition was to have a help function. By this, I believe that a digital user manual like what is offered with Microsoft Word and other applications would be helpful feature to be added. This is a feature that I did not think of and which is actually quite important to try and help the older generation of farmers out there as they may not be computer literate. This could be added by adding a digital user manual for a step by step of how to work the application, or by creating a web forum to let users help each other solve problems in a community based forum.

6.3. Bugs we Found Running

One bug that still remains is the closing of the bar chart reports. I believe this to be a bug unrelated to the code as the same code was used for both types of charts. The bug is; once the checkbox for any of the bar chart options is selected, a window showing the bar chart will open. However, the window is able to be closed using the close window button. This should not happen as it was designed to only close once the checkbox was unselected. This problem is not present in any of the pie chart windows as they will not close until the checkbox is unselected.

7. Review of the Requirements

This table shows the requirements and their status. Some of these requirements were added during the first phase of this project to add more functionality to the project if there was time. These requirements were the likes of online updates, profit and loss projections.

Functional Requirements:

Requirements	Status
Core finance tracking features.	Done
Capture expenses.	Done
Profit and loss.	Incomplete
Create a modular system to allow different components to install to the application.	Done
Online updates.	Done

Non-functional Requirements:

Requirements	Status
Should be reliable.	Done
Should be efficient with retrieving information.	Done
It should be easily maintainable.	Done
The graph screen show not look cluttered.	Done
The use of colours to help with graphs, spreadsheets and to add eye appeal to the Graphical User Interface (GUI).	Done
Spreadsheets will only show relevant information to what the user has asked it to return.	Done

The profit and loss projections was, once added, an ambitious requirement to add as I believed this to be a simple task to add appeal to the project. However, during the course of implementing this project, both time and a lack of knowledge in business or accounting caused me to stop the development of this requirement. This requirement has been kept and will be added to future developments.

The online updates requirement in this version is only an example using a bare minimum basic html website to show that this project can connect to the internet to check for updates. However, no updates will be available at the moment. This will be a priority once there is a new module has been created as this will allow for the user to have a choice which they would like to use.

8. Future Developments

Some future developments that will come in this project will be the addition of the profit and loss projection function which was not possible to get done during this phase of the project. This will be a priority to get done during the next phase of the project.

A future development that must be considered is the option of more styles of charts. At the moment there are two types of chart, bar and pie. From user testing, it has been asked if there will be more options added. For example, having the option to make the bar chart horizontal instead of vertical.

The option to add a basic expense needs to be built upon, particularly the choices, Labour and Machinery. These options need to have a few built in options that are relevant to only themselves. The reason for this is that each may have a different reason for that expense and it is just to break up the expenses that little bit more. For example, Labour can be broken up into the different employees. Machinery can be broken up into the different types of expenses, such as fuel, services, etc.

There were a few suggestions during the user testing to add different views while showing the expenses. These additions I believe will help improve the usability of the project and will allow more thorough inspection of the recorded expenses.

A help function will be added which will contain a digital user manual, contact information like an email and a website forum with a FAQ section. This will help users who are not computer savvy to see how the application works in a step by step explanation of each function. This will be built upon with each new function added to the project.

9. Abbreviations

FAQ	- Frequently Asked Questions
GUI	- Graphical User Interface
IDE	- Integrated Development Environment
SQL	- Structured Query Language
UML	- Universal Modelling Language
URL	- Uniform Resource Locator

10. References

10.1. Websites/CD URL's

<http://programmers.stackexchange.com/questions/144530/how-important-are-uml-diagrams-for-a-successful-project>

http://wiki.answers.com/Q/What_are_the_advantage_of_UML

Farm Finance Manager/Documentation/Farm Finance Manager.vpp

<http://www.gliffy.com/>

<http://www.serversfree.com/>

http://en.wikipedia.org/wiki/White-box_testing

http://en.wikipedia.org/wiki/Black-box_testing

Farm Finance Manager/Test Reports/From David/1.docx

10.2. Images

Figure 1	5
Figure 2	7
Figure 3	13
Figure 4	14
Figure 5	14
Figure 6	15
Figure 7	16
Figure 8	17
Figure 9	17
Figure 10	18
Figure 11	19
Figure 12	19
Figure 13	20
Figure 14	21
Figure 15	21
Figure 16	22
Figure 17	22
Figure 18	25
Figure 19	28