|  |
| --- |
| **Farm Finance Manager** |
| Finance Manager |
|  |
| This is the research phase of my fourth year project. In this documentation, I look into similar products, possible designs and the possible functions that could be used. |
|  |
| **R00044989 – Thomas Donegan** |
| **12/6/2012** |
|  |

Table of Contents

[1. Introduction 3](#_Toc342504456)

[2. Project Proposal 4](#_Toc342504457)

[3. Research 5](#_Toc342504458)

[3.1. Required Functions 5](#_Toc342504459)

[3.2. Programming Language Research 5](#_Toc342504460)

[3.2.1. Java 5](#_Toc342504461)

[3.2.2. C++ 5](#_Toc342504462)

[3.2.3. SQL 5](#_Toc342504463)

[3.3. Research for Similar Products 5](#_Toc342504464)

[3.3.1. AceMoney 5](#_Toc342504465)

[3.3.2. Quicken Starter Edition 5](#_Toc342504466)

[3.3.3. You Need A Budget 5](#_Toc342504467)

[3.4. Common Elements of Current Finance Managers 5](#_Toc342504468)

[4. Requirements 5](#_Toc342504469)

[4.1. Functional Requirements 5](#_Toc342504470)

[4.2. Non-Functional Requirements 5](#_Toc342504471)

[4.3. Data Requirements 5](#_Toc342504472)

[4.4. Project Specification 5](#_Toc342504473)

[4.5. Use Case Diagrams 5](#_Toc342504474)

[4.6. Data Dictionary 5](#_Toc342504475)

[5. Design 5](#_Toc342504476)

[6. Schedule and Risks 5](#_Toc342504477)

[7. Conclusion 5](#_Toc342504478)

[8. References 5](#_Toc342504479)

# 1. Introduction

This project came about after I was given an assignment to research and develop a project of my own undertaking for my Fourth Year project. The idea was conceived in early September as I was talking to my uncles who are beef farmers for if they had any ideas to what would make their life easier it their trade. They suggested a number of things, but one of them stood out to me, to be able to see where their money was going and how much money they would get in return. The reason this stood out to me was because I had heard of finance managers for budgeting the household budget or for an organisation, but I never heard of one to track the expenses of a farm.

This project is to make a ‘Farm Finance Manager’ that will allow for a farmer to kept track of where their money is going, for example how much money they spend on diesel for the vehicles, or for how much they spend on feed for the animals. This is not all that this finance manager is limited to as it will hopefully include profit projections based on current selling prices of life stock, dairy produce and crop sales.

This project can cover all elements to manage the finances of a farm. Each element will be a module that can be separately installed to allow a farmer to only install elements related to their needs. These elements will be able to be inserted and removed without relying on another element to work. This will allow for less memory to be used on the user’s computer and allow take out the possible trouble the user may find with the extra functions that are not relevant to their needs.

# 2. Project Proposal

**1. Name:**

Thomas Donegan

**2. Contact Details:**

thomas.donegan@mycit.ie, 0851139949, 37 Uam Var Ave. Bishopstown

**3. Project Title:**

Finance manager.

**4. Client Company and Contact Person (if applicable):**

N/A

**5. Project Background:**

This project is to help farmers manage their finances and to see where they are making and losing their money depending on the cattle they sell. It will also record other expenditures such as the amount spent on food, labour, etc. This may help the farmer’s project future earnings based on past sales or to help them to improve earnings by viewing past loses.

**6. Overall Project Goal:**

The project goal is to have a working system that can actually be used to help farmers maintain their expenditures and to see where they are spending and making their money. More goals include having a modular based application where different types of finance recording will come from the different types of farming, such as beef farming, dairy farming, tillage farming. These will each be kept as different modules that can be downloaded if the farmer should so chose.

**7. Objectives**:

i. Design and build a farm finance manager.

ii. Capture expenses.

iii. Track each animal from purchase to sale.

iv. Profit and loss projections.

v. Create a modular system to allow different components to install to the application.

vi. Online updates.

**8. Development Tools:**

I will be using Java for the main programming of this project. Other languages that may be needed are SQL for a database. NetBeans will be the IDE and the built in database which NetBeans uses called Derby. Other tools may be required for some of the objectives, such as online updates.

**9. Learning Outcomes:**

i. I believe I will learn about time management trying to keep to deadlines.

ii. I will hopefully become more proficient at programming.

iii. To learn how to prepare a business model as I am making this project with the intention that it can be sold.

**10. Software/Hardware Requirements:**

A Windows PC or laptop will be used to test software as I will be making a desktop application which will be run on either a Windows PC or laptop. The reason for it being windows is because it is the most common home computer operating system.

**Notes on the Project Proposal:**

I plan to use this project as something that I can place on my C.V.. This project will hopefully allow me to learn new things about embedding database SQL code into the java application code, and how to correctly modularise components from a big project. This project is to function as a modular application that will allow for the user to install different types of modules that will suit the farmer’s needs depending on what type of farming they take part in.

The beef tracker module is aimed at beef farmers. The overall plan and design for the beef tracker is to make it so that the farmer can check how much money was made from the herd and to open up the herd and see how much money was made from the individual cow. This is to allow the farmer to see why a particular herd didn’t make as much money as the other herds. The reason why this is being done is that a few cows may have needed extra visits from the vet, and this could have brought the overall profit from that herd down.

# 3. Research

The reason why I have researched into this project is to see where I can improve on existing projects and develop a new finance manager design to target a specific customer base, the local farmer. This finance manager is going to help keep track of farm expenses such as the cost of feed used for the cattle, and how much the cow was bought and sold for. This will help with one of the features, profit and loss projection. This will help the farmer see an estimated profit or loss depending on current rates of tax, selling prices and expenditures.

This research should help me with problems I may face at a later stage in my project development and can help me to highlight issues that I will face so that I can be better prepared to tackle them, if not to get rid of them entirely when the time comes. It will also allow for a better project scope as I can define what can and will be done in the project. I will also be able to define what won’t be done. This should allow for not assumptions to be made or for false expectations of what will be delivered.

I have broken up the research that I have done into several parts. Reason being was to reduce clutter and information overload. Each part deals with a specific topic that contributes to the understanding of why I am making this project and how I plan to do it. It will also hopefully show some examples of how it will look as research into other similar products will show some design ideas that I would like to incorporate into my project.

## 3.1. Required Functions

Some of the required functions that this project needs to meet the user’s needs;

* Keep track of the herd.
* Record information correctly.
* Display correct information in a readable graph.
* Show break down of expenses.
* Keep track of individual cattle.

These functions will need to be accurate in both their input and output to work correctly and to give accurate information to the user. It is the accurate information that the user is looking for and will stop using the application if it keeps giving them inaccurate information.

One way to stop the input of inaccurate information is to check the input the user enters and return an error to the user to re-enter the information again. This is probably one of the most important required function as one input error could throw all of the information out and with more information errors being enter, the more the information will become uncontrollably wrong.

Information will need to be stored in the correct database table so that once the information is called it can display all of the relevant information. This is important as it will mean less work for the Database Management System (DBMS) as it will not have to call more information tables than it needs to. It will also cut down on processing time. Another reason why this is important is because it allows the searching for information to be much easier as you can look at one table under the category you know the information is under rather than looking through all of the tables to find the information you want.

The required function to display the information on a graph I believe to be import to attract people to use the product using the common expression, ‘*A picture paints a thousand words.*’ The reason I say this is because the user will be able to see all of the information breakdowns from a quick glance and zone in to what they are looking for quickly. This could be the biggest slice of a pie chart or the tallest bar on a bar chart. The reason it is easier to zone in on a specific piece of information using a graph is because you see all of the information in a neat bundle, where if you where to look at the information in a text format, you may need to scan down through several rows of information to find what it is you are looking for.

The breakdown of expenses is more of a required function from the user perspective rather than the functional or operational requirements. The reason for this is that the application will function just fine without showing the breakdown of expenses, however the user would like to see this breakdown as it can be helpful for future planning. If the user was to see just the total cost or profit, and no breakdown of how it came to be that total, they would be unable to remedy to problem for a loss and or unable to continue the solution to a profit. These breakdowns need to be informative to make this application unique and to stand out from current finance managers.

The required function for projecting profit and loss based on past and current rates will be the hardest function to implement. However, it is also probably the most useful function that the user will want to use. Everyone wants to have some prior knowledge before purchasing something, like ‘*how much money will I have left.*’ This function will need to be as close to accurate as can be to help future planning for money expenses and for repeated use by the customers.

Once I first looked into researching what farmers from other countries spend money on, i.e. expenses for machinery, feed, etc. I have found that in America, they have ranch insurance. This is something that I may have overlooked as farms are treated like a business and would also have insurance on the farm should anything happen. This may be another function that will need to be added. However, I am unsure at present whether to have it placed in the skeleton code as a basic function or to have it in its own module with other basic functions like this, like the electricity used to power the farm, the expenses to pay for labour and fuel for machinery.

## 3.2. Programming Language Research

These are some of the languages that could possibly be used in this project during the implementation phase.

* Java
* C++
* SQL

### 3.2.1. Java

Java is a programming language developed in 1995 by James Gosling at Sun Microsystems. Applications in this language are compiled using a JVM (Java Virtual Machine), using Java byte code, an intermediate rather than to platform specific machine code. This makes the Java platform independent and able to run on any computer that has a JVM installed. Java derives much of its syntax from the C and C++ languages, but has fewer low-level facilities than either of them. Java is a class based, object-oriented language that once complied for one platform, it does not need to be recompiled to run on another. Java is one of the most popular programming languages in use, and is particularly popular for client-server applications.[[1]](#footnote-1)

My experience with Java comes from two years of college based projects which include desktop based applications along with some embed SQL code to be used alongside a database, and web-based projects using applets along with JSP and XML/HTML to make the holder page to contain the applet. Java is probably the language that I am the most comfortable with once it comes to making GUI’s (Graphical User Interface) as I have more knowledge in this language to make one.

### 3.2.2. C++

C++ is a general-purpose programming language. It is regarded as an intermediate-level language, as it comprises a combination of both high-level and low-level language features. It was developed by Bjarne Stroustrup in 1979 at Bell Labs. It was originally named ‘C with Classes’ as it was essentially the same language as C, only that it added object-oriented features, such as classes and other enhancements. It was renamed in 1983. C++ is one of the most popular programming languages in use and is implemented in a wide variety of hardware and operating systems platforms. It is an efficient complier to native code and its application domains include systems software, application software, embedded software and entertainment software such as video games. C++ can be used across platforms, as it avoids features that are platform specific or not general purpose.[[2]](#footnote-2)

My experience with C++ comes from two years of college based projects which all dealt with low level coding for operating systems and real time systems. I believe that this could be a more useful language to use for this project, however I do not believe myself to be experienced enough to use it to the extent that I would like. Therefore I do not plan to use C++ in this project unless I have to in the unforeseeable future.

### 3.2.3. SQL

“*SQL or Structured Query Language is a special-purpose programming language designed for managing data in relational database management systems.*” SQL is the language of choice for database management. There are many different dialects in the SQL language with many different database managers. These differences usually are small and include syntax differences. SQL is based on relational algebra and tuple relational calculus. It is able to insert, query, update and delete data from the database. It was initially developed by IBM by Donald D. Chamberlin and Raymond F. Boyce in the early 1970’s. It was originally called SEQUEL. SQL is a very high level code that uses a DBMS to interpret SQL code and translate it to a language like C++.

I will use SQL for the database manipulation. I have experience with the language for manipulating the database directly and I have even used it as embed code in a Java project I have made to manipulate a database using triggers. This is how I plan to use SQL statements to manipulate the database with the information the user is looking for.

## 3.3. Research for Similar Products

For my research I looked into other similar products to give myself ideas of how the GUI should look, how the interactions between machine and human work, i.e. does the person have to choose the maths formula each time or does the software decide which one is needed. This will be helpful in deciding on how I will implement my project. However, all the finance managers that I could find were all online, i.e. websites. There was no free downloadable desktop application. As I am planning to create a desktop application rather than an online application, I tried searching for the finance managing software top ten for desktops.

From a search online for the top ten finance manager software[[3]](#footnote-3), AceMoney (AM)[[4]](#footnote-4), Quicken Starter Edition (QSE)[[5]](#footnote-5), and You Need A Budget (YNAB)[[6]](#footnote-6) were the three top finance managers that could be bought online. AM and QSE charged $39.99 while YNAB was able to sell for $60.00. To compare all three, on both ratings and what they offer, YNAB excels in the areas it does offer while AM offers a wider range of services at above average ratings.

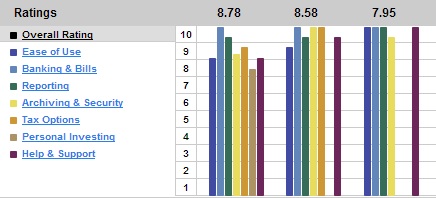


Fig. 1

1. AceMoney received a rating of 8.78.
2. Quicken Starter Edition received a rating of 8.58.
3. You Need A Budget received a rating of 7.95.

The first is AceMoney. They offer everything a finance manager should offer and have a rating of seven and higher for all of their services. They sell their product for $39.99 which, linked with the wide range of services the product includes, may contribute to it receiving the number one rank.

The second is Quicken Starter Edition. They offer every service except for Personal Investing. They however, even though they are second best, they have a rating of eight and higher across all of their services. Quicken Starter Edition also sells their product for $39.99. I believe that because it is the same price as AceMoney, although a better all rounder, it came in second to AceMoney due to one service, Personal Investing, was not included with the product.

The third is You Need A Budget. They offer very little in regards to the first two, however, they excel in everything they do offer. Everything is a ten except for Archiving & Security which is a nine. You Need A Budget is also the higher priced seller in comparison to the other two. This is leaves me to believe that it is a better product as they can sell their product for a 50% mark-up on their competitors prices and still offer less services. Based only on the services You Need A Budget offers, I believe it would come first out of the three top products due to its higher ratings as the ranks came from an average of all six services and if a product had a missing service, this would impact the overall average.

### 3.3.1. AceMoney

“*Using AceMoney, personal financial software for Mac OS X or Windows, to organize your personal finances is quick and easy. All the features expected from home budget or even small-business accounting software are supported.*”[[7]](#footnote-7) AceMoney supports different account types like checking, savings, etc. It even has a function to create your own custom type of account. AceMoney can also be configured to download your bank transactions to categorize them and give you updated balances of your home budget. AceMoney is an alternative to Quicken Personal Finance software.



Fig. 2

This image is the shows the graphs used by AceMoney. First looking at it, it looks too cluttered. I can only assume that this was chosen by the user to show this much graphs on the one page. It is hard to make sense of these graphs so I do not think I will be using AceMoney as a reference once it comes to designing my graph layouts.

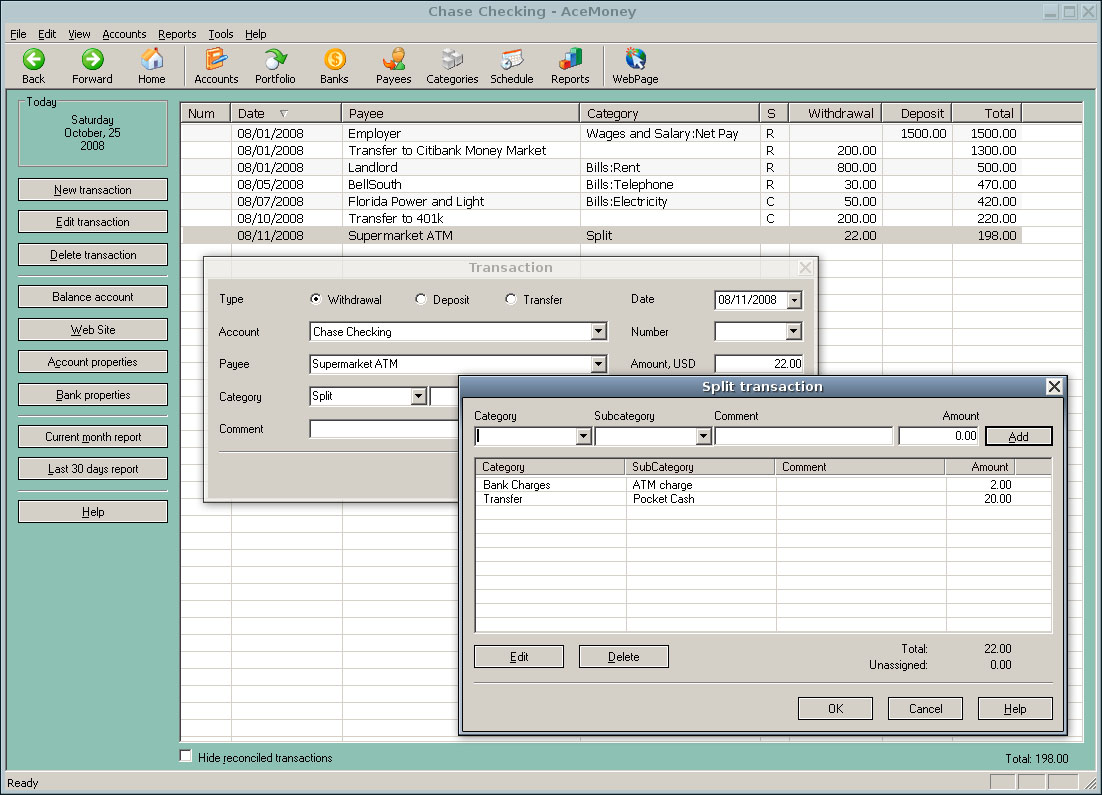


Fig. 3

This screenshot of AceMoney for Linux shows the possible features I may need to include in my application, although some will need to be removed as they will be of no use just as some different features will need to be added to make my application task specific to farm management.

I tried to cross reference this product on the company’s website; however, one did not exist to give information about the product. The website only showed reviews from customers and a buy now link to download the software. For these reasons most of my information about the product, AceMoney came from a Wikipedia source.

### 3.3.2. Quicken Starter Edition

The Quicken Starter Edition was first developed by Intuit, Inc. They have “*different (and incompatible) versions of Quicken run on Windows and Macintosh systems.*”[[8]](#footnote-8) They had previous versions that ran on DOS. They have several versions of Quicken, from starter to deluxe to business level finance software. Quicken’s major marketplace is North America and is sold to suit the needs of the American and Canadian people. They have other countries for which they sell to as well like Hong Kong, Germany, etc.

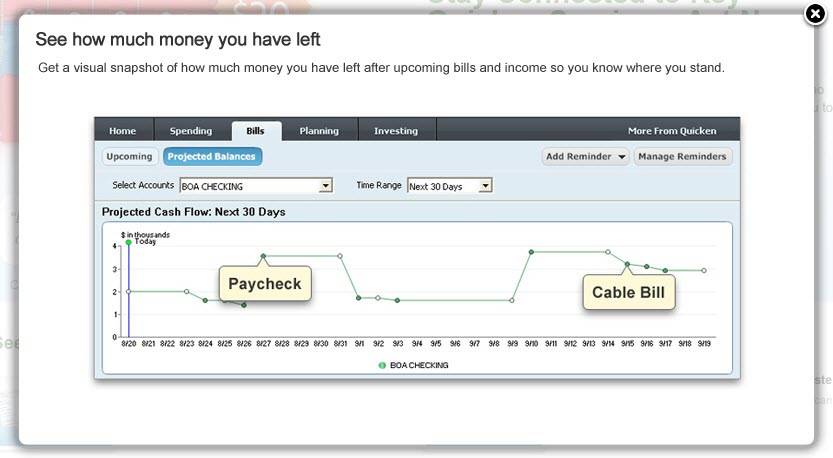


Fig. 4

The Quicken Starter Edition allows for profit and lost projections as you can input highlight dates and give them values of income, bills etc. This lets the user see where they stand in terms of how much money they have. This can be very useful feature to add to my program for definite income and expenses as farm payments, employee payments and feed payments will all be regular, fixed, periodic payments the farmer will have to pay. This feature could allow the farmer to see when it is a good time to invest money and when is a good time to be conservative with the money.

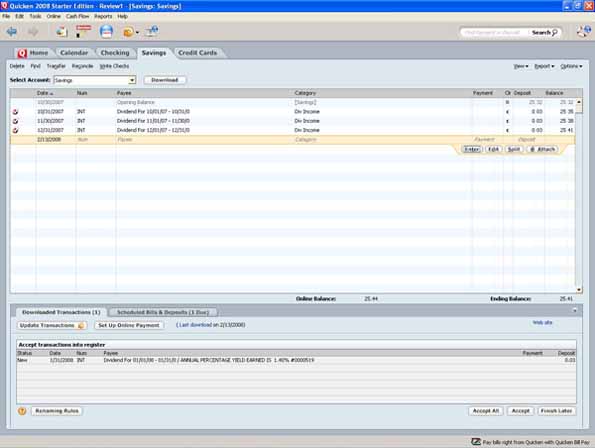


Fig. 5

This image has similar features to AceMoney in terms functionality, however, in terms of how it looks, the only thing similar is the spreadsheet layout for the information. Quicken uses a tabular interface that works like bookmarks. Each tab is given a name to clearly define what the tab holds and then once chosen, the main screen changes to that view. This is, for example, like how one tab on a book may show a picture on the page where another may show only text. This is similar as it does not need to contain similar layouts or information. For example, this image shows a checking account and a savings account but it also has a tab for a calendar.

I tried to cross reference this product on the company’s website; however, one did not exist to give information about the product. The website only showed reviews from customers and a buy now link to download the software. For these reasons most of my information about the product, Quicken Starter Edition came from a Wikipedia source.

### 3.3.3. You Need A Budget

“*You Need a Budget (YNAB) is a multi-platform personal budgeting program available via digital download or traditional CD-ROM.*”[[9]](#footnote-9) YNAB is designed for Linux and Windows, running on Adobe AIR. There were previous versions created in Microsoft Excel and Open Office Calc. It was designed to work closely to the companies budgeting methodology.

“*The YNAB Methodology uses four rules of cash flow management:*

*Rule # 1: Give Every Dollar a Job (Standard zero-based budgeting. Basically deciding where each and every dollar available needs to be saved or spent.)*

*Rule # 2: Save for a Rainy Day (This rule forces a person to accept the realities of life in the sense that there will always be "Rainy Days" when ample amounts of cash are needed in hurry for repairs or other emergencies. By allocating money to these types of budget Categories you can stave off panic over what to do when those events occur.)*

*Rule # 3: Roll with the Punches (Since even with a forward looking and planning system you can sometimes overspend in a Budget Category, YNAB has a method that spreads that overage across the next month's budget where it can be absorbed more readily. Since (by design) a person has money reserves at hand (The Buffer) the overage can be covered easily.*

*Rule # 4: Stop Living Pay check to Pay check (This is the ideal final state. A person will have reached a position where they have lived a month without spending income received in that month. This income is then allocated against budget categories for the month ahead.)*”[[10]](#footnote-10)

YNAB allows for proactive budgeting as it lets you look forward to plan for money before it is spend. They do this by using what they call a ‘Buffer’. This buffer equals one full months worth of expenses. This way a specific amount of money is known to have been saved and can be used. It also provides short term budgeting and long term budgeting. This can be useful if you can tie both short term and long term budgets together so you can budget for an overall goal while in being able to see if you went wild on a family holiday, for what affect it would have on your long term budget.

YNAB have also released mobile applications for both the iPhone and Android phones. They are not standalone budgeting applications, but instead designed the user to keep track of expenses and then upload these to the desktop application once they get home.



Fig. 6

This image is from YNAB run on the Mac OS. It shows a simple graph while also including a detailed legend with some stats at the top of the page. This looks easy to read and easy to understand for the user who may not have much financial knowledge.

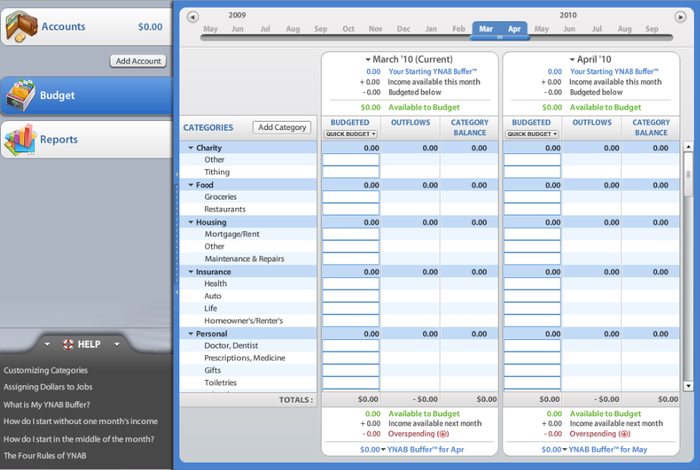


Fig. 7

This layout is clean, well organized, and collected. It contains all the information that a person would want to see in an easy neat format. Navigation looks particularly easy as it uses a slider to choose the months to view. In the categories panel, the categories are collapsible. This will allow for a lot of categories and to see the overview of that category while allow for the user to expand it and to see a more detailed breakdown of the expenses or income for that particular category.

I tried to cross reference this product on the company’s website, however, although one did exist, it gave information about themselves and their method to help save you, the customer, money, not the product. The website also showed reviews from customers and a buy now link to download the software. For these reasons most of my information about the product, You Need A Budget came from a Wikipedia source.

## 3.4. Common Elements of Current Finance Managers

The common elements of finance managers are that they manage the user’s finances. This can be understood either as managing past or future expenses and income. Most of the Finance Managers that I have researched do not have the function to project gain or loss. They only record past expenses and income.

Most have the same kind of spreadsheet look about them as the information is displayed on a main screen in a table format. There does not seem to be any distinguishing features about the display of the textual information, unless I was to mention about You Need A Budget’s collapsible categories that can either show the breakdown of expenses for that category or just show the overview of that category.

Of the three that I have researched in detail, the one that stands out most in my mind is You Need A Budget. They seem to have a really easy to use interface that displays all relevant information. The only problem in comparison to AceMoney and Quicken Starter Edition is that is does not deal with taxes. AceMoney’s interface I believe to be the worst for starters as it looks very cluttered, although it does show all the relevant information the user wants to see, the user will need to be familiar with the product to find what they a looking for. Quicken Starter Edition’s interface has a nice break down for categories using a tab function like a hardcopy accounts book to categorize the accounts, however this does not allow to compare accounts from a different category.

Quicken Starter Edition has a very nice profit and loss projection graph that is easy to tell where the money stands, however, it does not give much detail about the finances overall. AceMoney has an awful amount of graphs that make it hard to find what you are looking for. This does not allow for the user to find the information they want in an easy and quick manner. Although it does show all of the information, I believe it to be too cluttered to be of use. You Need A Budget has a nice, clearly laid out graph that allows the user to choose what information they want to see by using a checkbox on the textual information above the graph to graph the information in its own colour. This allows the user to see a graph for one specific piece of information or all of the information if they should so choose.

Most finance managers say that the work on two or more Operating Systems. This helps to broaden the customer base that the finished product could be sold to. However, it can cause limitations in the product. I plan to allow for my project to work on multiple platforms, and I will test the application at the testing phase for what systems it can run on. However, should it only be able to run on one Operating System, I should choose Windows as I believe most farmers will run a Windows Operating System as it has 70.39% of the market share across four of the Operating Systems they sell.

# 4. 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.

## 4.1. Functional Requirements

“*In requirements engineering, a functional requirements defines a function of a software system or its component.*”[[11]](#footnote-11) Functional requirements may be calculations or data manipulation that needs the system to accomplish this.

1. Core finance tracking features.
2. To be able to attach modules without

ii. Capture expenses.

iii. Track each animal from purchase to sale.

iv. Profit and loss projections.

v. Create a modular system to allow different components to install to the application.

vi. Online updates.

These requirements are derived from the objectives outlined in the project proposal. Each one of these requirements will help to achieve measureable goals and help to plan and schedule the projects timeline over the period given from 7th of January after I finish up with my exams until the 19th of April to complete the project.

## 4.2. Non-Functional Requirements

“*In requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours.*”[[12]](#footnote-12) These non-functional requirements focus on a different aspect of the project. Non-functional requirements focus on how a project should be, not what function it is required to perform, but how quick it should finish that function.

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

These non-functional requirements are needed if this application is to establish a consistent client base to use this product. The first three requirements listed are also expected from any and all software projects. The rest of the requirements are solely for this project.

## 4.3. Data Requirements

“*Data modelling is a process used to define and analyze data requirements needed to support the business processes within the scope of corresponding information systems in organizations. Therefore, the process of data modeling involves professional data modelers working closely with business stakeholders, as well as potential users of the information system.*”[[13]](#footnote-13)

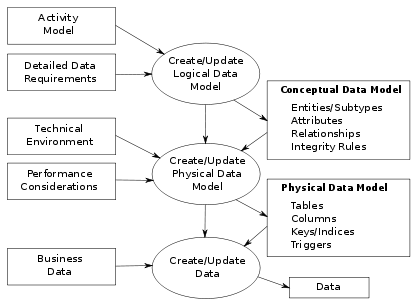


Fig. 8

This diagram shows the data modelling process. The figure illustrates the way data models are developed and how they are used today. These diagrams are usually based on the data requirements for an application that is being developed. These data requirements will be entered into my documentation in the implementation phase.

## 4.4. Project Specification

This project will include the following:

1. A login page for permitted users only to access information.
2. A base skeleton code which will include the Graphical User Interface (GUI).
3. The beef tracker module which will attach to the skeleton code and the database.
4. A function for profit and loss projections.
5. The function to display the information in a graph of the user’s choice, i.e. bar chart, pie chart.

This project will NOT include the following:

1. The module for tillage farming.
2. The module for dairy farming.
3. A function to update tax rates, sell rates, etc. from an online source.

Some elements have yet to be decided whether to be implemented into the skeleton code or into their own module. These elements include expenses such as farm insurance, machinery insurance, fuel expenses, electricity expenses, etc.

## 4.5. Use Case Diagrams

Here I have placed a few Unified Modelling Language (UML) diagrams to help understand the design of the project and a high level view of what steps are involved with communicating from one component to the other. I have used Gliffy[[14]](#footnote-14) for the creation of these models. Gliffy is an online UML diagram editor that allows for the user to create free UML diagrams.

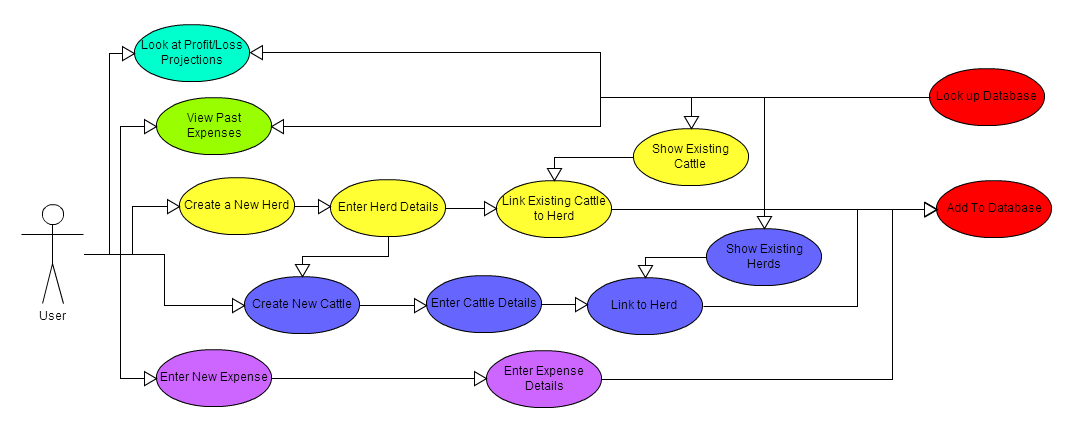


Fig. 9

This diagram is to show some of the functions that the user can use in the finished application. I believe that this diagram covers all of the major functions that the user requires. Look at profit/loss projections and view past expenses only look up the database and do not perform any other function besides this.

The create a new herd function will allow the user to enter herd details and then, either from a list of already listed cattle, can used checkboxes to check those cattle into the new herd or can create new cattle to be placed into the herd. This will use the function create new cattle to enter the cattle details. However, once it asks the user to link the cow to a herd, it will have the new herd checked. It will then add the new herd and the new cattle to the database.

The function to enter a new expense looks a little vague in this diagram. However, this function is to do exactly that, to enter an expense. The user will be given a choice of what type of expense they want to record and will then enter the details for that type of expense depending on what type they have chosen.

## 4.6. Data Dictionary

“A data dictionary, or metadata repository, as defined in the IBM Dictionary of Computing, is a "centralized repository of information about data such as meaning, relationships to other data, origin, usage, and format." The term may have one of several closely related meanings pertaining to databases and database management systems (DBMS):

* a document describing a database or collection of databases
* an integral component of a DBMS that is required to determine its structure
* a piece of middleware that extends or supplants the native data dictionary of a DBMS”[[15]](#footnote-15)

I have created some preliminary data dictionary diagrams for the database. The key information that is needed has been listed. Some will expose themselves during development and more may be added to different tables to make a more usable database structure. These are should not expand any more, however, as the project develops, in may not be new types of data to be stored, but maybe best to split one of the data stores below to make a better record of information.

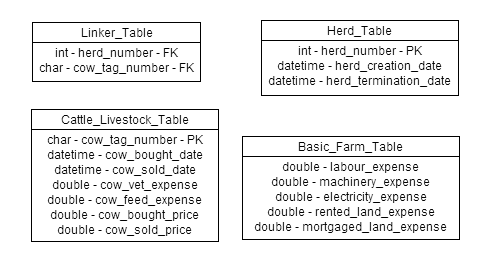


Fig. 10

These images are currently the database tables that will store the information needed to allow the user to keep track of their finances. These cover most of the required information needed, if not all, in the components that I will be developing.

The Linker Table is the table that will keep track of which cow belongs to which herd. This will grow with more modules that are added. It will keep track of not only which cow belongs to what herd, but also what crop belongs to what field in the tillage module, etc. They may even be more added to this table if a better solution is found and more tables need to be added.

The Herd Table keeps a list of all of the herds the farmer has. This one was tricky to design as I thought about putting a field in here to store how much the herd is currently worth and for what the herd sale price was when the herd is terminated. However, I decided not to as cattle may be moved from herd to herd quiet often. I felt that this may cause a lot of confusion for the farmer as they may look at a herd in the morning and show a profit and then move a cow to a different herd in the evening and show a loss. The function to show the herds net worth currently will show all of the cows individual selling price who belong to that herd. On termination of the herd, if cattle still belong to the herd and have not been sold, they will be grouped in a default herd with a default ID such as ‘0’.

The Cattle Livestock Table will store all relevant information of the livestock that the farmer would like to know about a particular animal. This will be the biggest table by far on the database as this will have multiple cattle being saved to the records every time the farmers goes to buy or sell cattle. This will store all of the expenses that an individual cow has used, such as feed expenses and vet expenses. This table will also store the price for which the cow was bought and sold for. The sold price will be null until it has actually been sold.

The Basic Farm Table belongs to the basic farm needs component that will come as default with the skeleton code. These data stores, as can be seen from the names, are to store the basic needs of a farm, electricity, fuel for machinery, expenses for rented or owned but mortgaged land, and of course labour. These where put into their own table so as to not cause duplication if more than one component was to be installed.

# 5. Design

The design of my project will be to make a basic skeleton that will allow constructs to be attached and removed without affecting one another. This will need to be done as to say that I am developing a farm finance manager is too vague and may lead people to believe that all of these elements will be addressed. I plan to only address one point at this time and that is to develop a construct to manage the finances for a beef farmer. These finances will include the purchase of feed, life stock and the selling of life stock.

I will take some features that I like from two of the three products that I have researched as I feel that they will help the overall look and feel for my project. I plan to use something similar to the You Need A Budgets interface as I like how the information is displayed in an easy to use style to show what you want to see when you want to see it. It also allows you to compare between two months at the same time. I will also plan to use the styles of graphs that both You Need A Budget and Quicken Starter Edition use. I believe that these graphs can display all the information that will need to be shown. These two graphs I believe are visually pleasing while at the same time showing relevant information that the user will want to see.

I plan to use colours as indicators in the Graphical User Interface (GUI) such as red numbers to show a loss and green numbers to show a profit. However, not all numbers are going to be coloured. Only if there has been a recent change in projections to change the outcome of the projected profit or loss. The difference will be highlighted in these colours, not the amount. There will also be colour used on the GUI itself to make it look more attractive and easier to stare at for some time without the user just looking at some generic colours.

During the development of this project, I will try to develop as much generic code as possible to allow for some changes that may be needed if this finished product was to be used outside of Ireland. I do not believe much change will be needed, however farms in the likes of America tend to have thousands of cattle to be considered big, where in Ireland, a couple hundred cattle would be big.

It can be either a huge advantage to have all farmers in the world to use some sort of standard practices to farming that will come under the same expenditures or a disadvantage if all countries have a different standard that will cause this application to be unusable outside of Ireland. If farmers use similar practices in farming, and have similar expenses, all that will need to change is the currency and the tax. This can be an added feature to the settings menu to allow for it to be change to a new country.

I have included two diagrams below which I used Gliffy[[16]](#footnote-16) to draw the diagrams. These two diagrams are solely for the purpose to show how I plan to design the code from a high level and basic view. This is just to help the understanding of what I mean but using a skeleton code with multiple modules. They are not actual UML diagrams in the sense of giving relevant information as to how this project will be designed.

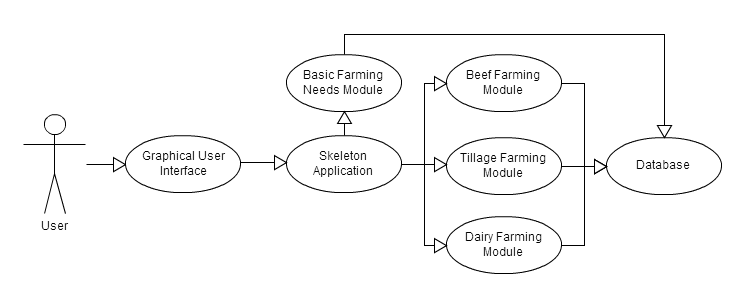


Fig. 11

This diagram shows the basic design of the project. The actor in this diagram is the user. They will use the Graphical User Interface (GUI) to communicate with the software to input their information and to receive the information they want. The GUI will be built into the skeleton of the application; however, I have separated them to show how the skeleton is to work with the installable components. The Beef Farming Module is the only component that I will be developing for this project at this point in time. The Tillage Farming Module and the Dairy Farming Module will be developed in a future project, however I have place them into this diagram to show how the components can be put in and taken out without affecting the running of the application. The Basic Farming Needs Module will be packaged with the Skeleton Application as all farms will need this, regardless of what type of farming is used. These functions have been put into their own module as it will reduce duplication of expense or having to fill in multiple places for exact expense if a farmer partakes in more than one type of farming.

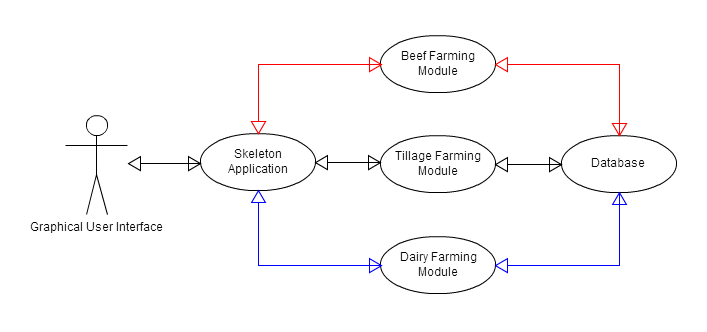


Fig. 12

The diagram Fig. 10 is a more broken down version of the diagram Fig. 9 earlier, to explain how the skeleton and component structure will work. In this case the actor is the Graphical User Interface (GUI) as it will be what starts the process. The rest of the use cases are different from the first diagram as they have arrows moving forward a backward to show the flow of information. The three components do not connect or talk to each other and this is why the components can be put in and taken out without affecting a different component. If the user had all three components installed, and wants to see information from all three components, each component will gather their own information and pass it back up to the skeleton to process the three piles of information and output it so the user can see the results. I also left out the Basic Farming Needs Module as this will be embedded in the Skeleton Application and will not be able to be removed.

# 6. Schedule and Risks

|  |  |  |
| --- | --- | --- |
| Date | Description | Status |
| 07-01-2013 | Design the Graphical User Interface (GUI) and implement it as the base of the skeleton code. | Not Started |
| 14-01-2013 | Design and implement the database tables. | Not Started |
| 21-01-2013 | Design the functions in the beef tracker module. | Not Started |
| 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. | Not Started |
| 18-02-2013 | Test how well the skeleton code works with the database and if the right connections can be made. | Not Started |
| 25-02-2013 | Test how well the skeleton code with the attached module, beef tracker connects to the database. | Not Started |
| 04-03-2013 | Design and build functions to turn the textual data into graphical output. | Not Started |
| 11-03-2013 | Design and build function to show projects profit and loss. | Not Started |
| 25-03-2013 | Test for accurate information. | Not Started |
| 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. | Not Started |
| 08-04-2013 | Fix any bugs found. Test the overall project again. Fix any bugs. | Not Started |
| 19-04-2013 | Final test of the project. If any bugs, highlight in documentation as current errors. Hand up of documentation and project. | Not Started |

This is probably the hardest part of this phase of the project. To try and schedule how to develop and test the project is a big challenge as I cannot say at this point in time how much effort will be needed to be put into developing the project, or how long it will take to test and fix any and or all errors. By scheduling now, I reduce the risk of going over the deadline limit of week twelve; however I am taking the risk of not meeting the deadlines of each week.

From looking at my schedule, I am testing after any major development as I feel that this reduces the risk of having an incomplete project where as if I was to test at the end of the timeline such as starting testing at week eleven. By testing after any major development, I will be able to find problems quicker as I will have a clear idea of the code and how I wanted to implement that particular piece the perform a certain way. This should hopefully allow for me to not run over time as I will be able to fix errors straight away instead of spending time looking for a small error and looking over notes on what I was thinking at that point in time to come up with the logic I used.

There is some risk with creating a small module of elements that do not fit into any particular group or which fit into more than one group causing a duplicate, to be attached to the skeleton code with the initial download. This may cause a change in how to design the Graphical User Interface (GUI), as some elements may need to be used and some may not. For those that are not used, this may cause problems with the profit and loss projections. It may take some planning to develop a way to get around this problem and make the application do a check to see if the unused elements are usable variables in the projections.

This schedule listed here is a little vague as it does not list the finer points of the project like developing menu for the Graphical User Interface (GUI) and also developing the layout for the GUI could take a day for each part to complete. These finer details have be left out of this schedule to allow for some manoeuvrability with time as some parts of the development process may take only a couple of hours to complete while other parts could take up to a couple of days to complete. This is why I have just outlined the bigger, more whole parts like the GUI, even though it has many small parts that will need to be developed.

# 7. Conclusion

My conclusion to this project is that it will be an interesting project to attempt to take on from both a personal and professional outlook. I will need to implement all of the skills that I have developed over the past three years of college and what I am learning at present. There will also be a lot of implementing new skills to achieve some of the project objectives, such as the profit and loss projections. This I feel will be a factor as to how successful this project can be as it may be time consuming trying to get the correct logic to manipulate the required data to give realistic projections. These personal reasons, along with the feeling that I will get to undertake the biggest project to date, will be awarding as I will have designed my own project from the ground up, from start to beginning on my own. From a professional view, it will hopefully be something that I can proudly show potential employers that I have created this project from ground up without fail.

I believe that this project from a sellers stand point will either make big sales or miniscule sales. It is hard to know if this project will actually be useful enough to farmers for them to purchase the final product. I have asked a number of farmers besides relatives to get an unbiased opinion and I have gotten good feedback that it could possibly help keep better track of outgoing and incoming money to help for better investing when the market picks up.

The release of this project to other countries outside of Ireland and the United Kingdom may not take as it may not suit that country’s farming standards. The farmers may need extra options for expenditures or may have to pay more taxes on products and sales as their country may require this. I could not find much information about how other countries control their farming matters, or how they farm or sell their products. I am assuming that the expenses will be the same here as they will be anywhere else in the world.

I will design the Graphical User Interface (GUI) under the assumption that all countries have the same types of expenditure like labour, feed, machinery, etc. However, I will prepare to allow for easy changes if later highlighted that there is differences in another countries needs. This will be done by making as much generic code as possible to allow for quick and easy changes to be made without affecting the rest of the code.

The design of another module is possible as I feel that there are enough elements to warrant their own module as some do not fit into any current module and some fit into multiple modules. Therefore, I am contemplating developing a smaller module that will come with the initial skeleton code. This will keep it from the skeleton code a means easier testing rather than having to edit the skeleton code if I need to add or remove an element. It will also reduce the risk of creating errors by keeping the code small rather than having all of the code in one file.

# 8. References

<http://en.wikipedia.org/wiki/Java_(programming_language)>

<http://en.wikipedia.org/wiki/C%2B%2B>

<http://personal-finance-software-review.toptenreviews.com/>

<http://ru.wikipedia.org/wiki/AceMoney>

<http://en.wikipedia.org/wiki/Quicken>

<http://en.wikipedia.org/wiki/You_Need_a_Budget>

<http://www.mechcad.net/products/acemoney/personal-finance-software-quicken-alternative.shtml>

<http://www.gliffy.com>

<http://en.wikipedia.org/wiki/Functional_requirement>

<http://en.wikipedia.org/wiki/Non-functional_requirement>

<http://en.wikipedia.org/wiki/Data_modeling>

<http://en.wikipedia.org/wiki/Data_dictionary>

1. <http://en.wikipedia.org/wiki/Java_(programming_language)> loosely based on description. [↑](#footnote-ref-1)
2. <http://en.wikipedia.org/wiki/C%2B%2B> loosely based on description. [↑](#footnote-ref-2)
3. <http://personal-finance-software-review.toptenreviews.com/> [↑](#footnote-ref-3)
4. <http://ru.wikipedia.org/wiki/AceMoney> [↑](#footnote-ref-4)
5. <http://en.wikipedia.org/wiki/Quicken> [↑](#footnote-ref-5)
6. <http://en.wikipedia.org/wiki/You_Need_a_Budget> [↑](#footnote-ref-6)
7. <http://www.mechcad.net/products/acemoney/personal-finance-software-quicken-alternative.shtml> [↑](#footnote-ref-7)
8. <http://en.wikipedia.org/wiki/Quicken> [↑](#footnote-ref-8)
9. <http://en.wikipedia.org/wiki/You_Need_a_Budget> [↑](#footnote-ref-9)
10. <http://en.wikipedia.org/wiki/You_Need_a_Budget> [↑](#footnote-ref-10)
11. <http://en.wikipedia.org/wiki/Functional_requirement> [↑](#footnote-ref-11)
12. <http://en.wikipedia.org/wiki/Non-functional_requirement> [↑](#footnote-ref-12)
13. <http://en.wikipedia.org/wiki/Data_modeling> [↑](#footnote-ref-13)
14. <http://www.gliffy.com> [↑](#footnote-ref-14)
15. <http://en.wikipedia.org/wiki/Data_dictionary> [↑](#footnote-ref-15)
16. <http://www.gliffy.com> [↑](#footnote-ref-16)