Project 2015-16

Dissertation

School of Computing & Information Engineering

Name: Pamela McCloskey

Project Title: Against the Grain

Supervisor: Dr Adrian Moore
Second Marker: Janet Allison

Submission Date: 1 September 2016

Ulster University

Plagiarism Statement

I declare that this is all my own work and does not contain unreferenced material copied from any other source. I have read the University's policy on plagiarism and understand the definition of plagiarism. If it is shown that material has been plagiarised, or I have otherwise attempted to obtain an unfair advantage for myself or others, I understand that I may face sanctions in accordance with the policies and procedures of the University. A mark of zero may be awarded and the reason for that mark will be recorded on my file.

Acknowledgements

I would like to thank my supervisor Dr Adrian Moore, for his guidance, assistance and feedback given throughout the project. Also thanks to Janet Allison for her feedback and support.

Table of Contents

Table of Figures

Abstract

This project seeks to solve the problem of creating a single, centralised data storage repository for commodity trade transactions. The solution provided should be accessible by multiple traders simultaneously with a unique user login. Data should be collected from and presented back to traders, for their required commodity only, in a uniform format, to ensure data consistency. The project seeks to add value to the data presently gathered, provide mobile access, enhance collaboration, and improve data integrity.

The proposed solution is a database driven, web application that can be accessed via a browser. The basic client/server model can allow traders to interact with the system via a simple GUI, with PHP used as the logic between the client and the backend MySQL database, where the data is stored. SQL queries can add and retrieve data for the application and can be further used to add value to that data. The solution seeks to enhance and modernise the traders current experience of data management and as such, attempts to utilise data to glean information for accounting, risk management and business generation purposes.

An analysis of the problem was undertaken with an investigation of the status quo and possible alternatives. Stakeholders were identified and their requirements gathered to inform the design of the new system. The architecture of the system was investigated and possible tools and technologies evaluated. An Agile methodology was employed for development work, with an initial prototype developed. Feedback from the prototype informed the subsequent development iterations and ad hoc testing of each functionality along the project path highlighted where extra functionality was needed.

Key words: **Centralised data repository, database driven, web application, commodity trades, browser, mobile, multiple users, collaboration, data consistency, data integrity, data added value, Excel.**

Chapter 1: Introduction

1.1, Introduction

In order to stay competitive today, companies need to keep up to date with the information revolution and the latest technologies. A quick google search of the term 'information revolution' yields the following definition. (Google, n.d.)

"noun; The proliferation of the availability of information and the accompanying changes in its storage and dissemination owing to the use of computers". It follows then that information itself or data, is a key asset of any company, how it is stored is of critical importance and how it is used or disseminated determines the value that can be added by that data.

In addition, users themselves also increasingly need to be mobile and workplaces need to be accessed from any location, virtual even. Instant and organised access to relevant information, by the relevant users, 24 hours a day from any location is becoming the status quo. Safely and securely storing that up to date information for users to access is only part of the story though. Data is increasingly being analysed to derive added value and with the plethora of tools available for data management today, companies need to arm themselves with faster, more accurate and informative ways to store and use their data.

1.2, The Problem

The problem this project investigates is stated as "To assist a grain trading company which imports grains to be sold to the agricultural sector, it would be highly beneficial to record data on transactions in a single centralised data repository, and access this data on the move. The aim of the project is to develop a web based, database driven application for this purpose. The project additionally seeks ways to derive added value from such data, for the purposes of accounting, risk management and business generation".

The grain trading company has four traders and a manager who all currently use their own personalised Excel spreadsheets to store transaction data. There is no uniformity of approach or consistency in data collection. There is no single data repository and each user can only access information from their office hard drive. This ties traders to their desk and discourages working out of the office. Excel in itself can be error prone, and users must be experts to fully avail of the full power of the tool. There are problems with version control, multiple users accessing the same data

and limited opportunity for scaling up as an enterprise grows. Excel is being incorrectly used as a database.

1.3, A modern solution

A modern solution to this problem is proposed, one that avails of today's tools and technologies, not least the internet and ubiquitous client/server solutions in daily use. A system built around trader interaction that is simple to use, fulfils basic requirements, is scalable, mobile and can have features added in the future is desirable. The solution should ideally grow and evolve with the company and provide more than just a data repository, but also value added analysis of the data. Key objectives of data analysis are; accounting, risk management and business generation purposes. A web based, database driven application is thus proposed as the solution.

An analysis of the problem was completed, with stakeholders identified and their requirements used as a template for the design of the system. Possible tools and technologies were researched, with development work then carried out in iterative sprints, in line with an Agile methodology. A simple GUI was created using HTML5 and CSS, with PHP used as the logic layer to communicate with the backend database of the system.

1.4, Aim & Objectives

1.4.1, Aim

The aim of the project is to create a single, centralised data repository for commodity trade transactions. The system should be accessible via a browser and not only store and retrieve data, but add value to that data for accounting, risk management and business generation purposes.

1.4.2, Objectives

- Complete an analysis of the problem and discover what can be achieved by finding an alternative to Excel.
- Gather requirements of the traders to inform the design and functionality of the system.
- Design a simple GUI for traders to interact with the system on a browser, to record and retrieve transactions, as well as extra functionality for data analysis.
- Create test cases for functional, regression and performance testing of the application.

1.5 Requirements for Development

- A lap top or PC
- Notepad text editor.
- XAMPP which contains Apache Server, MySQL database, support for PHP.
- Programming Skills – HTML5, CSS, PHP SQL, Javascript.
- Fusion Charts.

1.6 Dissertation Outline

Chapter 2 - The Analysis Chapter explores the problem with the status quo position of using Excel as a database and discusses what can be achieved by finding an alternative.

Chapter 3 – The Requirements Analysis Chapter identifies the end-user requirements for the system.

Chapter 4 – The Design Chapter is an overview of the design process, tools and architecture.

Chapter 5 - The Implementation Chapter details the implementation of the solution, including the Iterative Sprints taken to complete development work.

Chapter 6 - The Testing and Evaluation Chapter lists the Test cases used in testing of the final system and discusses the extent to which incorporated features are working as expected.

Chapter 7 – The Conclusions Chapter provides a look back at the entire process and solution achieved. It discusses the extent to which the problem stated at the outset has been resolved and the aim and objectives met. The chapter concludes with suggestions for future enhancements to the project.

Chapter 2: Analysis

2.1, Setting the Scene

A grain trading company carries out multiple transactions daily in the physical (or cash) commodity markets. A team of four traders buy four grain/oilseed products (corn, wheat, barley, soybean meal) from suppliers globally. One trader is assigned solely to one product, with that trader being responsible for the running of that products trading book. The individual profit and loss or P&L of each product is synonymous with the P&L of the individual trader who trades it. Each product can be supplied by one or more suppliers, with the trader undertaking a PURCHASE contract with the supplier to purchase their product. Container ships of the dry goods are then shipped from the supplier, with actual physical delivery taken by the company, who has to store the products in silos at their port location. The trader then sells their individual product on to multiple customers in the agri-sector, for example, feed manufacturers, as an ingredient in animal feed. This transaction from the trader to a customer takes place by means of a SALE contract and the customer collects their product from the port. It is usual for customers to buy their product 'forward'. For example, the feed manufacturer will want to make sure they have enough soybean meal for the next six months. They can order soybean meal to be collected by themselves at the port, but specify that they want to collect 10 tonnes each month from January, up to and including June. The trader must make sure there is enough stock of their product available over the time horizon that customers will need it. The trader must ensure that the timing between purchasing their product from a supplier and then selling their product on to customers is aligned. If stock is left in silos too long before the customer needs it, the product will perish. Similarly, if not enough stock is available when a customer requests it, the customer will go elsewhere. The trader must know in advance on a rolling monthly basis whether they are long (i.e. have too much product for their needs) or are short (i.e. not enough product for their needs). Although one or other of these positions may be advantageous in either a rising or falling market, the desire is generally to be neutral, especially in the closer months. The aim is essentially to provide a service to customers and sell them the product they need, at a profit, not to speculate on whether the market will rise or fall.

2.1.1, Transactions take place

Trading takes places on the telephone, with *purchases* made via brokers with *suppliers* and *sales* made directly to the *customer*. Each trader records their transactions in an Excel spreadsheet as they are carried out. Each trader keeps their own personal spreadsheet and formats this according to personal preference. The result is four different spreadsheets, one for each product, with different collections

of data for each product. One person then has to gather these into a master spreadsheet, for management to use, to get an overall picture of the combined position. This process takes place weekly. The data that needs to be captured is outlined below.

2.1.2, Grains are purchased to be re-sold

Purchases are those transactions the trader makes with the supplier. Data recorded is; trader name and corresponding product name, supplier name, purchase quantity (tonnes), purchase price, purchase date, delivery month(s).

Sales are those transactions made by the trader to one of their customers. Data recorded is; trader name and corresponding product name, customer name, sale quantity (tonnes), sale price, sale date, collection month(s).

2.2, The problem with the status quo as experienced.

- Traders are not Excel programmers, nor do they wish to be. Excel is undoubtedly a powerful tool and has many great functionalities. However, the traders in this scenario do not utilise Excel to its full capacity as they are not expert users. Manually maintaining data and formulas and correcting errors when things go wrong results in a cumbersome process.

- Spreadsheet maintenance is time consuming and error prone, especially when copying and pasting takes place or multiple users access and update files. Information sharing is not efficient in the scenario outlined, as each product is looked after by a different trader and they each record data differently. One master spreadsheet to include all products, is adding further time, more scope for error and is still not a real time snapshot of the current position when only completed on a weekly basis. To function as one team, data use needs to be consistent.

- Mobility is an issue as data cannot be accessed outside the office, unless the user copies this to a memory device or emails company data elsewhere, both prohibited practices in this instance. This restricts trader mobility in a world where workforces are growing more distributed and global, increases down time when traders need to visit suppliers or clients and removes the chance for home working, whether that be to simply catch up, or still do minimal work during sick or maternity leave. With today's growing trend towards a work/life balance and greater worker mobility, the current system is out of date.

- Physically retaining information on the hard drive of office pcs means losing out on the ability to store and access data on the cloud or elsewhere on a server.

- Scalability is not facilitated well as transactions increase and user numbers increase. Add the problems of version control and soon enough, 'too many cooks spoil the spreadsheet'. (Whitehouse, 2014)

- Wasted opportunities for data analysis. Big data is getting bigger, both as a concept and a way of improving businesses. The full potential of the data the company holds is not being realised. This data could be collected, stored and used in a more efficient way.

2.3, So how can value be added to data for commodity traders?

Accounting

There are regulatory requirements which need to be adhered to and a fast and accurate method is required to keep proper accounting records. Budgeting and forecasting is also an integral part of any business and keeping track of this via several spreadsheets is not the most efficient way.

Risk Management

One of the biggest problems faced by traders is risk management. As highlighted earlier, traders must make sure they have purchases and sales aligned because actual, physical delivery of thousands of tonnes of a product will take place. That product is perishable and the consequences of a lapse in quality control before the customer collects it are serious. This product will be used in animal feed by the agricultural sector and will directly impact the food chain in Northern Ireland. (NIGTA, 2016) Up to the minute accurate statistics regarding stock levels and customer balances are being sacrificed. This is counter intuitive to good risk management – if knowledge is power, then surely using this knowledge in a knowledgeable way (instead of just storing it!) is even more powerful.

Business Generation

This is also a missed opportunity for potential business generation. Could patterns of buying and drawing down of the physical commodity from each customer be identified in an attempt to pre-empt customer needs and provide a better service, as well as proactively take market share before competitors? This type of information is hard to glean from looking at spreadsheets alone. There may be many other connections to be made within the data and a pair of tired human eyes at the end of a long day, week or month, is sure to miss something, if they even have the will to look at the data this way in the first place! This project aims to investigate a more appropriate method of storing, retrieving and ultimately using data.

2.4, Excel - the world's most used database.

This problem is replicated across the globe in multiple industry types and is not just limited to the example given above. In his article "Excel is the world's most used database" (Baptiste, 2010) Jason L Baptiste argues that users attempt to use Excel as a database but that is not really its purpose. He states; "An alarmingly large number of individuals use Microsoft Excel to store non-numeric arrays of information that should probably be stored in a database / be created by a simple web application". It is argued that companies have been using Excel for years, it is familiar and it works easiest in the short run and users do not want to learn something foreign and new. In the long run though, it is not efficient. Scalability is a problem when the number of users increases and the issue of version control emerges. Additionally, if a company stores all their data in one or several worksheets, it is too easy to move the data to a new location and files could be easily dropped to the public. Examples of Excel being used as a database were experienced by Baptiste in Public Relations, where a company tracked all conversations in a spreadsheet, in Human Resources, a company tracked student absences in a spreadsheet they had used for 15 years. The feeling was that approaching IT about a solution was a 'headache'.

"I realized that there is an intense amount of opportunity around displacing Excel as a lightweight spreadsheet and process management tool. If large corporations are leaving risk management to an excel database, this has to be a very large problem".

Ventana Research states that nine out of ten people use spreadsheets, all or most of the time as they are the default tool for business users and have been since the 1980s. They state that spreadsheets are being 'spread too thin, they're being used for everything from analysis to reporting to data storage." (Ventana, 2015)

2.5, Excel is error prone…from a great height

When the Basel Committee on Banking Supervision (BCBS) and the Financial Services Authority in the UK refer to the need for effective controls by banks and insurers to manage risks when using manual processes or data flow systems such as spreadsheets, it is apparent how dangerous human errors can be when using Excel. This is the first time that spreadsheet management has ever been specifically referred to at such a high level. (Worstall, 2013) At JP Morgan, the Chief Investment Office needed a new value-at-risk model. This was created by a quant expert, mathematician and model developer using a series of Excel spreadsheets, where data had to be manually copied and pasted from one spreadsheet to another. The banks internal Model Review Group identified this as a problem but approved the model anyway. The spreadsheet used a sum function in one part of a formula instead

of average, which resulted in a loss of US$2 billion. (Wikipedia, 2012) A data management company ClusterSeven has stated "The failure of businesses to fully understand, control and monitor data held in spreadsheets leaves businesses worryingly exposed to unacceptable risks and recent indicators from the regulators now suggest that firm action will be taken against those that bury their head in the sand. The simple fact is that spreadsheets continue to be used extensively by all financial institutions and remain at the heart of many firms' data systems." (BobsGuide, 2012)

2.6, Stakeholders

The stakeholders related to this problem are initially the traders and then management, who need to keep a watch on individual trader behaviour, as well as a combined view of the positions. Management need to use this information for budgeting, forecasting and reporting requirements. Suppliers and customers are also stakeholders, in that they will be affected by any changes that are implemented. Customers have the potential for most benefit, as the service they receive may improve considerably as traders have more time to visit them. Additionally, customers own capacity for risk management may improve if traders can take a more proactive approach, due to time saving, better collaboration and a way of identifying patterns in customer behaviour.

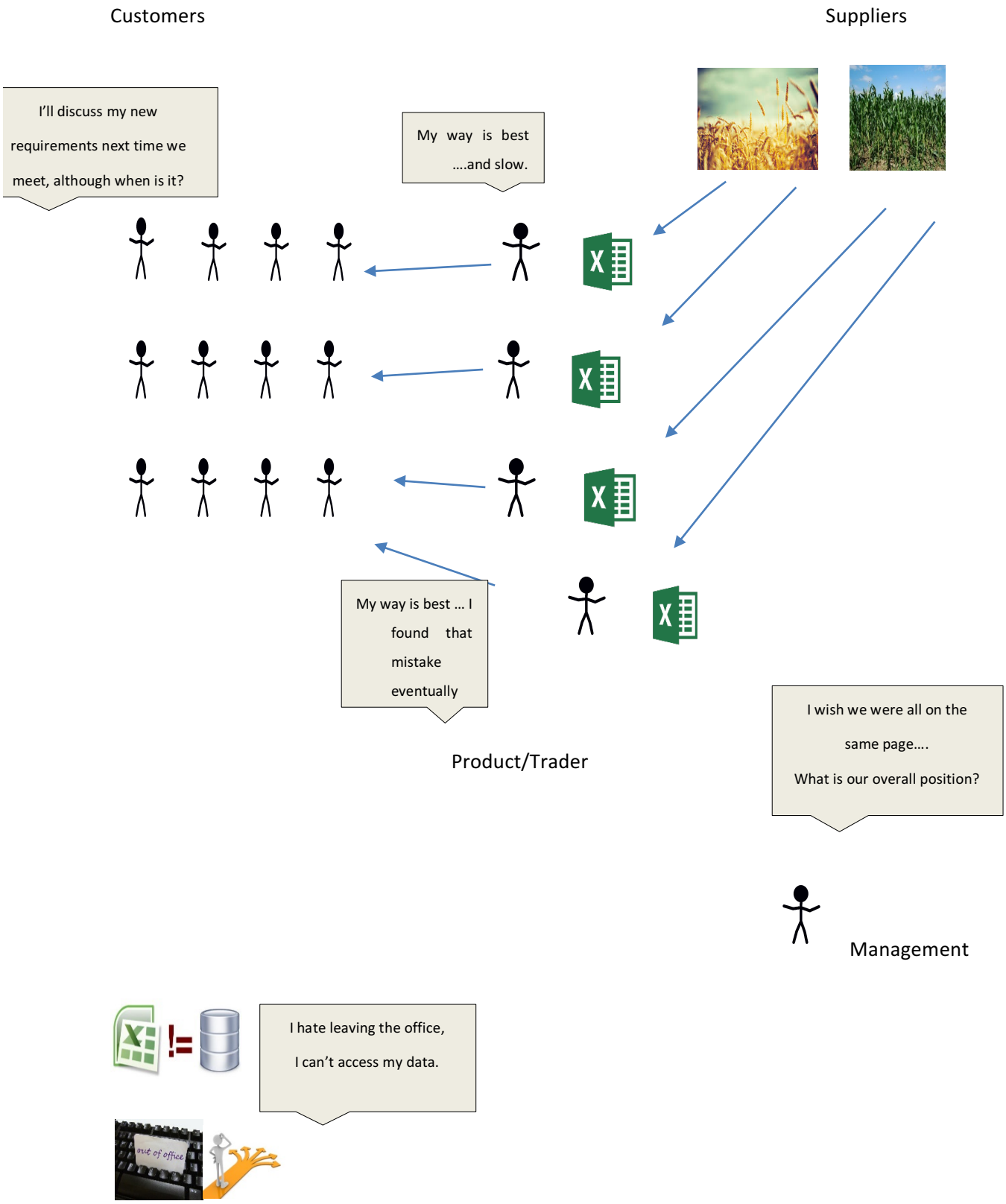


Figure 1 Rich picture showing the problem.

2.7, The Solution

What can be achieved by replacing Excel?

An example where a company has replaced Excel with a software solution is that of Thule Group's North American division. (Gittlen, 2011) The company had been a "classic spreadsheet-driven company". The Vice President of Finance states "We had one person in accounting who kept all the data as up to date as possible. However, it wasn't unusual for it to take a week to consolidate cross-company information, and it wasn't unusual to discover that what managers were looking at wasn't the final spreadsheet". The company abandoned Excel in 2008 and employed a software-as- a-service product from Host Analytics. This benefits were; real time collaboration, mobile access, the facility to create, edit and view real time business data and generate reports with a browser. Each user can carry out different functions depending on their level of authorisation. In addition to this; timeliness, accuracy and accountability has improved as each person is now responsible for their own data. "Ninety-five percent of our spreadsheets are gone, and we've completely eliminated the delay in receiving critical reports. Our business leaders also know that the data they have in hand is accurate and dependable."

Similarly, an US insurance company has seen benefits to replacing Excel for certain functions. (Gittlen, 2011) Each month one employee had to manually compile custom data and copy it into a 13-page Excel template. Once the customer numbers needing the report reached 50, the process took almost the entire month and required one person to work on that task alone. They now use a secure web page with a drop down menu that lets a client run a customised report on demand, not just monthly and incorporates real time data. This has freed up that one employee to carry out other duties.

2.7.1, What then instead of Excel?

The two examples above confirm the developers own experience of using Excel as a business tool for all occasions and reiterate the time consuming nature of carrying out repetitive manual recording of data. As well as being time consuming, the data provided to management once a week in the grain trader scenario outlined, is instantly out of date. There is no instant capacity to view real time data in an organised easy to visualise way.

An alternative solution is needed that saves time, reduces the risk of errors, has the ability to scale, allows for a real time snapshot of transactions, allows for sharing, mobility, different levels of authorisation, accountability and finally allows for some type of data analysis. At its most basic, the

solution needs to provide a single, centralised, storage facility that can be accessed by multiple users, any place, anytime.

1. Single, centralised, data storage facility – Database

2. Accessed by multiple users, any place, anytime – Web based

3. If these two aspects can be fulfilled, it then opens up the opportunity for a third element of improvement, which is data analysis.

2.8, The storage system

How then to store the data? A useful option is to use a relational database system. (Wikipedia, 2016) Data is recorded logically in several tables or relations of columns and rows. Each table represents one object or entity that data is needed for, for example a product. The rows represent each instance of the object, such wheat or corn and the columns represent the attributes that the entity/object possesses, such as price. The relationships between the data are very important. Each row has a unique identifier called a primary key, which allows for one row to be able to be selected and modified individually. It also allows for rows in a table to be linked to rows in other tables by adding a column for the unique/primary key of the linked row. This primary key will then be called a foreign key, and allows for relationships between tables to be established. Data is stored on a server, so there is essentially no limit to the amount of data stored and it means that new information can always be added, so that the data being accessed is always up to date. The data can then be queried to ask questions and glean certain information. (Hernandez, 2013)

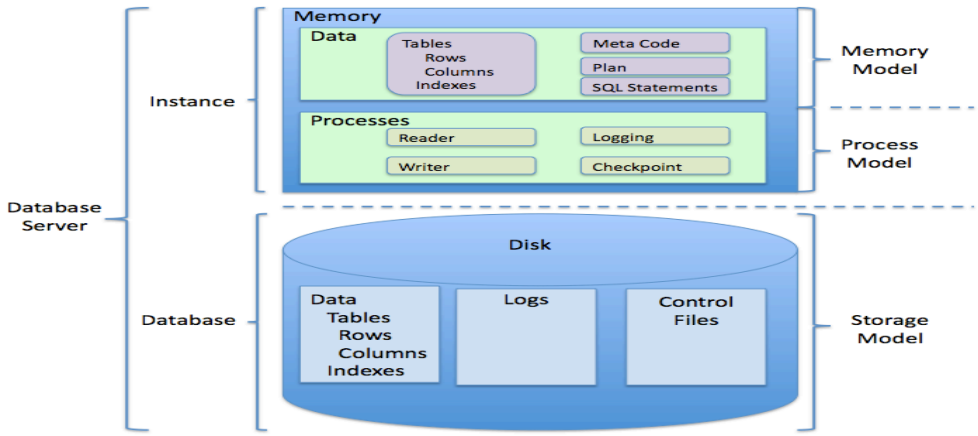


Figure 2 (Wikipedia, 2010) Showing database structure

2.9, Ensuring the storage system is accessible anywhere, any time.

The data that is stored in the relational database system described above needs to be accessed by multiple users, anywhere, anytime, which means making it a web based application. The web is a great example of such databases at work. When a user logs in with a username and password online, it is likely that their web browser (for example Firefox) is accessing a database server behind the scenes, to retrieve information to send back to the user on the web page they are viewing. Everyday applications like Facebook and Hotmail are set up in this way.

2.10, Basic Toolkit options for the project:

- Web browser - to provide GUI (web pages) for user and a way of sending and receiving data (such as Firefox, Safari, Explorer)
- A local web server to send user requests - for example Zend Server or Apache.
- Web page languages – To build web pages (the required languages are; HTML5, CSS and JavaScript), with SQL to query data captured. (Matthews, 2015)
- Server side language - for example, .Net languages, Perl, Python, Java or PHP.
- Database server - where data is stored in a relational database.
- Text Editor or IDE – for example Zend, Microsoft Studio Code, Dreamweaver, Notepad to write code.
- API for charting capability – such as Google charts or Fusion Charts to display trade data in a meaningful way.

2.11, A closer look at database options.

According to the Gartner research company, the leading commercial relational database vendors are Oracle, IBM, Microsoft, SAP and Teradata, with Oracle taking about half the market share. The leading open source (widely available and free) implementations are MySQL, PostgreSQL and SQLite. (Wikipedia, 2015)

Microsoft Access - This runs with SQL but is desktop based. A solution that allows users access on the move and from different devices is preferred. In addition, information can be stored on a server instead of user machines.

SQLite - SQLite manager can be downloaded as an add-on to Firefox. Everything is accessed on the client side (browser) and data is stored on the host machine, not a server. Skype uses SQLite and Adobe Acrobat Reader. (Wikipedia, 2016)

MySQL - This can be used with a web based interface – phpMyAdmin (Delisle, 2012) to create a database – with information being stored on a free server such as Apache (using XAMPP). This option is cross platform and cross browser and offers greatest speed and flexibility. Performance is a priority with web based applications and MySQL is designed to work well with web based servers. (Veltman, 2013)

PostgreSQL is an Object Relational Database, with features similar to SQL but the user can extend the implementation by adding new index methods, procedural languages and aggregate functions. It is considered to be more feature rich but also slower. It is not as widely used as MySQL and has less community support as a result. (Olivia, 2011)

2.12, Putting it all together

From all the database options listed above, MySQL is the most appropriate for this project, with SQL queries to retrieve data. (Stephens, 2015) It is free, has an easy to use interface with PHPmyAdmin, (Delisle, 2012) allows data to be stored in a relational database and data to be stored on a server instead of on the user's machine. The advantages of speed and flexibility as outlined above, are also appealing and it lends itself to the web based goal of the data storage facility. The web pages require HTML5, CSS and Javascript as defined above, and a server side language is also needed. PHP proved to the preferred server side language to use for this project. MySQL and PHP are a common pair and have several advantages; (Suehring & Valade, 2013). Three IDE options were considered. Initially Zend appears as a good match for MySQL and has good support but is not free and so has been discounted as an option for this project. Dreamweaver was considered but there was a learning curve which slowed the project down, as was the case with Visual Studio code. As a result, no IDE was used and Notepad proved a more than suitable text editor for code. The local web server chosen was Apache, which is the obvious choice, as it is the most commonly used on the internet. It is free, secure, runs on most operating systems) (Suehring & Valade, 2013)

- Free
- Web oriented
- Easy to use

- Fast
- Communicate well with one another
- Wide base of support for both
- Customisable

XAMPP provides MySQL, phpMyAdmin, Apache and support for PHP and as a result, this package was used for development, alongside Notepad as a text editor for code.

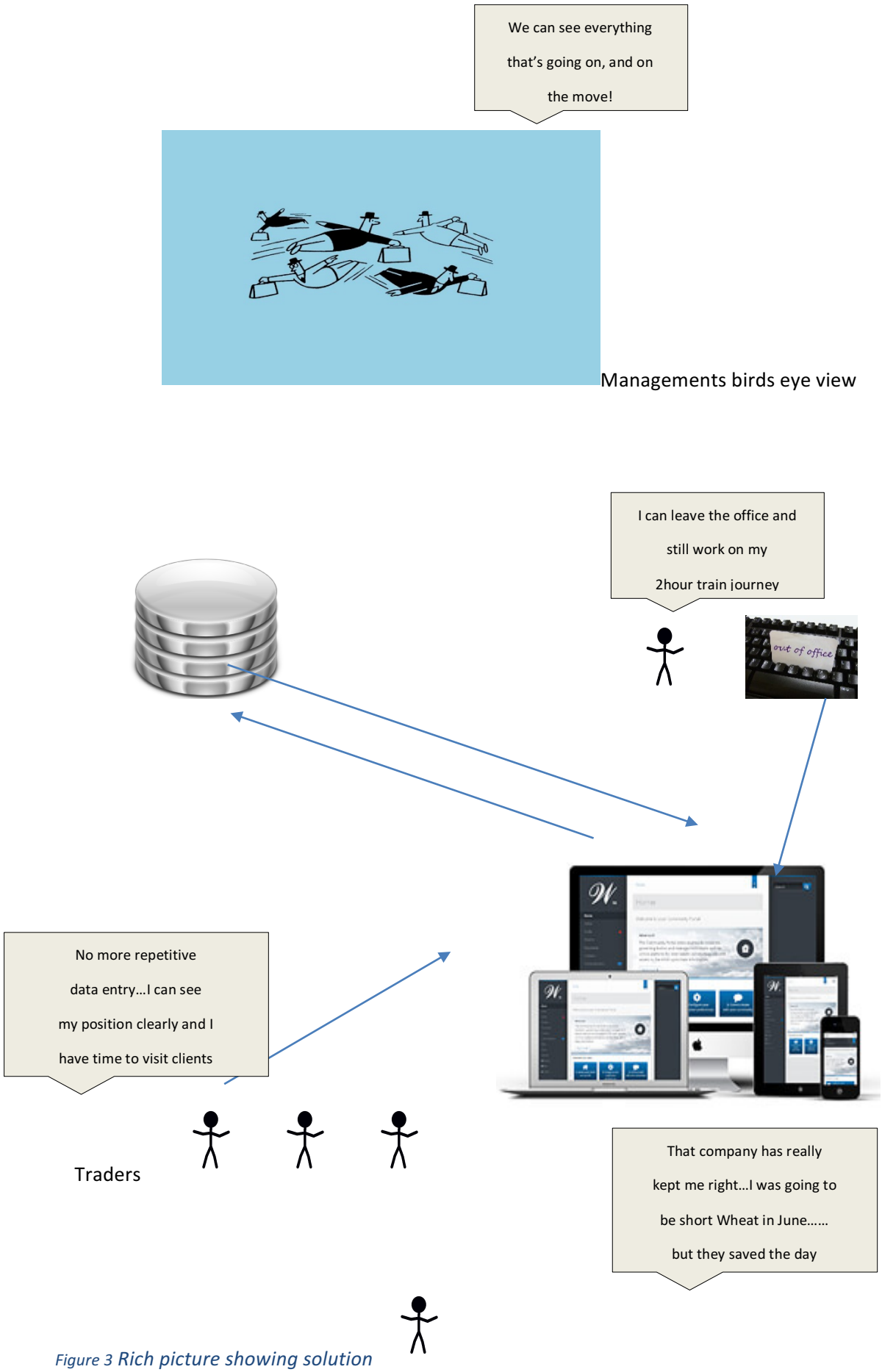


Figure 3 Rich picture showing solution

Chapter 3: Requirements Analysis

3.1, System Requirements

A single centralised storage system that is web based is beneficial as data can be captured and stored once. It will be immediately accessible to all traders and management without the need for copying and pasting or issues about version control. Maintaining Excel formulas will no longer be necessary and users can have confidence in data integrity. Traders do not need to up-skill technically with Excel training courses, saving time and money. If data can be captured as it happens on a database that can then be viewed from a browser, an up to date, real time profile can be seen at all times. This will greatly assist management, who are required to travel a lot. It will also free up traders time, as the repetitive weekly process of updating spreadsheets will not need to be carried out. Traders will also be able to work remotely and have more time for client meetings, which are conducive to building and maintaining customer relationships. Happier customers and a better appreciation of what they need can only be profitable in the long run. In addition, information sharing will be seamless and errors will be less likely. It will also be vitally important for management to be able to see at a glance, each trader's exposure and the overall position. Potential benefits are as follows;

- A single centralised data storage and retrieval system.
- Scalability – user numbers can increase easily.
- Multiple users can access simultaneously.
- Collaboration is facilitated as users have a uniform view.
- Improved data integrity – No risk of formula mistakes.
- Up to date information at the user's fingertips.
- User friendly GUI to edit and use system.
- Efficient and automatic information sharing with management.
- Time saving – No copying, pasting and checking required.
- Data can be accessed on the move via a browser.
- More time to spend on client visiting, improving relationships.

- Improved risk management by keeping track of stock in a user friendly format.
- Greater transparency over traders' positions.
- An overall bird's eye view of combined positions for management.
- Easier accounting, budgeting and forecasting.
- Improved capacity to manage legal requirements in relation to accounting and reporting.
- Data analysis capabilities and the potential for a Business Intelligence Dashboard.
- Opportunities for business generation via observation of customer patterns.
- Easy install and update as updates are server side and not on each users device.
- The potential for integrating with other web based services or system
- A system that can grow and change as the business does and meet future needs.

3.1.1, System Users

Users have been identified as individual members of the trading team and management. Their needs have been identified from personal experience in the industry and conversations with trader colleagues and management. The other examples given in the solution section describing how other companies have moved on from Excel have also informed the user needs and requirements. The specific functional requirements of the users have been condensed into User Stories below, to inform the design of the system.

As a manager I want each trader to access their own product area only via a secure login.

As a manager I want a real time birds eye view of all products combined.

As a trader I want a secure login.

As a trader a want to enter data as a purchase or a sale.

As a trader purchase data on: Trader, product, supplier, tonnage, price, delivery month.

As a trader sales data on: Trader, product, customer, tonnage, price, collection month.

As a trader I would like to access this system outside of the office.

As a trader I would like to be able to use this system at the same time as colleagues.

As a trader I would like to be able to view real time data on a product basis.

As a trader I would like to be able to view real time data on an order basis.

As a trader I would like to be able to view real time data on a customer basis.

As a developer I want the system to fulfil all user requirements.

As a developer I want the system to allow for further users to be added as needed.

As a developer I want the system to modular & flexible so further features can be added.

Figure 4 User Stories to express functional requirements of the system

3.1.2, Functional Requirements

1. The system should allow for each user to login uniquely and securely.
2. Each trader should have access to their own product area only.
3. Management should be able to view any product and also a combined view of all products.
4. A function to enter data as either a purchase or a sale is required, with several key data points captured.
5. A function to view data on a product basis.
6. A function to view data on an order basis for each product.
7. A function to view data on a customer basis.
8. A function for real time data analysis on product and customer views, to look for patterns, alert to short or long positions.

3.1.3, Non-Functional Requirements

Usability & Accessibility– The system should be easy to navigate and use.

Efficiency – Data must be able to be entered quickly and stored quickly with good system performance and data integrity.

Portability – Users need to be able to access the system via a browser when outside the office and potentially on a wide range of devices.

Reliability – The system should run in the required manner with error handling as appropriate.

Security - Company information and user data must be kept secure and backed up regularly.

Ethical – Customer data must only be used in accordance with Data Protection Laws and not held for longer than necessary.

Chapter 4: Design

4.1, User Interface

The user interface is designed to be simple and intuitive with self-explanatory navigation.

Company Home page – Provides a Login button.

Figure 5 The Home Page

Login Page – Provides fields for email and password to be entered, with a unique secure login for each user.

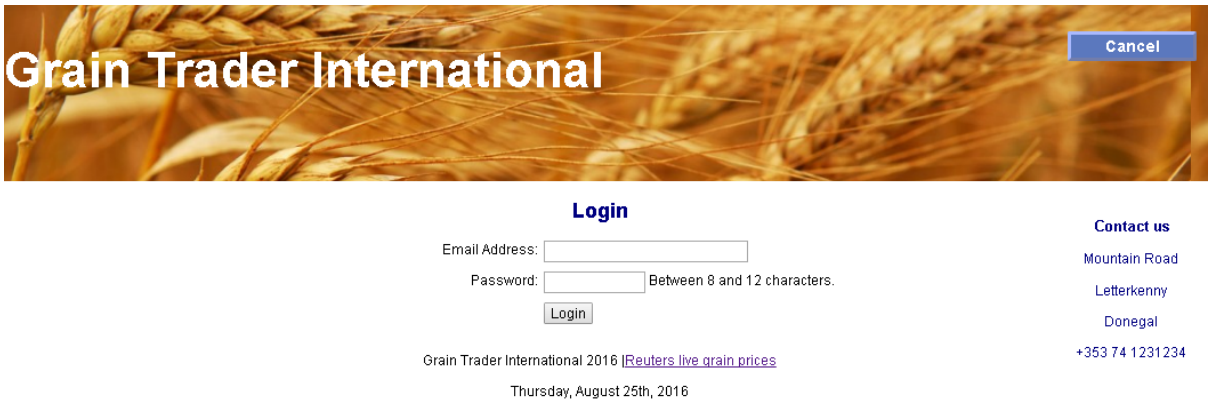


Figure 6 The Login Page

Dashboard Page – Trader user is taken to their own Dashboard, which is Corn, Wheat, Barley or Soybean meal. Manager user is taken to the Management Dashboard. There is an instant visual snapshot of the position for that trader without any further clicks being needed. The three main objectives beyond data storage and retrieval are: *Accounting, Risk Management* and *Business Generation.* The Dashboard seeks to give instant, live information to assist these objectives.

4.1.1, Accounting - Live P&L snapshot for the current year.

1. Is the trader in profit or making a loss at this point in time? Traders refer to this position as their 'P&L', short for 'Profit and Loss', which is referring to their cash position. It is simply the difference between money in from sales and cash out from purchases.

   P&L = Total Sales – Total purchases.

The figure displays in red if the trader is currently at a loss and displays in green if the trader is in profit. This web application is not for the wider public, but in the event that a colour blind user did use the application, the positive and negative values of the numbers will still be self-explanatory.

2. Is the trader long or short of stock at this moment? This simply refers to how much stock is on the ground right now in silos. It is the difference between what has been delivered to date and what has been collected to date. This figure ignores stock that is being delivered or collected at future dates.

   Stock position/Stock in silos = Deliveries -Collections.

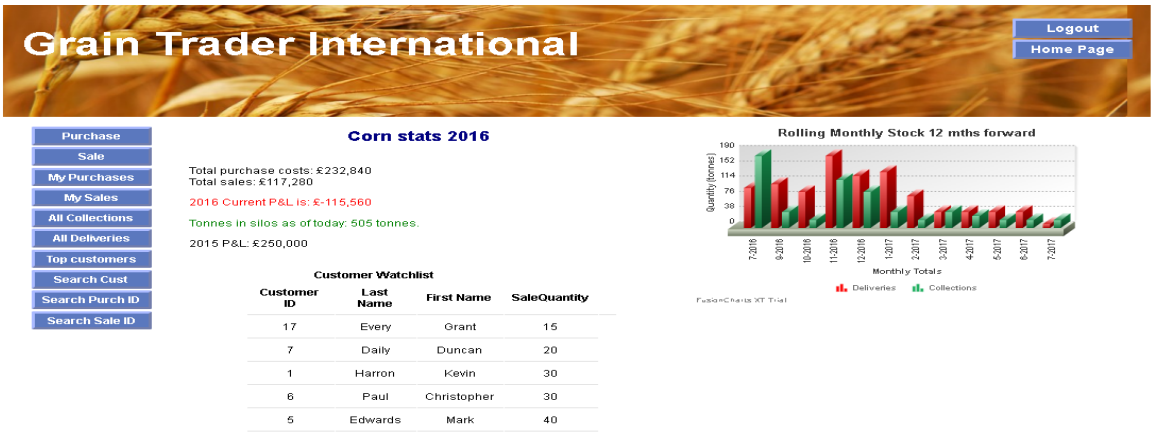


Figure 7 The corn 'Dashboard' page for the corn trader only.

4.1.2, Risk Management - How is P&L managed against inventory?

The trader now knows their current stock position, but also need to know their stock position for the forward months, to align purchases and sales correctly. Purchases result in multiple future deliveries and sales result in multiple future collections. For this reason, a chart of 'Rolling Monthly Stock 12 mths forward' is displayed. The red bars represent the sum of Deliveries due for each of the next 12 months going forward. The green bars represent the sum of Collections due for each of the next 12 months going forward. At a glance, traders can identify their position, long or short, for each delivery month. They must match the timing between purchases and sales, as the product is perishable and they must ensure they always have enough stock for future obligations.

4.1.3, Business Generation – Customers who may need a call.

The 'Customer Watchlist' displays the five customers with the lowest sales volumes for the current year. This could identify customers where potential sales could be made, or where market share could be diverted away from competitors. If traders have more time to visit customers, there could be an opportunity to discover why customers are potentially using other suppliers too.

4.2, Trader interaction with the system

It is only when the trader wishes to interact with the application that they need to perform another click. Interaction is then focused on the particular requirement of the trader at that moment and can be fulfilled by a click on the relevant clearly labelled button. After the trader has clicked on any one of these buttons, a 'My Dashboard' button is displayed on the top right of the header menu for the trader to go back to the Dashboard page.

Figure 8 The trader side menu bar for the system functionality

4.2.1, Purchase

The purchase function allows the user to enter a purchase of stock from their supplier, followed by delivery details. After clicking on the Purchase button, the particular trader's product is already displayed. They need only select the Supplier from the drop down menu, instead of typing it in and can then enter the price and quantity needed before clicking Submit. This minimal interaction has stored their transaction details securely and quickly.

Figure 9 The Corn Purchase Form

Once purchase is clicked, the trader is then taken to the Delivery page. The auto generated Purchase ID is displayed as well as the number of tonnes that need to be entered for delivery. This is giving the user all the information they need.

Figure 10 The Delivery Input form

4.2.2, Sale

The sale function allows the user to enter a sale of stock to their customer, followed by collection details, as outlined below.

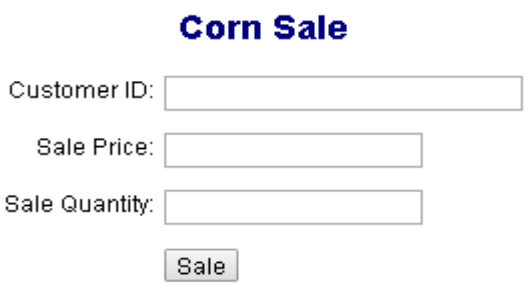


Figure 11 The Corn Sale Form

Sale Number 1 is complete.

5 tonnes left to input for collection.

Collection Quantity:

Collection Month - Select -

Collection Year - Select -

Add collection

Figure 12 The Collection Input Form

4.2.3, My Purchases and My Sales Buttons

The trader can then view the Purchases and Sales they have made for the *current year* through one click on the 'My Purchases' or 'My Sales' buttons, with a total figure shown at the bottom.

Corn purchases 2016

Supplier Name	Supplier ID	Purchase ID	Purchase Order Date	Purchase Price (Stg.)	Purchase Quantity (Tonnes)	Purchase Total(Stg.)
Cargill	2	58	2016-08-01 00:00:00	100	50	5000
Bunge	1	99	2016-08-14 14:40:07	1	1	1
Bunge	1	100	2016-08-14 16:30:20	10	10	100
Bunge	1	101	2016-08-14 16:30:38	78	10	780
Bunge	1	102	2016-08-14 16:30:54	10	10	100

Total purchases: £5,981

Figure 13 The Corn purchase transactions

4.2.4, All Collections, All Deliveries

The trader can click on 'All collections' or 'All Deliveries' to get details of *all these transactions to date*, not just the current year. However, there is a bar chart to the right of the page that shows the Deliveries and Collections for the *current year only*, not rolling forward 12 months like the Dashboard page.

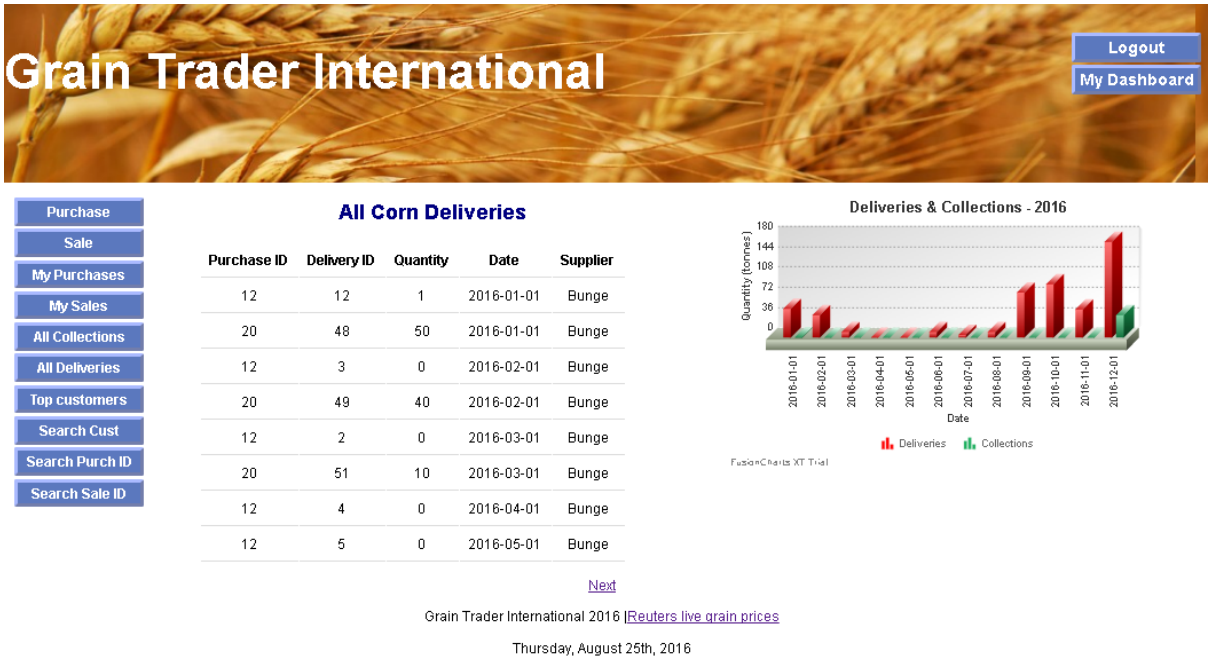


Figure 14 Deliveries and Collections Chart for the current year

4.2.5, Top Customers

A click on the 'Top Customers' Button, takes the trader to a list of their top customers by descending order of sales volume for the current year. These will be the most important customers and they receive more preferable prices, terms and conditions.

4.2.6, Search functions

The trader can then search the database for a customer by name, or for a transaction by Purchase ID or Sale ID.

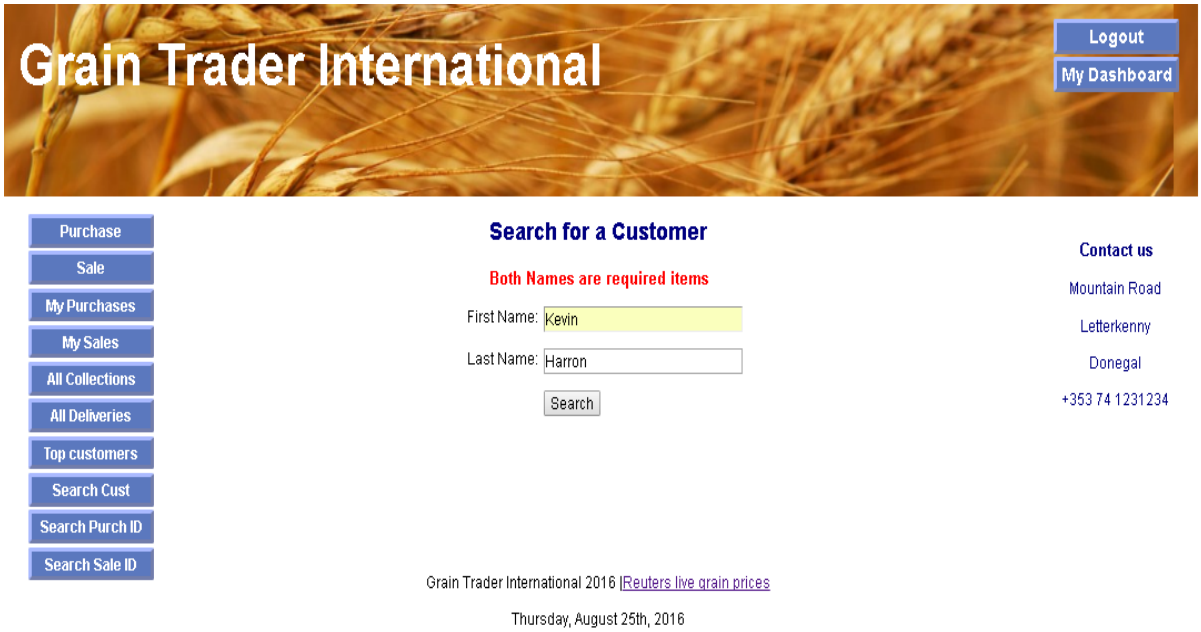


Figure 15 The Search Customer Function

Results are then retrieved as below.

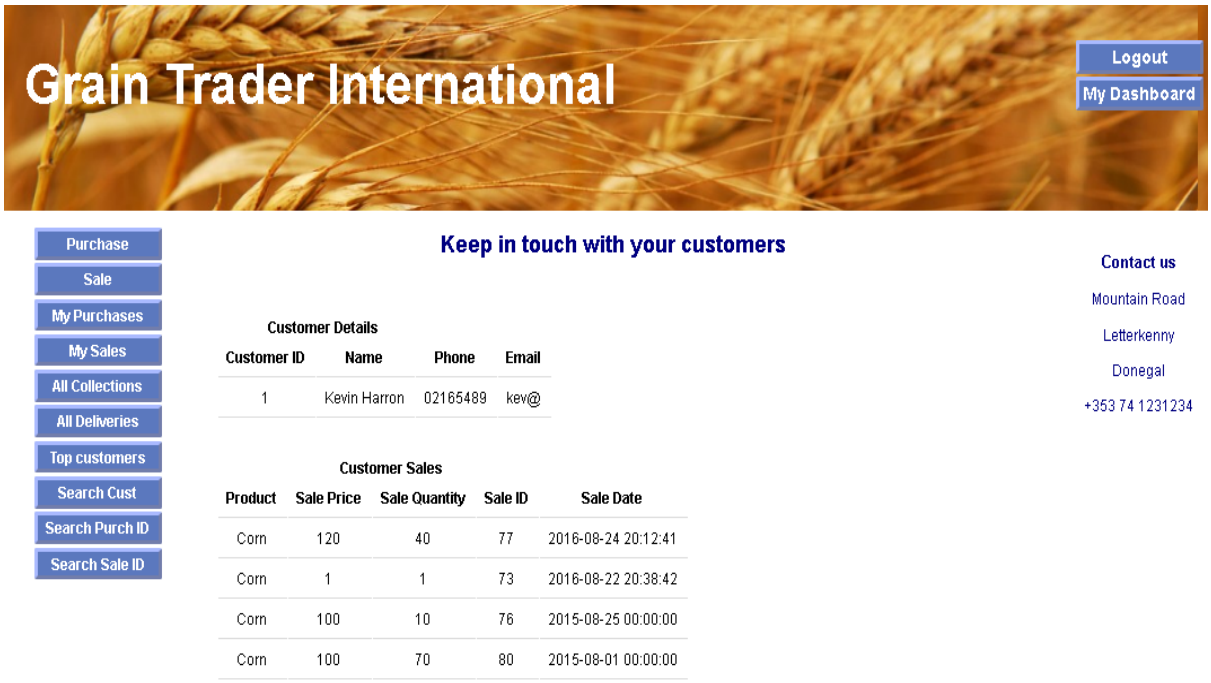


Figure 16 Search Customer Results

Each of the four traders should have a similar experience and consistency. This is ensured with the same layout, colours and buttons.

4.3, Management Dashboard

The manager has a separate overall view and so the GUI will direct this user to their own unique page. An original design had links from the Manager's Dashboard page, to each of the traders' pages but after navigating around the website mid-development, it was felt that it was distracting from the main purpose of the manager's interaction and that keeping the statistics for each product in a single centralised place on the 'Dashboard' page was preferred. The aim of the GUI is to present an instant visual snapshot of the key information the manager needs, instead of clicking back and forward through further pages.

Figure 17 Management Dashboard

4.3.1, Management Functionality

Further management functionality is available once the user wants to interact with the application, by simply clicking on a clearly labelled button.

- Total Purchases – Displays purchase totals across each product for the current year.

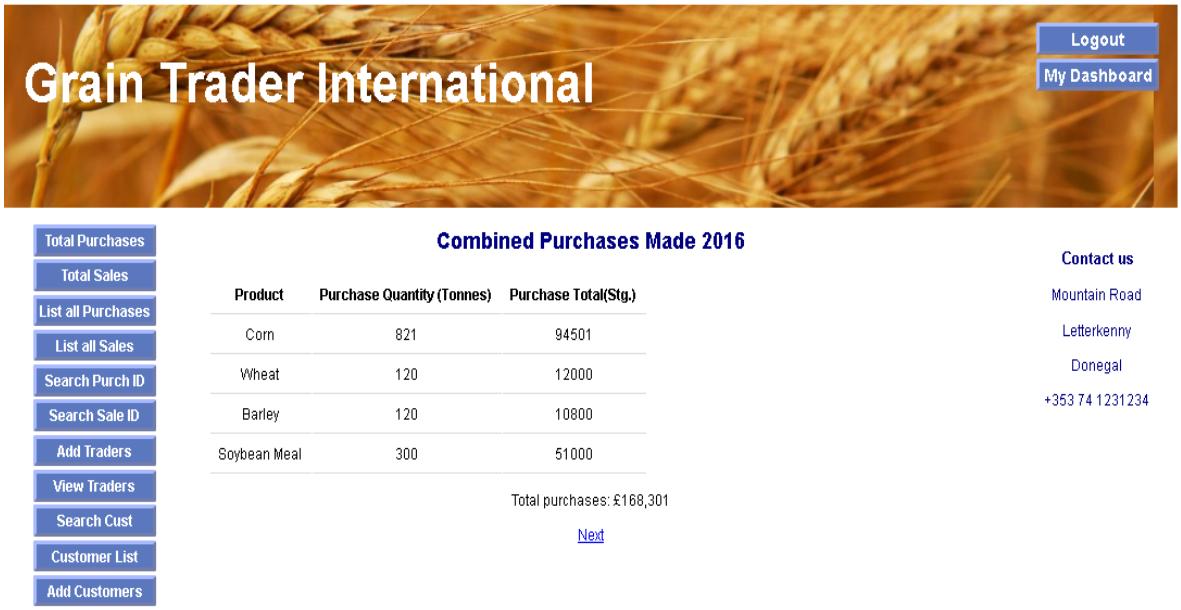


Figure 18 Combined Purchases for Management

- Total Sales - Displays sale totals across each grains product for the current year.

- List all Purchases – Lists all purchase transactions, not just current year, for all grains.

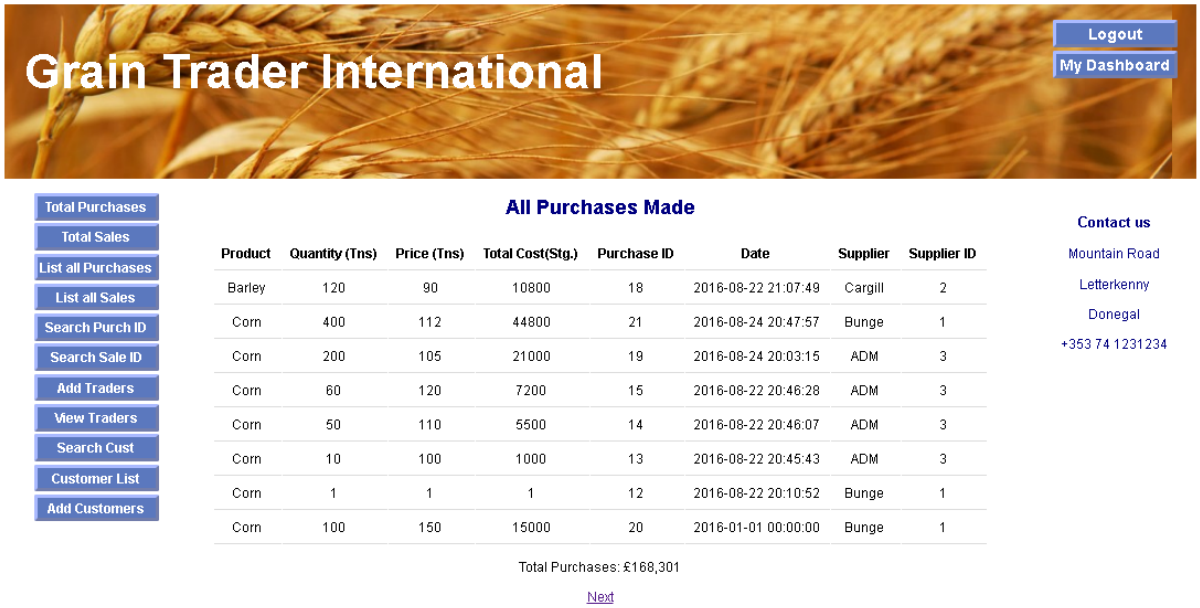


Figure 19 List all Purchases function

- List all Sales – Lists all sale transactions, not just current year, for all grains.

- Search Purch ID – Allows for a search of transactions by purchase ID.
- Search Sale ID – Allows for a search of transactions by sale ID.

Figure 20 Search by Sale ID function

Results display as follows, with particular sale and corresponding collection details.

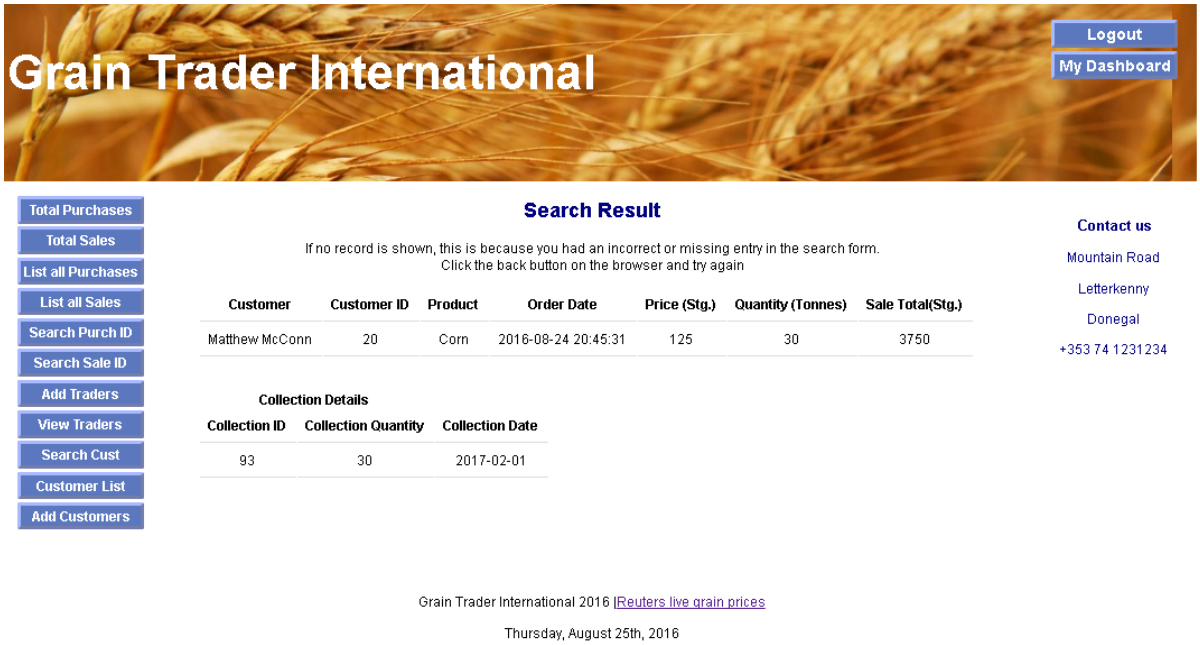


Figure 21 Search results from Search Sale ID function

- Add Traders – One click displays a form to enter new trader details, which can be submitted to the database.
- View Traders – Clicking here displays information about current traders.
- Search Cust – Allows for customers to be searched for by name to retrieve their details.
- Customer List – Displays a list of customers and their details.

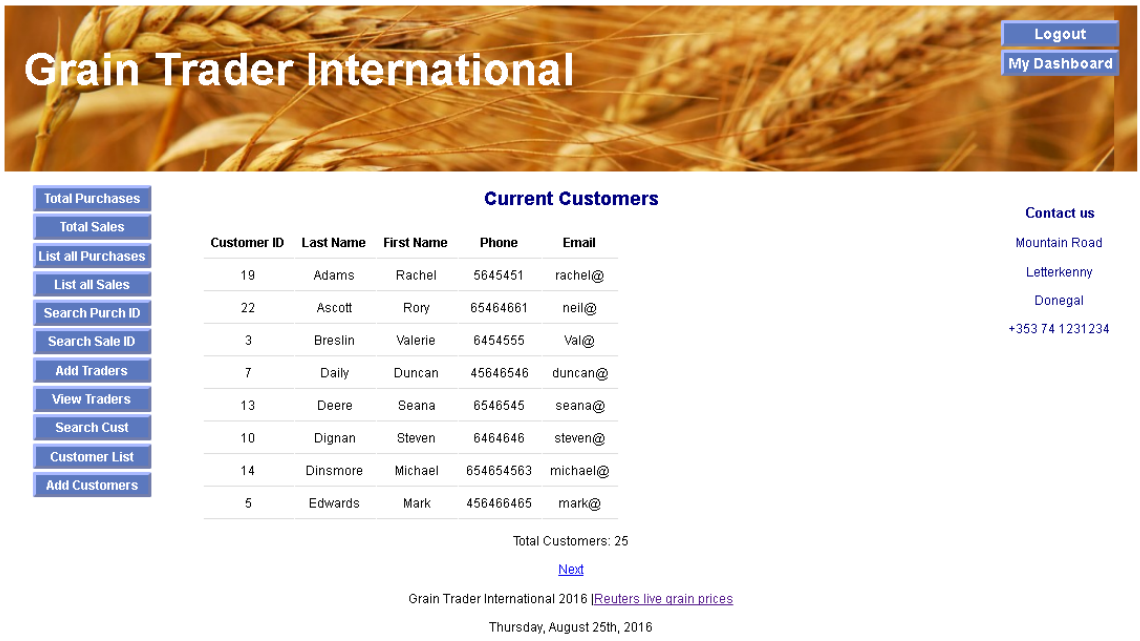


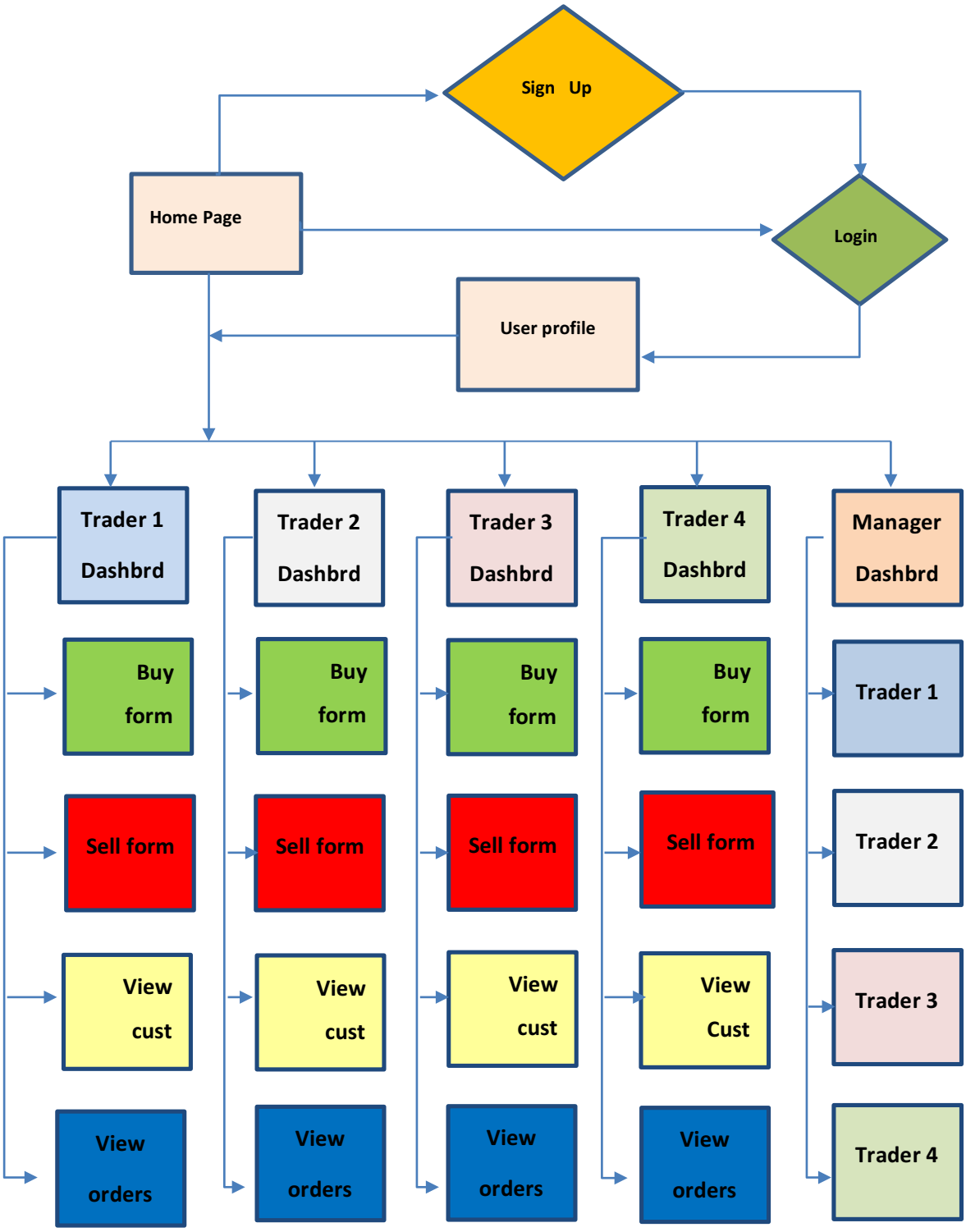
Figure 22 Customer List

- Add Customers – One click displays a form to enter new customer details, which can be submitted to the database.

4.4, How the design evolved.

The sitemap below shows the original system design, followed by the updated version. The most important functionality was to build an application that could store purchase and sale transactions and allow these to be retrieved, as well as view customers.

Figure 23 Original Sitemap of the system



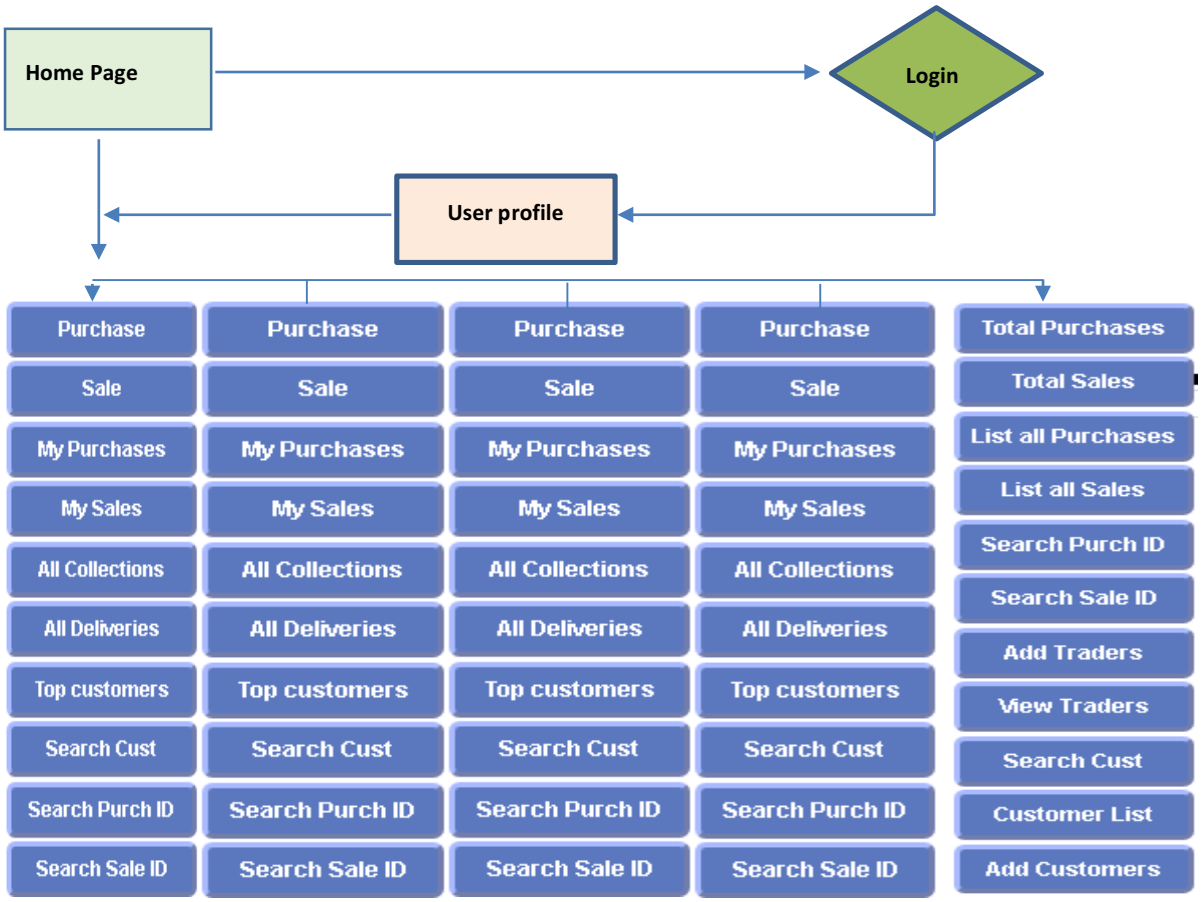


Figure 24 Updated Sitemap of the system

As development work continued, further functionalities were added. In the sitemap above, each trader is taken to a dynamically created page based on their login details, corresponding to their particular product. Data is presented and analysed in a consistent manner to the traders. Management is taken to a different page, with overall data for all products created on the fly for this user when they login. The original design had no functionality for retrieving transaction data for deliveries or collections. Given the difference in time between purchases and deliveries, sales and collections, a way of viewing this data was also necessary. Hence, this was added via the All Deliveries and All Collections tabs. These pages display all deliveries and collections respectively, not just for the current year. A chart visually representing monthly totals for Collections and Deliveries is also displayed on these pages.

The original design allowed only for viewing orders and viewing customers. However, during development, it was felt that a way of pinpointing a particular purchase, sale or customer was needed, instead of just paging through lists of transactions. Hence, the Search Cust, Search Purch ID and Search Sale ID buttons were added to facilitate a focused search of the data.

4.5, Database Design

The GUI front end allows the user to interact with the application back end/storage by entering, storing and retrieving data. MySQL is a Relational Database Management System (RDBMS) that organises data into tables that are related to each other. The 'objects' or entities are the items for which data must be stored. Each object/entity is represented by a row, with its attributes contained in the columns. Each row needs a unique identifier, called the primary key. These primary keys are then used to relate one table to another, with the subsequent use of a primary key in one table, referred to as a foreign key when it is used to relate one table to another. (Hernandez, 2013)

4.5.1, Updated Database Design

Objects and associated properties are described below. Database objects (and hence tables) are as follows, with primary keys underlined.

User/Trader/Product object – The system user represents the trader and product which are combined into one single object. Each User/product 'book' is run by one trader only. The profit and loss of that product, is the profit and loss of the individual trader who runs the book. Attributes are; user_id, fName, lName, email, psword, registration_date, user_level, product. The user_level attribute refers to the mechanism used in the database to give different users different access, so that they are directed to their own unique home page. Traders are user level 0 and management is user level 1.

Purchase object – The trader must purchase their particular product from one or more suppliers, using a purchase contract. Attributes are; purch_id, user_id, supplier_id, purOrDate, purPrice, purchaseQuantity.

Supplier object – The supplier will deliver the product on the required future date. Attributes are; supplier_id, supplierName.

Sale object – The trader will then sell the product on to customers using a sale contract. Attributes are; sale_id, user_id, cust_id, saleOrderDate, salePrice, saleQuantity.

Customer object – The customer will collect their product at an agreed future date. Attributes are; cust_id, custFname, custLname, phone, email.

Delivery object – Products can be delivered over a specific time horizon, for example, 12 tonnes of soymeal be can be ordered in January for delivery of 6 tonnes in March and 6 tonnes in April. So although the order date may be the same, the delivery dates for specific parts of an order can be

different. This is why delivery is represented in a separate object. A delivery is always as a result of a purchase from a supplier (as opposed to a sale resulting in a collection by a customer). Attributes are; delivery_id, purchase_id, Ddate, Dquantity

Collection object – Similarly, a customer can order 12 tonnes of soybean meal in January and choose to collect 6 tonnes in March and 6 tonnes in April. So although the order date may be the same, the collection dates for specific parts of an order can be different. This is why collection is represented in a separate object. A collection is always as a result of a sale to a customer (as opposed to a purchase resulting in a delivery by a supplier). Attributes are; collection_id, sale_id, Cdate, Cquantity.

The original database design had JanQty, FebQty, MarchQty, AprilQty, MayQty and JuneQty for the delivery and collection objects, but during development, it progressed to the structure above, with just a Ddate and Cdate, to allow flexibility for further dates. Additionally, when SQL queries were being created to query the database, an additional 'months' table, with just the numbers 0-11 was created, to represent the months of the year, which can be accessed by any query for date purposes.

Months object – Attributes are monthnum (0-11).

4.5.2, The database schema from MySQL is displayed below, followed by an Entity Relationship Diagram to graphically represent the relationships between the objects in the database.

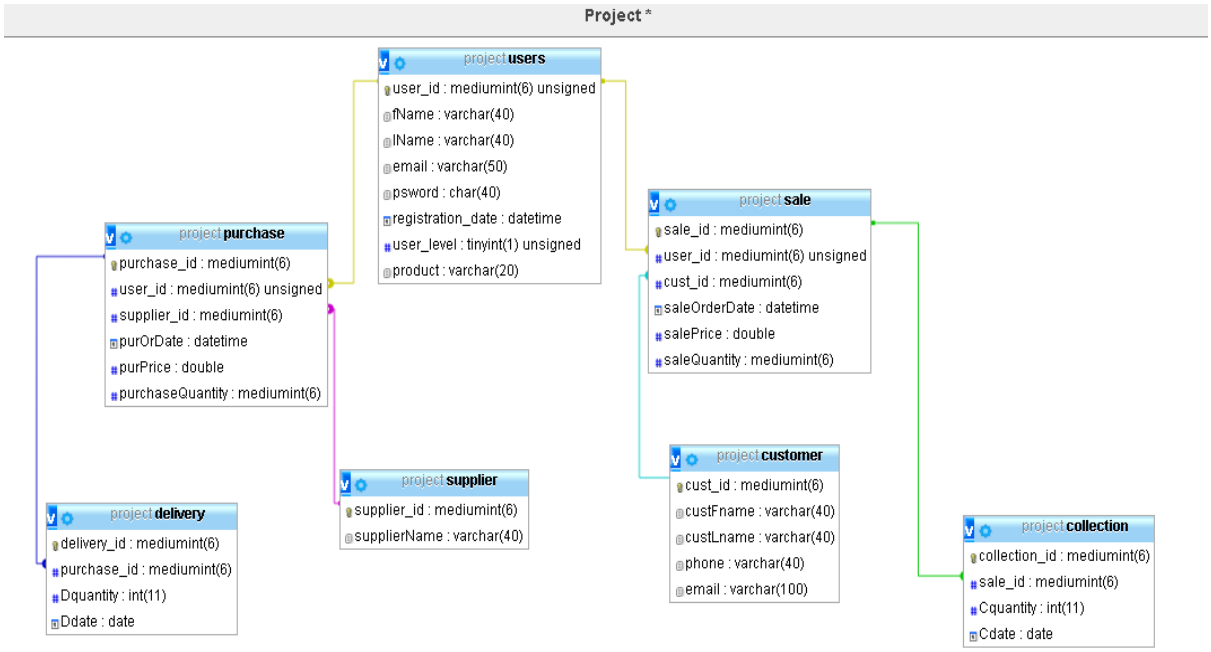


Figure 25 Database schema from MySQL Database

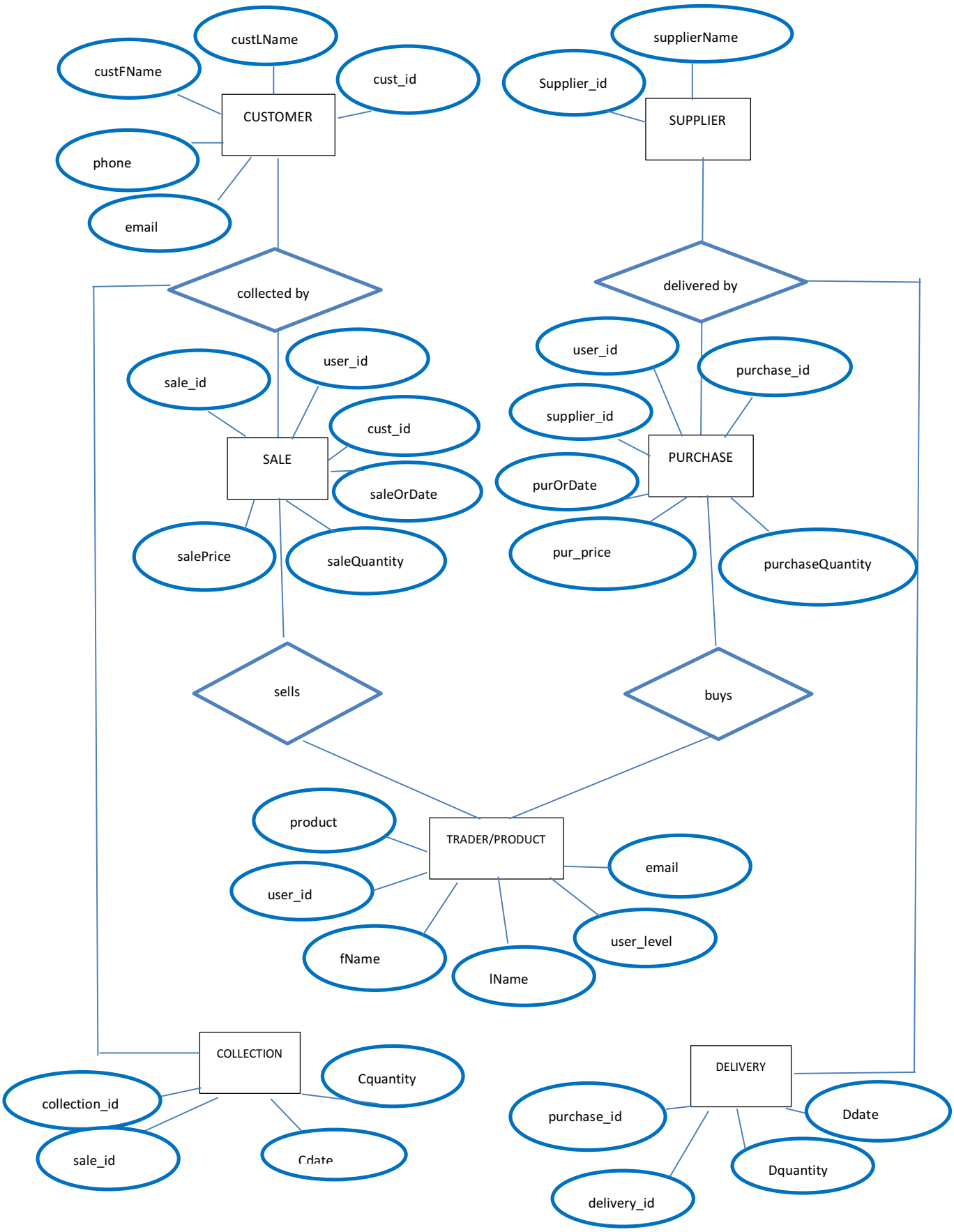


Figure 25 Entity Relationship Diagram of Database

4.6, Architectural Design

The aim of the software is to be a data storage and retrieval system which can be accessed anywhere and anytime. For this reason, a web based application has been chosen, which means the standard client/server model is appropriate. A browser acts a universal client for any web application. The user therefore, has a browser as their interface with the system, where they send requests to the server, which holds the database and the information the user sends to it, or requests from it.

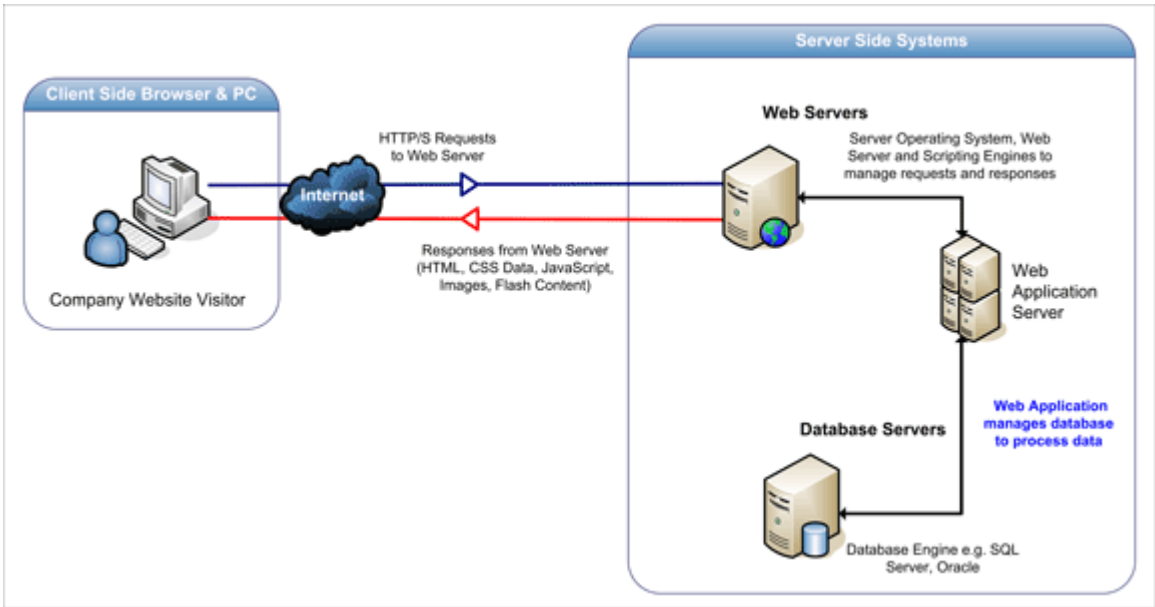


Figure 26 Client/Server Architecture (acunetix, 2016)

4.7, Languages

Client side languages learned for this purpose are HTML5 and CSS. PHP was also learned for use as the server-side scripting language. This allows communication with the database server via PhpMyAdmin to send queries to MySQL. Apache, the free web server is used to deliver pages to the client.

The application seeks to follow the n-tier approach, with three tiers. (acunetix, 2016)

Presentation Tier – web browser provides the GUI and means for user to send a request.

Application Tier – application logic using PHP which services user by making queries and updates.

Storage tier – MySQL database.

This model also loosely follows the Model View Controller approach (MVC). (Zend, 2016). The application consists of three interconnected parts in order to separate internal representations of information from how the user actually sees it on the GUI.

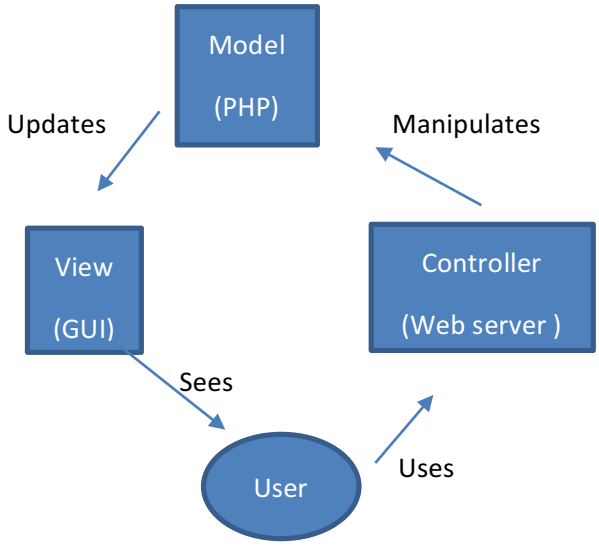


Figure 27 MVC Framework (Wikipedia, 2016)

Chapter 5: Implementation

5.1, A reminder of the Technology tools required.

- A lap top or PC
- Notepad text editor.
- XAMPP which contains Apache Server, MySQL database, support for PHP.
- Programming Skills – HTML5, CSS, PHP, SQL, Javascript.
- Fusion Charts.

5.2, The Agile Approach

The worst case scenario for any project is failure, despite the expenditure of significant resources. This project used an Agile approach to try and alleviate some of that risk, instead of the traditional Waterfall approach to project management. (Layton, 2012). The Waterfall approach was developed in the 1940s and 50s, when the most important part of computing was the hardware itself, with room sized computers. The software aspect took a back seat and manufacturing processes were more appropriate. The process had a logical sequence whereby one phase is completed in its entirety, before the next phase is undertaken.

1.Requirements

2. Design

3.Development

4. Integration

5. Testing

6. Deployment

According to the Standish group, 24% of projects failed completely, 44% were challenged (costs and time not as expected) and 32% succeeded. (Layton, 2012) Planning has to occur up front with a Waterfall approach, when least is known about the project. There is no opportunity for change or

learning as the project progresses and a wish list of everything that is needed or may be needed is produced up front, leading to 'Scope Bloat'. According to the Standish group, 64% of software features are rarely or never used.

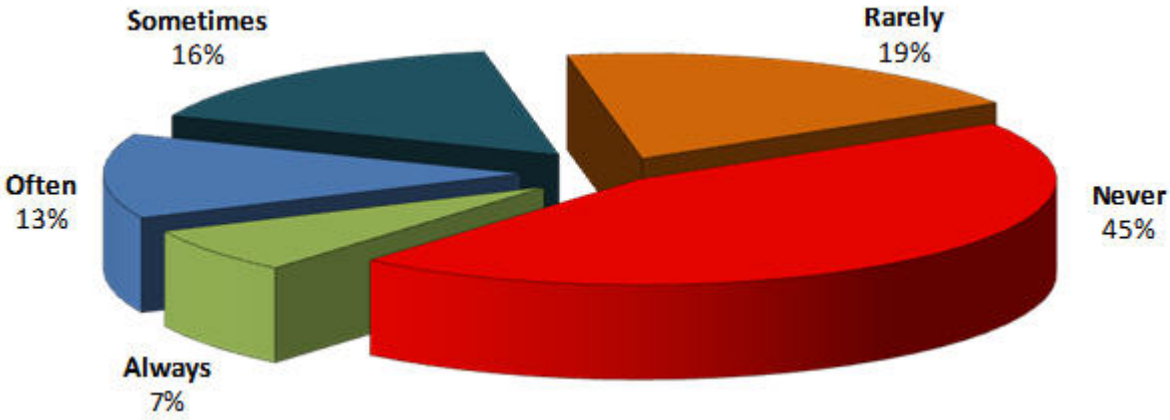


Figure 28 Features of Software actually used (Layton, 2012)

The Agile approach emphasises the prioritisation of features, delivered via smaller segments of the overall project called 'iterations'. The aim is for continuous delivery of working aspects of the software, so that there is always some value to the customer. This is progressively added to with further learning and testing, instead of waiting until the very end of the time horizon for a piece of software that may or may not deliver. The most important features should be delivered earlier with the opportunity for feedback to be integrated.

5.2.1, An Agile solution.

Initial planning, analysis and design took place to identify requirements, provide an outline solution with preferred technologies and database design. These requirements translated to application functionalities in order of importance. This ensured that following development time spent, the user has a functioning piece of software that can then be added to in subsequent sprints, with the benefit of learning along the project path. It also forces development to be carried out on a priority basis, so that the most important features can be added early on, to mitigate issues down the line with resource constraints and lessen the risk of 'scope bloat'.

For the purposes of this project, sprints typically lasted between one and three days, not the two to four week iterations common in the technology industry. The flexibility of an Agile approach lends itself to being down scaled for a smaller project with one team member for this project.

5.3, Initial Iterative Sprints carried out

Sprint One – Create a database to house data and retrieve data, in a uniform manner for all users according to the proposed design. Build a HTML5 framework for the web pages. (w3schools, 2016) (West, 2013)

Sprint Two – Create a login function to allow 4 traders to log into the system with unique IDs. When logged on, the system should dynamically create a page specific to that user, with their commodity displayed. Additionally, a manager can log in but is directed to a different page as they will see different data. The use of PHP sessions and isset() method, facilitates different users and is repeated as necessary throughout the application code.

Create logout function, create Home button, Home Page based on user logged in.

```
<?php
session_start();
if (!isset($_SESSION['user_level']) or ($_SESSION['user_level']
!= 0))
{  header("Location: login.php");
   exit();
}
?>
<?php
if (isset($_SESSION['user_id'])){
$uid = "{$_SESSION['user_id']}";
}
```

Figure 29 PHP sessions to remember users

Sprint Three – Create PHP forms with the POST method to allow user input to add purchase and sale data into the database. For example, the following code presents the user with a GUI form to fill in for a purchase transaction. It asks for a Supplier Name, which accepts user input and posts it to the supplier_id field in the MySQL database. There is a choice of four suppliers to limit the possibility of false input and make the system easier to use for the trader.

```
<form action="purchase.php" method="post">

     <br><label                                    class="label"
for="supplier_id">Supplier</label>
     <select name="supplier_id">
```

```
<option value="">- Select -</option>
     <option value="1"<?php if (isset($_POST['class']) AND
($_POST['class'] == '1')) echo ' selected="selected"';
?>>Bunge</option>
     <option value="2"<?php if (isset($_POST['class']) AND
($_POST['class'] == '2')) echo ' selected="selected"';
?>>Cargill</option>
     <option value="3"<?php if (isset($_POST['class']) AND
($_POST['class'] == '3')) echo ' selected="selected"';
?>>ADM</option>
     <option value="4"<?php if (isset($_POST['class']) AND
($_POST['class'] == '4')) echo ' selected="selected"';
?>>Glencore</option>
```

//Use SQL to add the information to the database.

```
$q = "INSERT INTO purchase (purchase_id, user_id, supplier_id,
purOrDate, purPrice, purchaseQuantity, costOfPurchase) VALUES
(NULL, '$uid','$supplier_id', NOW(),'$purpri', '$purqty',
'$totalPrice' )";
```

//PHP POST method puts the information into the database.

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
     $errors = array(); // Initialize an error array
// Check for a supplier name:
if (empty($_POST['supplier_id'])) {
     $errors[] = 'You forgot to enter supplier name.';
} else {
     $supplier_id = trim($_POST['supplier_id']);
```

Figure 30 Code for purchase form

Sprint Four – Add View my Purchases and View my Sales buttons to retrieve records from the database based on the commodity of the user who is logged on. The user views only data for the current year, 2016, not all records, with the most recent displaying first.

5.3.1, Early prototype and recommendations implemented

Following this sprint, presentation of the initial prototype software took place with basic functionality and feedback was received with specific recommendations.

- After login, the user should be presented with a Product page, not a trader page.
- Four columns should be deleted from database – purchase_id was dropped from the Supplier table and sale_id was dropped from the customer table to aid normalisation. SaleTotal and purchaseTotal were also dropped from the sale and purchase tables respectively. It is more efficient to use PHP code to calculate these values on the fly, than storing them as columns in the database.
- User ID entry for sale and purchase forms should not be included, as the application knows who has logged in.
- A drop down menu should be included with a choice of supplier names on the Purchase form, so that only one of four choices can be made, instead of allowing the user to manually enter a supplier.

These recommendations were implemented.

5.4 Post Prototype Sprints

Sprint Five - Add delivery and collection forms to input data relating to purchases and sales respectively. The original database design had the below structure for the delivery and collection objects.

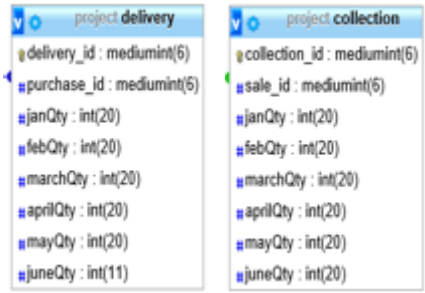


Figure 31 Original Delivery and Collection Objects

However, during development, this approach lacked flexibility and limited delivery months. There also needed to be a way to include data from previous years, if the possibility of having a function in the future to compare profit and loss from this year to last year was to be implemented.

As a result, the structure of these tables was changed to the following:

Figure 32 Updated Delivery and Collection Objects

Sprint Six– Add a search function to retrieve records from the database based on customer name, purchase ID and sale ID, based on the records for the commodity of the user who is logged on. Customers are searched by name as this is easier to remember. The purchase ID search will retrieve the particular purchase along with its associated delivery details. The sale ID search will retrieve the particular sale along with its associated collection details and the customer ID search will retrieve that customer along with their sales transactions.

The PHP request method is used:

```
$custFname=$_REQUEST['custFname'];
$custLname=$_REQUEST['custLname'];
```

Figure 33 PHP REQUEST();method used to search for a customer by name

Sprint Seven– Add 'My top Customers' button to retrieve data for the commodity of the user who is logged on, calculating total sales data for the customer for the current year and returning records in descending order of sales quantity.

5.4.1 – Fulfil requirements for accounting, risk management and business generation.

Sprint Eight – Add accounting functionality

The objective of this Sprint was to allow the user to log on and instantly be presented with a live snapshot of profit and loss, or P&L, for the current year so far. For example, is the corn trader in profit or in loss? This reflects the difference between total sales and total purchases so far in 2016. A

positive number will display as green to give an instant visual representation of the position. A negative number will display in red. A figure for 2015 is displayed for comparison purposes.

The overall stock number for the year so far will also display. As a corn trader, is the current position long or short? This figure reflects what stock in on the ground right now, in silos. It reflects deliveries minus collections for 2016 so far up to the current date. Positive numbers are in green and negative in red.

```
//Calculate the total sales by a particular trader for 2016.

$q = "SELECT SUM(saleTotal) FROM sale WHERE user_id = $uid AND
YEAR(saleOrderDate) = '2016'";

//Calculate the total purchases by a particular trader for 2016.
$q = "SELECT SUM(costOfPurchase) FROM purchase WHERE user_id =
$uid AND YEAR(purOrDate) = '2016'";
```

Figure 34 Code to calculate Total Purchases and Sales

Sprint Nine – Add Risk Management functionality.

This sprint builds on the accounting functionality of sprint three. The user will know if they are long or short of a commodity, now they need to know where their exposure is. As a trader, what is the monthly position for each of the 12 months rolling forward? A user may have an overall long position, but be short for certain months, or hold an overall short position, but be holding a long in particular months. If a trader has a negative P&L, it will be useful to drill down and see instantly where they have excess stock to sell and can concentrate efforts there. It is also extremely important from a risk management perspective to not have stock lying in silos too long, at the risk of spoiling. Similarly, a trader must have stock available in all delivery months to meet demand, or customers will go elsewhere.

This important functionality should not be presented in tabular form, but in graphical form. An outside chart vendor, Fusion Charts, supplies a free trial. (Fusion Charts, 2016) This was used to feed a SQL query to and retrieve the relevant information from the database.

The following query was used, with a similar one for collections (MYSQL Reference Manual, 2016):

```
$q = "SELECT SUM(DQuantity),MONTH(delivery.Ddate) AS MonthDate,
YEAR(Delivery.Ddate) AS YearDate
FROM months
LEFT JOIN delivery ON TIMESTAMPDIFF(MONTH,
STR_TO_DATE($firstOfMonth, '%Y%m%d'), delivery.Ddate) =
months.monthnum
LEFT JOIN purchase ON purchase.purchase_id =
delivery.purchase_id

GROUP BY months.monthnum, purchase.user_id
HAVING COALESCE(purchase.user_id, $uid) = $uid
ORDER BY months.monthnum";
```

Figure 35 Code to create chart with rolling 12 month stock levels

Sprint Ten – Add Business Generation functionality – Customer Watchlist.

This sprint builds on the functionality of sprint three and four. It displays customers who have the lowest sales volumes and where there may be scope for getting more business from competitors.

```
$q = "SELECT sale.cust_id, SUM(saleQuantity),
customer.custLname, customer.custFname, users.user_id,
users.product

FROM sale, customer, users

WHERE sale.cust_id = customer.cust_id AND sale.user_id =
users.user_id AND users.user_id = $uid AND YEAR(saleOrderDate)
= '2016'

GROUP BY sale.cust_id

ORDER BY SUM(saleQuantity) ASC LIMIT 5";
```

Figure 36 Code for Customer Watchlist

5.4.2, – What about Deliveries and Collections Transactions?

At this stage, as features were added, and tested ad hoc for functionality, it was discovered that there was not enough coverage for Deliveries and Collections. The profit and loss figures and cash amounts out and in for purchases and sales, are very separate from the deliveries and collections data which take place in the future over multiple months. For this reason, it was felt necessary to add to the functionality.

Sprint Eleven – All Collections and All Deliveries buttons were added to retrieve data for all these transactions, not just the current year, 2016. They are ordered by date of occurrence.

A chart was also created for the current year, 2016 data only, to represent total deliveries versus total collections for each month, to give the trader an instant snap shot of this year's entire activity only. (Fusion Charts, 2016) This is because the chart on the Dashboard page already displays the forward rolling 12 months data. The following query was used for Deliveries and a similar one for collections and the same chart is included for reference on both the All Collections and All Deliveries buttons:

```
$q = "SELECT sale.sale_id, collection_id, Cquantity, Cdate,
custLname, custFname
FROM sale, collection, users, customer
WHERE sale.sale_id = collection.sale_id AND sale.user_id =
users.user_id AND sale.cust_id = customer.cust_id AND
users.user_id = $uid
ORDER BY CDate ASC LIMIT $start, $pagerows";
```

Figure 37 Code to retrieve Collection details

5.5, Management functionality.

Given that the traders' home page or Dashboard, had been created, it was simply a matter of adapting and repeating the functionality for the manager page. The aim of the system is to have one administrative or manager user who is responsible for adding customers and traders to the system. It is not a publicly available system that anyone can sign up to.

Sprint Twelve - Create management dashboard with combined P&L figures for the current year, 2016. This is effectively the sum of the four traders P&L figures. At this point, it was decided that a higher

level snapshot of the combined trader positions was required, instead of a link to each of the other four traders' pages, as had previously been envisaged.

The following query was used to calculate the combined trading position, with the total purchases then being subtracted from the total sales figure:

```
$q = "SELECT SUM(saleTotal)FROM sale where YEAR(saleOrderDate) = '2016'";
$q = "SELECT SUM(costOfPurchase)FROM purchase where YEAR(purOrDate) = '2016'";
```

Figure 38 - Code to calculate total sales and total purchases for 2016

Sprint Thirteen – Create a chart to display sales totals for each of the four products for the current year, 2016. A graphical representation was again appropriate here to display the sales for each product for the current year. (Fusion Charts, 2016)

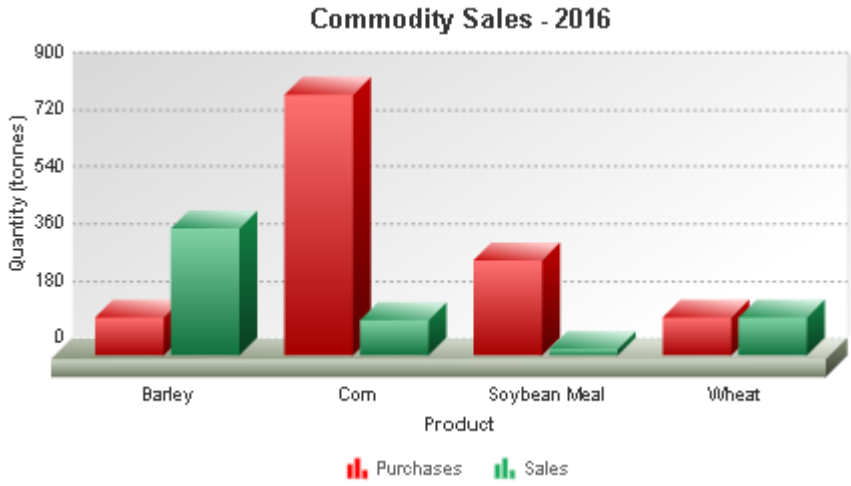


Figure 39 Chart showing total sales by product

Sprint Fourteen – Create buttons to view Total Purchases and Total Sales to display tables of the total sum of purchases and sales for each product for the current year and give more information than just the snapshot figure provided on the Dashboard.

List all Purchases and List all Sales buttons to retrieve all transactions, for every year, not just the current year. For future reference, the details of all sale and purchase transactions are included, with the most recent first.

Sprint Fifteen – Search functions added for searching by purchase ID, sale ID and customer. Customers are searched by name as this is easier to remember. View traders and view customers functions added to see a list of all traders and customers. This includes all products, not the product specific search the traders view has.

Sprint Sixteen – Amend the registration process for traders to be a function accessed by management only, via Add Traders button. This functionality was also extended to include Add Customers button, to add customers to the database.

Chapter 6: Testing and Evaluation

6.1, Test Cases for trader pages.

The below test cases are for specific functionality testing purposes but also serve as a full regression test, since all the features are being tested after complete integration. Multiple browsers are also tested to allow multiple users to log in simultaneously and hence cover performance testing also. ***All of the below tests passed.***

Preconditions: User has opened XAMPP with administrator rights, and has MySQL and Apache started and running successfully.

Additional Information

Each of 4 traders can log on using the below credentials and taken to their own product page:

Username: vic@	Password: victoriamac	Product: Corn
Username: dee@	Password: deeoates	Product: Wheat
Username: david@	Password: davidjones	Product: Barley
Username: callum@	Password: callumplatt	Product: Soybean Meal

Transactions take place from the traders' point of view.

Purchases are those transactions a trader carries out with their broker, to buy a commodity for delivery as their own stock.

Sales are those transactions a trader carries out with their customer, to sell a commodity for collection by the customer.

6.1.1, Login Failure Test:

Step 001 - Navigate to: localhost/grains/index.php **Expected Result:** User is brought to login page.

Step 002 - Click Enter without entering any login credentials. **Expected Result:** An error message should be displayed.

Step 003 - Input incorrect login details: victor@ **Expected Result:** An error message should be displayed.

6.1.2, Login Success Test for multiple users:

Step 001 – Enter login details Username: vic@ Password: victoriamac **Expected Result:** User is brought to Corn Product Page.

Step 002 – Click Back button on browser to return to login screen.

Step 003 – Enter login details Username: dee@ Password: deeoates **Expected Result:** User is brought to Wheat Product Page.

Step 004 – Click Back button on browser to return to login screen.

Step 005 – Enter login details david@ Password: davidjones **Expected Result:** User is brought to Barley Product Page.

Step 006 – Click Back button on browser to return to login screen.

Step 007 – Enter login details Username: callum@ Password: callumplatt **Expected Result:** User is brought to Wheat Product Page.

6.2, Dashboard Functionality Test

One user profile can be tested in this scenario.

Step 001 – Login as Username: vic@ Password: victoriamac **Expected Result:** User is brought to Corn Dashboard with 3 functionalities.

Figure 40 Trader Dashboard

Step 002 – Functionality One – Accounting

P&L for that particular product for 2016 so far should be displayed on the left hand side of the dashboard page. The final P&L for 2015 should also be displayed. Figures for current tonnage in silos should also display, reflecting deliveries minus collections to date, this year. Future deliveries and collections are not included. Positive numbers should appear in green, negative, should appear in red. Check that the data matches what is displayed in the database. **Expected Result:** Live P&L reflects current snapshot of total sales minus purchases for year 2016, for corn. The final figure for 2015 should also appear. Stock currently in silos for Corn should also show, as described above. Positive numbers appear in green, and negative appear in red.

Step 003 – Functionality Two – Risk Management

Check the monthly stock chart displayed on the right hand side of the dashboard page. The monthly sum of corn deliveries for 12 rolling months forward should be displayed in red bars. The sum of corn collections for each month, for 12 months rolling forward, should be displayed in green bars. **Expected Result:** Data is displayed as expected and reflects the figures from the database.

Step 004 – Functionality Three – Business Generation

The lower left hand side table should display customers to call for the Corn trader who have the lowest sales volumes for the year so far. **Expected Result:** This should display the five customers with the lowest sales figures.

6.3, Test cases for Side Menu buttons.

The following test cases will test correct functionality of the left sidebar buttons as well as correct data return. Only one user profile needs to be tested.

6.3.1, Purchase

Step 001 – Click on Purchase Button. **Expected Result:** User is presented with Purchase Form.

Step 002 – Failure Test, click submit with no purchase details entered. **Expected Result:** Error message is displayed, prompting user to fill in fields for Supplier, Purchase Price and Purchase Quantity.

Figure 41 Error handling for Purchase Form

Step 003 – Insufficient data entered failure test. Fill in only one field and hit Submit. **Expected Result:** Error message is displayed, prompting user to fill in fields as above.

Step 004 – Input correct data to fields. For e.g. Supplier: ADM, Cost: 100, Purchase Quantity 20. **Expected Result:** User is taken to Delivery Page. The data is entered to the purchase table in the database with the correct date and timestamp.

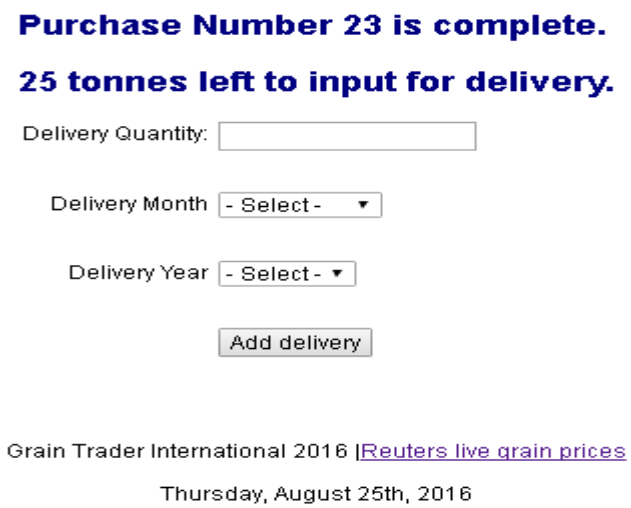


Figure 42 Form for Delivery details

6.3.2, Delivery

Step 001 - Following the previous steps, the user is now on the delivery page. **Expected Result:** The delivery page displays the purchase ID of the just completed purchase and the number of tonnes that need to be entered for delivery.

Step 002 – Failure Test – Click enter without entering any delivery details. **Expected Result:** The page displays that tonnes need to be entered for that purchase for delivery.

Step 003 – Enter partial delivery details, out of the 20 tonnes purchased, enter delivery amount of 10 tonnes and date of Dec 2016. **Expected Result:** Delivery details have been entered to database with the corresponding Purchase ID. Screen displays the remainder of tonnes needed to be added to complete delivery details for the purchase.

Step 004 – Enter remaining 10 tonnes. **Expected Result:** Screen displays that delivery details complete and delivery form no longer displays on screen.

N.B If user had entered 20 in the above step, a message would display that the delivery amount exceeded the purchase amount and the form will disappear so that no more deliveries can be added. To correct delivery details after a purchase is made, this needs to be done manually via the database. This is a security function that trades cannot be amended after the fact by traders. Ideally, there should really be a flag somewhere to highlight this discrepancy.

6.3.4, Sale

Step 001 – Click on Sale Button. **Expected Result**: User is presented with Sale Form.

Step 002 – Failure Test, click submit with no sale details entered. **Expected Result:** Error message is displayed, prompting user to fill in fields.

Step 003 – Insufficient data entered failure test. Fill in only one field and hit Submit. **Expected Result:** Error message is displayed, prompting user to fill in fields.

Step 004 – Input correct data to fields. For e.g. Customer: 1, Cost: 110, Sale Quantity 20. **Expected Result:** User is taken to Collection Page. The data is entered to the sale table in the database with the correct date and timestamp.

6.3.5, Collection

Step 001 - Following the previous steps, the user is now on the collection page. **Expected Result:** The Collection page displays the sale ID of the just completed sale and the number of tonnes that need to be entered for collection.

Step 002 – Failure Test – Click enter without entering any collection details. **Expected Result**: The page displays that tonnes need to be entered for that sale for collection.

Step 003 – Enter partial collection details, out of the 25 tonnes sold, enter collection amount of 10 tonnes and date of Dec 2016. **Expected Result**: Collection details have been entered to database with the corresponding Sale ID. Screen displays the remainder of tonnes needed to be added to complete collection details for the sale.

Step 004 – Enter remaining 10 tonnes. **Expected Result:** Screen displays that collection details complete and collection form no longer displays on screen.

N.B If user had entered 20 in the above step, a message would display that the collection amount exceeded the sale amount and the form will disappear so that no more collections can be added. To correct collection details after a sale is made, this needs to be done manually via the database. This is a security function that trades cannot be amended after the fact by traders. Ideally, there should really be a flag somewhere to highlight this discrepancy.

6.3.6, My Purchases

Step 001 – Click on My Purchases button. **Expected Result:** User is presented with a list of their Purchase transactions only for the current year.

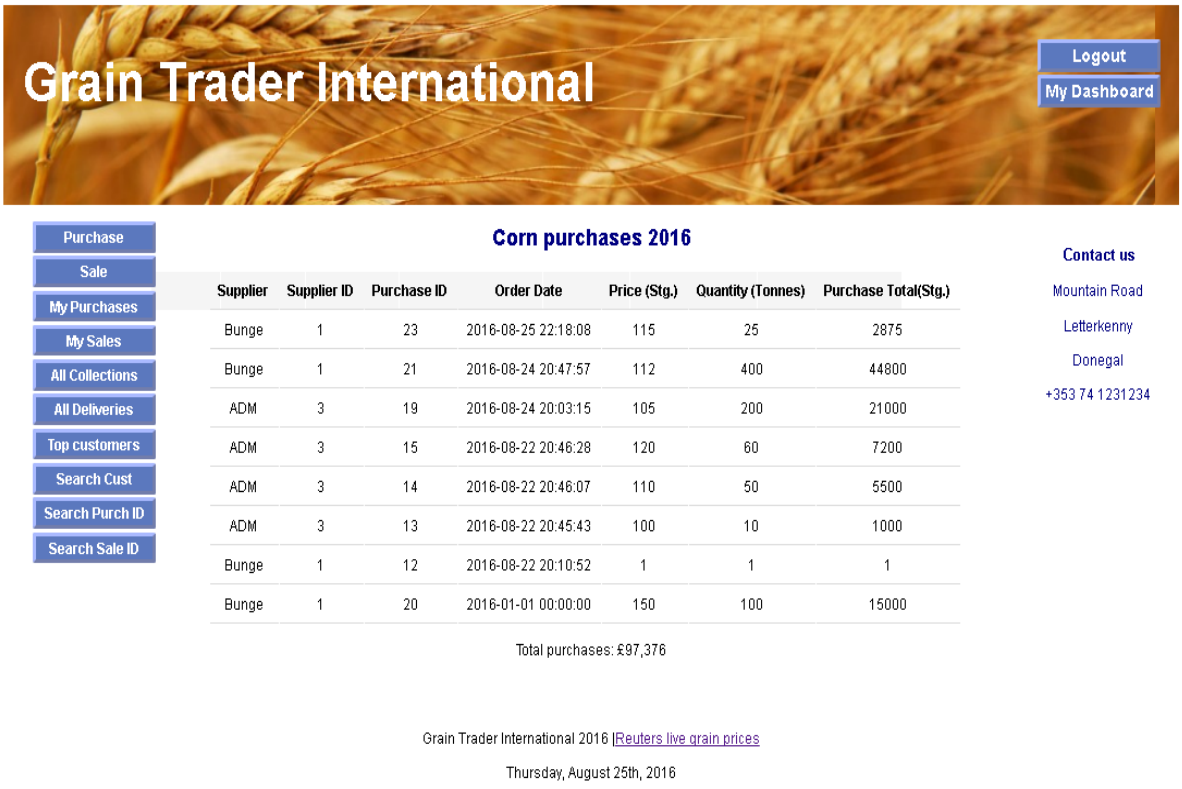


Figure 43 My Purchases Function

6.3.7, My Sales

Step 001 - Click on My Sales button. **Expected Result:** User is presented with a list of their Sale transactions only for the current year.

6.3.8, Top Customers

Step 001 – Click on top 5 Customers button. **Expected Result:** User is presented with table of up to 5 customer names, ordered by descending total sale amount for the current year.

Figure 44 Top Customers Function

6.3.9, Search Customer

Step 001 – Click on Search Customer button. **Expected Result**: User is presented with Search page.

Step 002 – Enter Customer first name – Kevin, Surname – Harron and click submit. **Expected Result:** User is taken to the result page for that customer. Customer details are displayed in a table and another table attached below provides sale details. These details match the database.

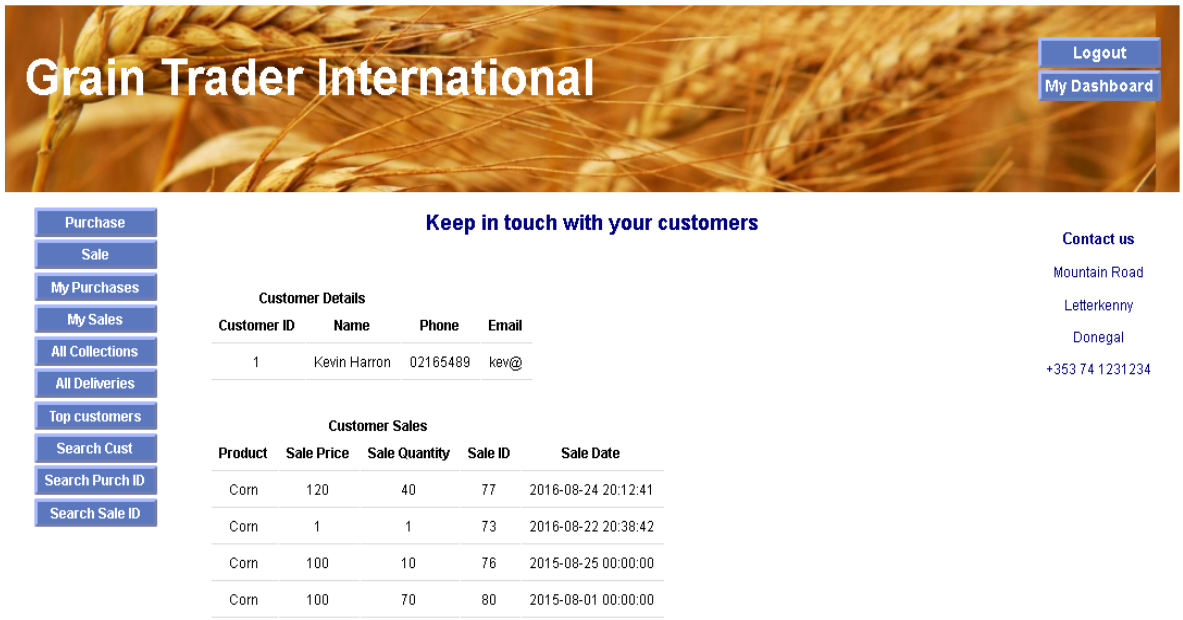


Figure 45 Search Customer Results

6.3.10, Search Purch ID

Step 001 – Click on Purch ID button.

Step 002 – Enter purchase ID 71 and click enter. **Expected Result:** The details for purchase 71 is displayed and these match the database.

Step 003 – Click on Purch ID button. **Expected Result**: User is presented with Search page.

Step 004 – Click submit without entering any details. **Expected Result:** User is presented with Search page unchanged.

6.3.11, Search sale ID

Step 001 – Click on Sale ID button. **Expected Result**: User is presented with Search page.

Step 002 – Enter Sale ID 1 and click enter. **Expected Result:** The details for Sale 1 is displayed and these match the database.

Step 003 – Click on Sale ID button. **Expected Result**: User is presented with Search page.

Step 004 – Click submit without entering any details. **Expected Result:** User is presented with Search page unchanged.

6.3.12, All Deliveries

Step 001 – Click on View Deliveries button. **Expected Result:** User is presented Delivery Page with a list of deliveries for that Product since records began. The next and previous links allow pagination where the page is full.

Step 002 – Check that Deliveries & Collections chart displays. N.B. The same chart displays for both the View Deliveries & the View Collections buttons. **Expected Result:** Chart displays Deliveries and Collections chart with total monthly deliveries and collections for the 12 months of 2016 displayed.

6.3.13, All Collections

Step 001 – Click on View Collections button. **Expected Result:** User is presented Collection Page with a list of collections for that product since records began. The next and previous links allow pagination where the page is full.

Step 002 – Check that Deliveries & Collections chart displays. N.B. The same chart displays for both the Deliveries & Collections buttons. **Expected Result:** Chart displays Deliveries and Collections chart with total monthly deliveries and collections for the 12 months of 2016 displayed.

6.4, Test Header Menu:

The following test cases will test correct functionality of the header menu buttons as well as correct data return. Only one user profile needs to be tested.

Figure 46 Header Menu Buttons on accessing the system

Step 001 – Enter login details Username: vic@ Password: victoriamac **Expected Result:** User is brought to Corn Product Page. Dashboard is displayed by default.

Step 002 – Click on Home button on the header menu. **Expected Result:** User is taken to the login page.

Step 003 – Re-Enter login details Username: vic@ Password: victoriamac **Expected Result:** User is brought to Corn Product Page. Dashboard is displayed by default.

Step 004 – Click on one of the left hand menu buttons. For eg: Search Sale ID: **Expected Result:** The header menu should now display Dashboard button as below.

Figure 47 Header Menu buttons when a button has been clicked on the side menu

6.5, Functionality Testing for Manager page.

Preconditions: User has opened XAMPP with administrator rights, and has MySQL and Apache started and running successfully. ***All of the below tests passed.***

Additional Information

The manager can log in using:

Username: test@test Password: testtest

6.5.1, Login Failure Test:

Step 001 - Navigate to: localhost/grains/index.php **Expected Result:** User is brought to login page.

Step 002 - Click Enter without entering any login credentials. **Expected Result:** An error message should be displayed.

Step 003 - Input incorrect login details: victor@ **Expected Result:** An error message should be displayed.

6.5.2, Login Success Test for manager:

Step 001 – Enter login details Username: test@test Password: testtest **Expected Result**: User is brought to Manager Dashboard Page.

6.5.3, Dashboard Functionality Test

Step 001 – The combined P&L for each product should be listed, this should be the total of the P&L figures that appear on the individual Product Dashboards, Corn P&L + Wheat P&L +Barley P&L +

Soybean Meal P&L = Manager's combined P&L figure. **Expected Result:** Combined P&L is displayed as expected and data matches that in the database.

Step 002 – The P&L for each product, Corn, Wheat, Barley, Soybean Meal should be listed. This should be a repeat of the figure that appears on each individual Product Dashboard. **Expected Result**: P&L is displayed as expected and data matches that in the database.

Step 003 – Chart of Sales Tonnage Data for 2016 for all products

A chart should display showing the tonnage for each product for the current year. The product name should appear on the X-axis with the quantity of tonnes on the Y-axis. **Expected Result:** Chart is displayed as expected and data matches that in the database.

Figure 48 Management Dashboard

6.6, Side menu buttons on the manager's page.

6.6.1, Total Purchases

Step 001 – Click on Total Purchases Button. **Expected Result:** A page is displayed with the total Purchases figures listed for each commodity for 2016.

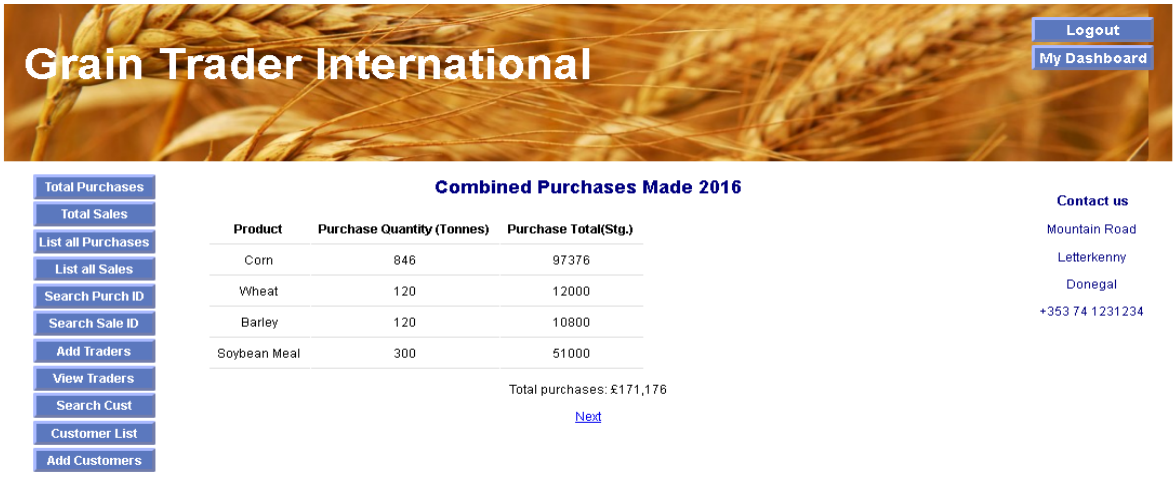


Figure 49 Total Purchases for all products

6.6.2, Total Sales

Step 001 – Click on Total Sales Button. **Expected Result**: A page is displayed with the total sales figures listed for each commodity for 2016.

6.6.3, List all Purchases

Step 001 – Click on List all Purchases Button. **Expected Result:** A page is displayed with every purchase transaction for every commodity, not just 2016.

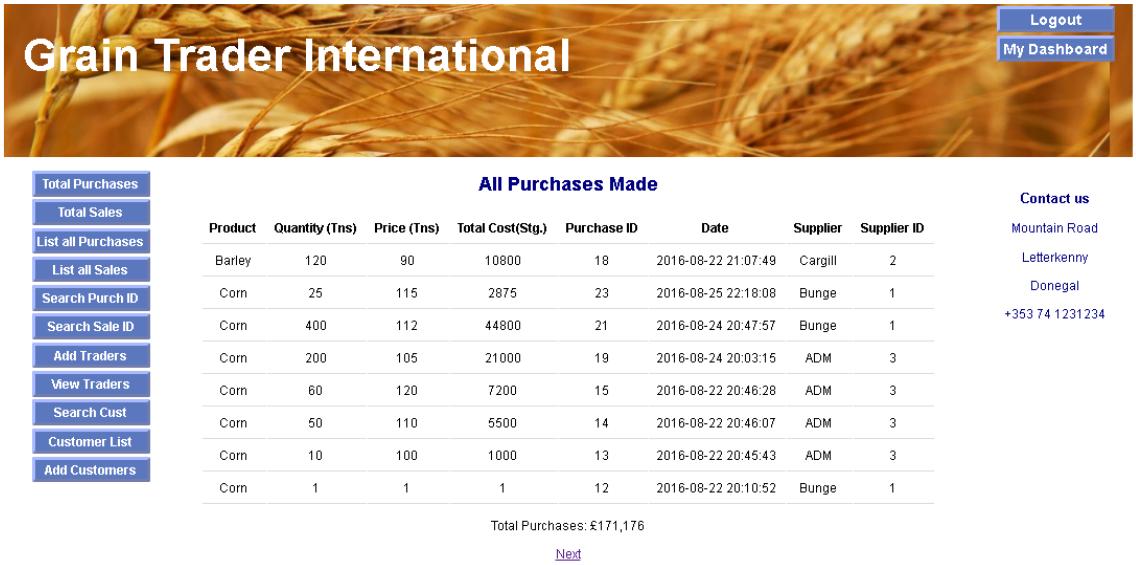


Figure 50 List of all Purchases Made

6.6.4, List all sales

Step 001 – Click on List all Sales Button. **Expected Result:** A page is displayed with every sale transaction for every commodity, not just 2016.

6.6.5, Search Purch ID & Search Sale ID - This is tested above on Trader page

6.6.6, Add Trader

Step 001 – Click on Add Trader Button. **Expected Result:** A form appears as below.

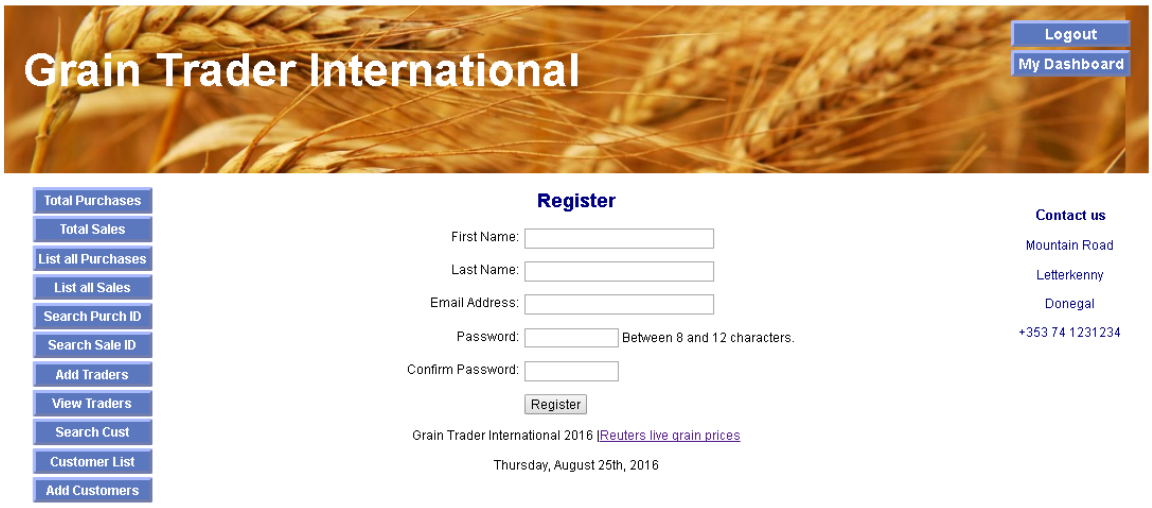


Figure 51 Add trader functionality

Step 002 - Fill in form data and hit submit to enter a trader in the system. **Expected Result:** The trader has been added to the database.

6.6.7, View Traders

Step 001 – Click on View Traders Button, **Expected Result:** User is presented with a list of all traders who have been in the system, past and current.

6.6.8, Customer List

Step 001 – Click on customer List Button. **Expected Result:** user is presented with a list of all customers who have been in the system, past and current, for all commodities.

6.6.9, Add Customers

Step 001 – Click on Add Customer Button. **Expected Result:** A form appears requesting the following; First Name, Last Name, Email, Phone number.

Step 002 - Fill in form data and hit submit to enter a customer in the system. **Expected Result:** The customer has been added to the database.

6.7, Evaluation

Have the requirements from the specification and User Stories been met? A reminder of the user stories follows;

- As a manager I want each trader to access their own product area only via a secure login.

Sprint two fulfils this requirement. Each trader can log in with their own unique profile to access their product data only.

- As a manager I want a real time birds eye view of all products combined.

Sprint twelve created a combined P&L for all traders' positions, plus a snapshot of each trader's P&L.

- As a trader I want a secure login. Sprint two fulfils this requirement. Each trader can log in with their own unique profile to access their product data only.

- As a trader I want to enter data as a purchase, with purchase data on: trader, product, supplier, tonnage, price, arrival month. As a trader I want to enter data as a sale, with sale data on: trader, product, customer, tonnage, price, collection month.

Sprint three fulfilled these objectives by providing a Purchase and Sale button to interact with the Database and record these transactions.

- As a trader I would like to access this system outside of the office.

The system can be accessed via a browser outside the office on a PC or lap top. The extent to which this can work on a mobile or tablet device has not been established though.

- As a trader I would like to be able to use this system at the same time as colleagues.

Different users can access the system via a browser.

- As a trader I would like to be able to view real time data on a product basis.

Sprints four, eight and nine fulfil this requirement. Real time data on a product basis is provided by a P&L figure and a stock figure. A trader can also see a chart of rolling monthly stock levels and collections and deliveries for the current year.

- As a trader I would like to be able to view real time data on an order basis.

Sprint six covers the above as a trader can search for a particular order by Purchase ID or Sale ID.

- As a trader I would like to be able to view real time data on a customer basis.

This has been covered in sprint six also by the facility to search by customer name. Additionally, sprint seven created a list of the top customers by descending order of sales values and sprint 10 created a 'Customer Watchlist', to alert which customers could be targeted to purchase more.

- As a developer I want the system to allow for further users to be added as needed.

Sprint sixteen covered this as new traders and customers can be added to the system via the Add Customer and Add Trader buttons.

- As a developer I want the system to be modular and flexible so that further features can be added.

It is possible for this system to be expanded to include future development. A QA (Quality Assurance/Testing) copy of the system could be created to add further enhancements which would need full functional and regression testing before being added to the Production/live version of the system that traders use every day.

6.7.1, Other functional requirements fulfilled

The following requirements have been fulfilled by the creation of a database for all users; A single centralised data storage and retrieval system, scalability, accessible by multiple users simultaneously, hence improving collaboration. Additionally, the requirements around easy install and update have been covered, as updates are server side and not on each users' device. The system has the potential for integrating with other web based services or systems. The system that can grow and change as the business does and meet future needs and users have up to date information at their fingertips.

The GUI is simple and time can be saved by avoiding spreadsheet maintenance and by accessing data on the move via a browser. This leaves more time to spend on client visiting, improving relationships. There is improved risk management as stock can clearly be seen for each month with the collection and delivery data displayed. There is greater transparency over traders' positions as an automatic overall bird's eye view of combined positions is available for management. There are also opportunities for business generation via observation of customer patterns.

6.7.2, Room for improvement.

However, the below points could be improved upon.

- Improved data integrity - There is improved data integrity with regards to Excel formulas and functions. There is an issue though when sales and purchase transactions details are entered. If the trader does not enter collection and delivery details, this could be a problem. It is easy to look up a transaction and see if there are corresponding collections or deliveries and where there are none, this can be picked up. However, it would be useful for an alert to remind a trader to add collection and delivery details or for a report to be generated where there are mismatches.

- Easier accounting, budgeting and forecasting – further reports could be added to more adequately cover this area. Budgeting and forecasting is not touched upon at all.

- Data analysis capabilities and the potential for a Business Intelligence Dashboard – Only the tip of the ice berg has been touched upon, but there has been progress made in this regard.

- A function to view data on a product basis. This is fulfilled as traders can enter and retrieve data for their product. However, an edit and delete function is not present and it is not clear if this should be assigned to the trader or just management.

6.7.3, Non-Functional Requirements

Have the non-functional requirements listed in the Analysis phase been met?

Usability & Accessibility– The system is easy to navigate and use.

Efficiency – Data can be entered quickly and stored quickly with good system performance and data integrity.

Reliability – The system runs in the required manner with error handling as appropriate.

In summary, the objective of this software is to provide a web based, database driven application that can be used securely by multiple users. This is an alternative to users keeping their own Excel records in different formats, which are then collated onto one master Spreadsheet for an overall view to be created. The web based nature of the application means that multiple users can use it from any location and potentially across different devices. It is not an exact science measuring 'improved collaboration' or less risk of error, but the following concrete improvements can be identified.

1. Multiple users can use the same system instead of separate spreadsheets.

2. Data is viewed on a consistent basis, not different formats.

3. Live data is dynamically and accurately created.

4. P&L, stock levels and customers who may be low in stock are instantly presented. This was not the case previously with Excel spreadsheets. Advanced capabilities were not being exploited in Excel, due to lack of knowledge and time. Traders are not Excel programmers or data analysts.

5. Mobility is possible – Excel spreadsheets were kept on the office hard drive and copying from here is prohibited.

Chapter 7: Conclusions and Recommendations

7.1, Aim and Objectives

The aim of this project is to create a single, centralised data repository for commodity trade transactions that can be uniquely accessed by multiple users simultaneously, via a browser and can not only store and retrieve data, but add value to that data for accounting, risk management and business generation purposes. Data should be collected from and presented back to traders, for their required commodity only, but in the same format, to ensure data consistency. Management should have an overall snapshot of the combined traders' positions. The project seeks to add value to the data presently gathered, provide mobile access, enhance collaboration, and improve data integrity. The current system used by traders is a personalised spreadsheet with individual formatting and processing of data. A master spreadsheet is collated from these for management to get an overall view. Traders can only access information from their hard drive in the office. This project seeks to find an alternative solution which modernises the current process.

The Objectives of this project are as follows;

- Complete an analysis of the problem and discover what can be achieved by finding an alternative to Excel.
- Gather requirements of the traders to inform the design and functionality of the system.
- Design a simple GUI for traders to interact with the system on a browser, to record and retrieve transactions, as well as extra functionality for data analysis.
- Create test cases for functional, regression and performance testing of the application.

7.2, Project path

Following analysis of the problem, research was carried out to explore what could be achieved by replacing Excel. Stakeholders were identified and their requirements gathered to inform the design of the system. Tools and technologies were researched to find the appropriate fit for the design and architecture of the system. Knowledge of programming languages was gained along the project path. Development work then took part in iterative sprints, with a prototype ready after sprint four. Feedback was received to change the home page from a trader page to a product page, as traders may change or be absent from the office, but the commodities traded will remain the same. The database structure was also improved by removing redundant fields. As the project evolved, the

collection and delivery tables needed to be changed to allow for more flexibility and better chart creation for analysis purposes. It was also discovered that there was not a way of reporting back transactions on deliveries and collections and so a functionality to view these was added, as well as a graph for visual representation of stock levels.

7.3, The final system

The final system has a simple GUI for traders to interact with the system via a browser to record purchase and sale transactions in the MySQL database used for the backend of the system. Clearly labelled buttons allow the trader to retrieve these transactions and search for a particular transaction or customer. The Home page or Dashboard for each trader instantly displays data relevant to their transactions only. At a glance their P&L (Profit and Loss) can be seen for the current year, their 12 month rolling stock levels and 'Customer Watch' List of customers to call. This functionality seeks to fulfil the goals of a single, centralised data repository and the use of data for information on accounting, risk management and business generation purposes.

7.4, Lessons Learned

The project was developed using Notepad and while this did not present an obstacle, time could possibly have been saved at various points by using an IDE such as Visual Studio Code or Dreamweaver. An attempt to use these was made during the project path but there was not enough time to spend on learning how to use these adequately. In retrospect, time was a scarce commodity during this project and ideally this should have been the sole focus of the developer and not one of many concurrent projects. The Agile methodology used was an appropriate approach as the little by little iterative sections helped the project move along consistently. Having a separate Design and Analysis phase and Design and Implementation phase was an important contributor to the successful completion of the project as it allows for focused development time and is in keeping with the spirit of an Agile methodology.

7.5, Future Development/Recommendations

The following are recommendations for future development that were outside the scope of this version of the system.

1. Multiple currencies

In a real world scenario, the users of this system will buy in Euros from European suppliers and in buy in USD from US suppliers. Commodities will then be sold to customers in Northern Ireland in Sterling

and in Euros to customers in the ROI. Thus three main currencies are involved, Euros, Sterling and US Dollars. These currencies are also bought on a forward basis to match the delivery dates. An extension to the system could also have link to live currency spot and forward rates and market analysis so that traders can build currency risk into their decision making. Transactions could then also reflect the correct currency and currency reserves could also be managed.

2. Budgeting and forecasting capabilities.

A way of storing and comparing monthly and yearly budgets could be added to allow traders to identify at a glance what targets and expectations are. This can be compared to the current position to give an idea of progress through the year. It would let managers see at a glance who is on track for their targets.

3. More data analysis

Due to the nature of the data storage in a database, it means that different queries can be performed to access the data in different formats as needed by the user. The charting capabilities that have been integrated mean that the data can be used to return different analysis options, for example more analysis of past transactions to help identify patterns in customer behaviour or prices, to aid future decision making.

4. Alerts for thresholds

Once stock reaches a certain level, it would be useful to have an alert to signal this event. This can be for the stock in silos or the stock that customers have purchased.

5. Different Port locations

The five national port locations could be incorporated to add more information for the collection and delivery transactions.

6. Other data functions

An Edit/Delete function could be added. A filter could be added on dates to reduce the size of reports on the page and drill down to necessary data more quickly. The page displays that are generated could be exported to PDF and saved, printed and emailed.

In conclusion, the main user requirements of this version of the system have been met. There is scope to add numerous enhancements to the system. While this project is focused on commodity trade

transactions, it is noted that the same principles could apply to any business where there is a need to store data and retrieve it. Where data is collected, it follows then that useful information can be gleaned in this new 'Big Data' era. The natural order of things has been to default to Excel spreadsheets for far too long, as the all-seeing, all doing solution. Maybe it's time to go against the grain.

References

acunetix, 2016. *Web applications: What are they? What of them?.* [Online]
Available at: http://www.acunetix.com/websitesecurity/web-applications/
[Accessed January 2016].

Baptiste, J. L., 2010. *jasonlbaptiste.com.* [Online]
Available at: http://jasonlbaptiste.com/startups/microsoft-excel-is-the-worlds-most-used-database/
[Accessed January 2016].

BobsGuide, 2012. *ClusterSeven: Regulators Warn on Spreadsheet Risks in Quest for Data Excellence.* [Online]
Available at: http://www.bobsguide.com/guide/news/2012/Oct/18/clusterseven-regulators-warn-on-spreadsheet-risks-in-quest-for-data-excellence.html
[Accessed January 2016].

Delisle, M., 2012. *Mastering PHPMyAdmin 3.4 for Effective MySQL Management.* s.l.:Packt Publishing.

Fusion Charts, 2016. *Fusion Charts.* [Online]
Available at: www.fusioncharts.com
[Accessed July 2016].

Gittlen, S., 2011. *Ditch the spreadsheet for better data analysis.* [Online]
Available at: http://www.computerworlduk.com/tutorial/data/ditch-the-spreadsheet-for-better-data-analysis-3257509/
[Accessed Jan 2016].

Google, n.d. *Google Search.* [Online]
Available at:
https://www.google.ie/?gws_rd=cr,ssl&ei=5OGgVonzF8PDOeHKrMAF#q=information+revolution
[Accessed January 2016].

Hernandez, M. J., 2013. *Database Design for Mere Mortals.* Third ed. s.l.:Addison Wesley.

Layton, M. C., 2012. Agile Project Management for Dummies. In: New Jersey: Wiley, p. 12.

Matthews, M., 2015. *PHP and MYSQL Web Development - A Beginner's Guide.* s.l.:McGraw-Hill.

NIGTA, 2016. *NIGTA - The first link in the food chain..* [Online]
Available at: http://www.nigta.co.uk/
[Accessed 2016].

Olivia, 2011. *Difference between MySQL and PostgreSQL.* [Online]
Available at: http://www.differencebetween.com/difference-between-mysql-and-postgresql/
[Accessed January 2016].

Stephens, R. J. A. D. P. R., 2015. *Sams Teach Yourself SQL In 24 hours.* Sixth ed. s.l.:Sams Publishing.

Suehring, S. & Valade, J., 2013. *PHP, MySQL, JavaScript & HTML5 for Dummies.* s.l.:Wiley.

Veltman, N., 2013. *SQL: The Prequel (Excel vs. Databases).* [Online]
Available at: http://schoolofdata.org/2013/11/07/sql-databases-vs-excel/
[Accessed January 2016].

Ventana, 2015. *Office of Finance Research demonstrates importance of using effective financial software.* [Online]
Available at: https://blog.ventanaresearch.com/tag/microsoft-excel/
[Accessed Jan 2016].

w3schools, 2016. *HTML - The Lanaguage for building web pages.* [Online]
Available at: www.w3schools.com
[Accessed 2016].

West, A. W., 2013. *Practical PHP & MySQL Website Databases: A simplified approach.* s.l.:Apress.

Whitehouse, A., 2014. *Read from True Sky's Blog - Too many cooks spoil the spreadsheet.* [Online]
Available at: http://www.truesky.com/too-many-cooks-spoil-the-spreadsheet/
[Accessed June 2016].

Wikipedia, 2010. *RDBMS Structure.* [Online]
Available at:
https://en.wikipedia.org/wiki/Relational_database_management_system#/media/File:RDBMS_structure.png
[Accessed January 2016].

Wikipedia, 2012. *2012 JPMorgan Chase trading loss.* [Online]
Available at: https://en.wikipedia.org/wiki/2012_JPMorgan_Chase_trading_loss
[Accessed January 2016].

Wikipedia, 2015. *Relational Database Management System.* [Online]
Available at: https://en.wikipedia.org/wiki/Relational_database_management_system
[Accessed January 2016].

Wikipedia, 2016. *Model-View-Controller.* [Online]
Available at: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
[Accessed January 2016].

Wikipedia, 2016. *Relational Database.* [Online]
Available at: https://en.wikipedia.org/wiki/Relational_database
[Accessed January 2016].

Wikipedia, 2016. *SQLite.* [Online]
Available at: https://en.wikipedia.org/wiki/SQLite
[Accessed January 2016].

Worstall, T., 2013. *Microsoft's Excel Might Be The Most Dangerous Software On The Planet.* [Online]
Available at: http://www.forbes.com/sites/timworstall/2013/02/13/microsofts-excel-might-be-the-most-dangerous-software-on-the-planet/#2715e4857a0b18788df672ae
[Accessed January 2016].

Yank, K., 2012. *PHP & MySQL Novie to Ninja.* 5th ed. s.l.:Sitepoint Pty. Ltd..

Zend, 2016. *Zend Framework & MVC Introduction.* [Online]
Available at: http://framework.zend.com/manual/1.12/zh/learning.quickstart.intro.html
[Accessed January 2016].