Stock Management and Communication website for Herb Production Company

Niall Hunt
B.A. (Mod.) Business and Computing
Final Year Project April 2015
Supervisor: Prof. Séamus Lawless

School of Computer Science and Statistics
O’Reilly Institute, Trinity College, Dublin 2, Ireland
Code available on disk attached and demonstration on niallhunt.com
Username = niall
Password = password
ACKNOWLEDGEMENTS

I would like to thank my supervisor, Professor Séamus Lawless, for his support, guidance and trusting throughout this self-proposed project. His experience was invaluable and this report would not be in the standard that it is without his help.

I would like to thank my friends, especially all of my colleagues in Business and Computing. All the long days spent in the bunker have paid off.

I would also like to thank Tom O’Hanlon, and all of the employees at O’Hanlon Herb for giving me the opportunity to develop such a system and for their invaluable insights and patients. I would especially like to thank Roy Kenny, who is responsible for me getting the job. Without him this project would have never happened.

Last be not least, I would like to thank my family for their support and understanding during this difficult year. Thanks you for putting up with me for as long as you have.
DECLARATION

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

Name:

Date:
PERMISSION TO LEND

I agree that the Library and other agents of the College may lend or copy this thesis upon request.

Name:

Date:
<table>
<thead>
<tr>
<th>Chapter 1. Introduction</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 - General Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.2 - Research Motivations</td>
<td>1</td>
</tr>
<tr>
<td>1.2.1 - General Problems</td>
<td>2</td>
</tr>
<tr>
<td>1.2.2 - Owner &amp; Management Problems</td>
<td>2</td>
</tr>
<tr>
<td>1.2.3 - Glasshouse Staff Problems</td>
<td>3</td>
</tr>
<tr>
<td>1.3 - Research Objectives</td>
<td>3</td>
</tr>
<tr>
<td>1.4 - Technical Approach / Methodology</td>
<td>4</td>
</tr>
<tr>
<td>1.5 - Overview of Report</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 2. Background</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 - The Herb Production Cycle</td>
<td>7</td>
</tr>
<tr>
<td>2.1.1 - The Business</td>
<td>7</td>
</tr>
<tr>
<td>2.1.2 - Glasshouse Infrastructure</td>
<td>8</td>
</tr>
<tr>
<td>2.1.3 - Production &amp; The Incubator</td>
<td>9</td>
</tr>
<tr>
<td>2.1.4 - Spacing</td>
<td>10</td>
</tr>
<tr>
<td>2.1.4 - Spraying</td>
<td>10</td>
</tr>
<tr>
<td>2.1.5 - Dispatch</td>
<td>11</td>
</tr>
<tr>
<td>2.2 - Domain Analysis</td>
<td>11</td>
</tr>
<tr>
<td>2.2.1 - A Large ERP System</td>
<td>11</td>
</tr>
<tr>
<td>2.2.2 - Glasshouse Management Systems</td>
<td>12</td>
</tr>
<tr>
<td>2.2.3 - Growth Analysis Software</td>
<td>13</td>
</tr>
<tr>
<td>2.2.4 - Alternative Solutions</td>
<td>13</td>
</tr>
<tr>
<td>2.2.4.1 - Inventory Management Systems</td>
<td>14</td>
</tr>
<tr>
<td>2.2.4.3 - Non-Technical Solutions</td>
<td>14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 3. Design</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 - Methodology</td>
<td>16</td>
</tr>
<tr>
<td>3.2 - System Requirements</td>
<td>18</td>
</tr>
<tr>
<td>3.2.1 - Source 1: Author</td>
<td>18</td>
</tr>
<tr>
<td>3.2.2 - Source 2: The Client &amp; Staff</td>
<td>19</td>
</tr>
<tr>
<td>3.2.3 - Source 3: Domain Analysis</td>
<td>19</td>
</tr>
<tr>
<td>3.3 - System Architecture</td>
<td>20</td>
</tr>
<tr>
<td>3.3.1 - Linux - Apache - MySQL - PHP</td>
<td>21</td>
</tr>
<tr>
<td>3.3.1.1 - How LAMP works</td>
<td>22</td>
</tr>
<tr>
<td>3.3.1.2 - Why LAMP was selected</td>
<td>23</td>
</tr>
<tr>
<td>3.3.2 - Model-View-Controller &amp; 3-Tier Architecture</td>
<td>24</td>
</tr>
<tr>
<td>3.4 - Tier 1 - User Interface Design</td>
<td>25</td>
</tr>
<tr>
<td>3.5 - Tier 2 - Functional Design</td>
<td>29</td>
</tr>
<tr>
<td>3.6 - Tier 3 - Database Design</td>
<td>30</td>
</tr>
<tr>
<td>3.7 - Security Considerations</td>
<td>32</td>
</tr>
<tr>
<td>3.7.1 - Password Protection</td>
<td>32</td>
</tr>
<tr>
<td>3.7.2 - Email Verification</td>
<td>33</td>
</tr>
<tr>
<td>3.7.3 - Data Sanitisation</td>
<td>33</td>
</tr>
<tr>
<td>3.7.4 - Error Messages</td>
<td>34</td>
</tr>
</tbody>
</table>
Chapter 1. Introduction

1.1 - General Introduction

The word communication comes from the Latin commūnicāre, meaning “to share”. Business communication is defined as “the sharing of information between people within an enterprise that is performed for the commercial benefit of the organisation”. For individuals to operate cohesively within a team, they need to have a collective knowledge and understanding so they can work towards the same goal. However, this becomes more difficult as organisations grow and are departmentalised. A physical distance between two departments in an organisation can result in a lack of communication between them and therefore a lack of knowledge necessary for making decisions. When this divide is between the higher and lower levels of an organisation it can be especially detrimental, effectively removing management's ability to make informed decisions, and leaving workers without direction and working towards goals that don't bring value to the enterprise.

In organisations that produce large quantities of stock, monitoring what is being produced is an essential part of developing this knowledge base within the organisation. For most of these organisations, the production cycle, from beginning to end, is reliant on the speed and efficiency of the system, however, for herb production factories there is a month long journey of growing, spacing, spraying and maintaining required before the product is ready to be sold to the customer. Therefore, an “out-of-the-box” stock management system isn't sufficient for the organisation's needs. Therefore, a more tailored product is required. Small and medium-sized enterprises’ (SME) don't have the capabilities to develop a system such as this “in-house”, and financing the development of one by a company that does is beyond the reach of many SMEs. This leaves them with three options: using a product that doesn't meet their needs; incurring a substantial expense to purchase a system that does; or going without.

1.2 - Research Motivations

During the summer of 2014, the author worked at one of Ireland's largest herb production factories where he gained knowledge of the industry and the operations of a typical herb production factory. He learnt how the production team planted the herbs in large batches onto
benches which were stored in the glasshouse, how they space herbs to give each plant more room to grow and spray them to prevent disease, and how the dispatch team harvest the herbs and ship them to the customers. Further details on the herb production industry will be provided in chapter 2.

As a SME, the company was experiencing the same problem presented above in that it had grown to a size where its departments were located in separate buildings. It is for this reason that the author decided to use this organisation as a case study for this report to demonstrate that the system developed meets their needs and is relevant to the organisation.

An essential part of the software development cycle is the requirements stage. Understanding what each stakeholder needs in a system requires knowledge of the processes they follow, their business and what current systems do poorly and what they do well. During these investigations, the author observed that there were certain problems that were felt throughout the organisation, but there were also problems specific to each of the stakeholders. These stakeholders include the owner of the business, the companies management and the glasshouse staff. This helped the author refine and develop his understanding of the systems needs.

1.2.1 - General Problems
- The communication within the organisation was disjointed and there were times when entire departments were left out of important conversations simply because there were unaware of the meeting taking place.
- There was no official record of the stock being produced and used.

1.2.2 - Owner & Management Problems
- The owner of the business found it difficult to communicate important decisions and plans with the staff.
- They were unaware of what was available to sell in the glasshouse and communication was difficult as the office and the glasshouse were located on different parts of the premises.
• Management was unaware of the amount of stock being used and the amount of waste being created.

1.2.3 - Glasshouse Staff Problems
• The stock's details were not available from outside of the glasshouse. Staff located in the dispatch area would have to find the bench within the glasshouse by physically searching through the ~600 available benches.
• The location of stock was also unknown.

1.3 - Research Objectives
The objective of this project is to investigate whether a prototype platform can be designed and developed that suites the communication and stock management needs of herb growing companies. The system will allow staff to communicate both publicly and privately within the organisation, and to record the creation, maintenance, and harvesting of stock as it moves through the production cycle. This system is designed for small and medium sized herb producers and will enable them to build the knowledge base on a platform where it is both accessible and secure.

The benefit of this system over it's competitors is that it will be cheaper and less complex than a large industrial system such as SAP Business One, and yet it will have more specific, niche functionality than a general stock management system. For this system to be a viable option it will need to be secure to ensure that any sensitive information being stored by the system is protected. By encrypting valuable and sensitive information, the organisation can ensure that their knowledge, does not become anyone else's knowledge. Using profile privileges, the system ensures that only certain users can gain access to sensitive information and functionality. For example, the owner of the organisation should be able gain access to administrative functionality, however, glasshouse staff should only be able to use the communication and stock management functionality.

It is important that the system does not just meet the needs of the user but that it is easy to use and that its industry specific terminology is understandable. The system needs to be easy
to learn and understand regardless of the users experience in the herb production industry and with using technology.

As an organisation grows its software system should be flexible enough that it is still usable. This system will have multiple users interacting with it at once and it will needs to store and display large quantities of information in a way that is understandable and visually appealing. It is also important that the system is designed in such as way that new features can be built on by other developers without causing the system any disturbances.

1.4 -Technical Approach / Methodology

Prior to developing the systems requirements, an analysis of the business context of the system must be conducted. Understanding this context is essential for a tailor-made piece of software as every industry has it’s own terminology, practices and rules. As the author had gained this knowledge through the direct experience of working within the organisation, this industry specific information had already been acquired prior to the beginning of the project. However, there were still four other investigations that needed to take place in order to successfully implement an effective prototype.

The first investigation conducted was to work with staff within the organisation to develop a more thorough understanding of their opinions of, and perspectives on the problem that needed to be solved. It was from this that the author developed the system specifications and requirements. Understanding the what, why and how of the problem helped add context to the investigation.

The next investigation that needed to be conducted was to research the existing systems that the company could adopt. The strengths and weaknesses of each of these systems was reviewed, with respect to their applicability to the herb production industry. This provided the author with an insight into the features that would be most beneficial to implement in a bespoke system and what features would be unnecessary for the herb production industry.

Determining which technologies would best suit this problem was an important step as there were certain characteristics that the client required in a system, making some languages and
technologies more appropriate than others. The client specified that they wanted a system that could be easily adopted and that wouldn't require much training to use. It was for this reason that the author decided that a web-based portal would be most appropriate as most of the staff had experience using websites such as Twitter and Facebook.

Lastly, once the prototype was implemented, the system was presented to the client to investigate whether it met the needs of the organisation and to determine any future improvements that could be made.

1.5 - Overview of Report

This report provides a description of the research, process and decisions that were made during the project. Many decisions were made as a result of trial and error, however many were chosen by the client or are part of the operations of the client's business. These choices and the other options will be discussed and analysed.

The ‘Background’ chapter will provide the reader with a description of the current alternatives to the problem presented and the potential advantages and disadvantages of these systems, if they were implemented in the organisation. It will also discuss the processes of a herb production factory and how these processes relate to the final application.

The ‘Design’ chapter will describe how the requirements derived from the background review resulted in the system design. This will include how the systems architecture is structured, why certain UI and functional decisions were made, how the database was structured and any other concerns or considerations.

The ‘Implementation’ chapter will discuss the technical components of the system, examining some of the functions used and the rationale behind certain decisions. An overview of the different functionality sets will be provided and the key challenges during development will also be analysed here.

Finally the ‘Evaluation and Conclusion’ chapter will discuss the results of the system run-through with the client and the challenges involved when developing for an organisation. It
will also present the final thoughts on the project including future research and whether the research objective has been achieved.
Chapter 2. Background

This chapter will contain the background information for the project. This will include providing the reader with a brief introduction to the production process of a herb production factory and the terminology used within the industry. Also included will be an investigation into the alternative solutions currently available on the market for the herb production company in question and the reasons why the proposed system is a superior alternative.

2.1 - The Herb Production Cycle

The terminology and operations of a business can be difficult for outsiders to understand. For this reason this section of the report will provide a brief overview of the information required to understand the following sections of the report. A description of how the business operates, the infrastructure and the maintenance practices will be provided.

2.1.1 - The Business

The business which is the focus of this research is a medium sized herb production factory, based in Wicklow Ireland, that sells potted herbs directly to retailers such as Tesco, Aldi and Lidl. Currently they produces between 25,000 and 35,000 potted herbs per week which includes conventional herbs such as Basil, Coriander and Parsley, as well as more unusual and seasonal herbs such as Lemon Verbena, Dill and Orange Thyme. It employs less than 60 members of staff and has an annual turnover less than €50m therefore making it a SME. Each potted herb sells for ~€1.50 and is produced, grown, maintained and shipped by the business.

The premises of the organisation consists of 2 buildings, the office and the glasshouse. The office is where the management and the owner of the business work. They interact with the customers via phone calls and email, and are the ones who agree sales with the customers. The office employees make the decisions within the organisation and they are the ones who initiate the changes within the business, such as the changes to the orders structure, which took place over the summer of 2014. When orders come in from customers they are sent to

---

the glasshouse staff to be prepared. The glasshouse is divided into 4 areas, the actual glasshouse itself, the production area, the dispatch area and the incubator. The functions of each of these areas will be discussed further in the following sections.

2.1.2 - Glasshouse Infrastructure

The glasshouse is a large metal and glass building where the herbs are grown, sprayed and watered. The growing conditions within the glasshouse are strictly controlled to ensure that the plants are kept in ideal conditions year round. The glasshouse temperature is controlled using industrial fans which spread warm air evenly throughout the building. The brightness of the glasshouse is controlled using blinds of different permeability.

The herbs are grown in batches of between 900 and 400 on storage surfaces known as benches. The best way to understand what a bench looks like is by thinking of it as having similar dimensions to an elongated professional snooker table. Approx 5m long x 1.8m wide x 17cm deep. Continuing this metaphor, what would be the felt of the snooker table, on the bench is a sheet of hard plastic with small valleys running throughout it to allow water to flow underneath the herbs (figure 1). The structure of the bench is made of a metal frame which forms the walled surrounding, similar to the wood of a snooker table. By storing large quantities of herbs together on these benches you can water entire benches quickly and at the same time. It also means that the herbs can be maneuvered throughout the glasshouse easily.

The glasshouse is made up of many lines, and each line is made up of
many benches, sitting side by side. At either end of the lines is the runway, which allows the benches to move backwards and forwards throughout the glasshouse and into the dispatch and production areas which would be located at the bottom of figure 2.

2.1.3 - Production & The Incubator

The herbs lifecycle begins in production. The staff here are responsible for planting the herbs and maintaining them as they move through the production cycle. Not every species of herb is the same and some require different soils, pots sizes and different quantities of seeds. These variables can dramatically affect the growth of the plant and whether the end result will be acceptable for the customers so it is important that the correct combination is used. Once the pot has been filled with seeds and soil, it is placed in batches of between 900 and 600, depending on the pot size, on a bench. It is then sent into the incubator until the seeds have germinated.

![Fig. 3 - Yellow ID Tag Components](image)

![Fig. 4 - Yellow ID Tag on bench](image)

Each bench is given yellow identification tags which look like the image above with information about the bench on them (figure 3 & 4). This includes:

- The type of herb on the bench, this is shortened to a three letter version of the word, e.g. Basil = BAS, Coriander = COR, Mint = MNT
- The quantity of that herb on this bench when it was originally planted
- The person who planted the bench
The date the herbs were planted in the format DD-MM

- The three digit bench reference number. The first two digits are the week number the herb was planted in, e.g. the 1st of January is 01, the 11th of February is 07, the 31st of December is 52. The third number is the day number the herbs were planted on, e.g. Monday = 1, Tuesday = 2, Wednesday = 3, etc.

The bench is then moved into the incubator which is a darkened storage area which is kept at a warm temperature and high humidity. The herbs are first stored here so that the seeds have an opportunity to germinate without having to survive the intense heat and sunlight of the glasshouse. After the bench leaves the incubator it is sent out to the glasshouse until it needs to be spaced.

2.1.4 - Spacing

When new a bench is created, all of the 900 plants are packed tightly together to get the most amount of herbs onto one bench. However, as the plants begins to grow their stalks begin to knit into one another. To prevent these plants from knitting together, you have to space them. Spacing is taking the 900 plants from one bench, and putting them onto 2 or more benches with more space in between each plant. This gives each plant more space to grow and prevents them from growing too tall and thin. This is done twice in the plants life, each time moving onto more benches and with more space between each herb. This is also done to prevent fungus from growing on the bench, which could promote the spread of disease throughout the glasshouse.

2.1.4 - Spraying

As the herbs in the glasshouse are all stored close together, disease and pests can spread from one bench to another very quickly. This can result in huge quantities of stock being thrown out and a substantial expense for the business. As a result of this, some of the herbs are sprayed with pesticides and herbicides. For a period of time after this, the herb is dangerous to consume and so it is marked with a red spray tag which displays the “spray date”. This is the date after which the herb is safe to sell again. If these herbs were to be sold to a customer, there would be a total recall of the sales for that day, which would not only
result in a substantial expense for the business but would also diminish the confidence the retailers has in the organisations.

2.1.5 - Dispatch

Once the herbs has reached a height of over 14 cm, it is fully grown and is made available for the dispatch team to use. Dispatch is responsible for harvesting the herbs and prepping them for their journey to the retailers. However, sometimes benches of herbs can overgrow, especially during warm summers, and must be thrown out. This is also the case when the bench catches a disease. This waste must also be recorded to ensure that the company is not incurring unnecessary expenses as a result of production inefficiencies.

2.2 - Domain Analysis

It is clear from the overview provided above that the production cycle of a herb production factory is quite different to that of a conventional manufacturer. They have industry specific needs that are currently only being met through additional efforts of their workforce. This effort involves the user spending time checking benches in the glasshouse and communicating this to the office. It is for this reason that a software system is required to support the industry-specific combination of tasks. However, there is an abundance of solutions available, all of which claim to offer a range of services and support to a business. This can make it difficult to distinguish which system or approach is in fact best suited to your organisation. These other systems will now be presented and their strengths and weaknesses will be discussed in relation to how they would meet the case organisation’s needs. With an abundance of choice available to the organisation they face the paradox of having so much to choose available, that they end up choosing nothing. However worse than this would be if the organisation was to choose incorrectly and end up incurring a substantial expense for a system that doesn't meet their needs.

2.2.1 - A Large ERP System

The first potential solution this organisation could adopt is a large Enterprise Resource Planning (ERP) systems such as SAP Business One, Oracle ERP Cloud or Microsoft
Dynamics.² An ERP system is a business process management solution that is integrated into all facets of the organisation and centralises all of the business's information into one online resource. One of the primary benefits of a system such as this is that as the business grows, it's software has the functionality and capabilities to grown with it. Large ERP systems have so many features that they are able to adapt to almost any business problem. They have the additional benefit of having an extensive development team behind them so any problem not currently addresses, can be reasonably quickly.

Industrial ERP systems come with the promise that within eight weeks of acquisition, and sometimes in as short as two weeks,³ the implementation process will be completed and the business will be on the cutting edge of technology within their industry, gaining a substantial competitive advantage within their market. However, systems such as these can be quite intimidating for SME’s as they require substantial amounts of training and process adjustment throughout the organisation in order for staff to fully adopt them.

Lastly, large ERP systems don't come cheap. An expense of this size is only justifiable if the benefits that comes from it are also substantial. If the adoption of this system was to fail, it could be an expense that puts the company into a worse positions than it is already in. A system such as this could be considered overkill for the case organisation that is facing a problem which could be solved by a system far more tailored and streamline. It is for these reasons that a large ERP system is not considered to be the optimal solution.

2.2.2 - Glasshouse Management Systems

The solution that would be most convenient and beneficial for the organisation in question would be if there was an existing system available on the market that was designed for small and medium sized herb production factories. This would mean they could simply purchase, implement and adopt the system with the knowledge that it had been developed specifically for their industry and that it has worked for similar organisations in the past. The main technologies currently within the glasshouse management industry are Grower Vertical for

Sage ERP ⁴, and Picas Finished Grower ⁵. These are large ERP systems similar to what was discussed above, only with industry specific functionality for crop and plant production. Both designed for American herb production companies, these system could be the ideal solution for the organisation as they were designed to meet the industries requirements, however, the organisations these system were designed for are factories much larger than the one discussed and, as was the case for the ERP systems, much of the functionality they process, such as processing sales orders and barcode scanning devices, are not required within the organisation yet. Although these system may be viable options when the organisation increases in size, currently they are too large for the organisation’s needs.

2.2.3 - Growth Analysis Software

The other class of technologies designed for crop and plant production industries are the growth analytics group, which contains FlowerOnTime ⁶ and Virtual Grower ³ ⁷. These systems take the location of the organisation and provide a simulation as to how long a it will take for a particular species of plant to grow based on variables such as temperature, sunlight and humidity. These system provide the user with valuable information regarding the organisation's production and harvesting schedule, however, their database of plants is currently limited primarily to flowers neither contains any data for any of the herbs the glasshouse grows. In addition to this, the production manager is a well respected expert in the field and so develops the production cycle himself. The author discussed this functionality with the client and was informed that this was not a problem that the system needed to address and so it was omitted from the system requirements.

2.2.4 - Alternative Solutions

As presented above, an industrial ERP system can be a substantial step for a SME to take. Incurring the expense and having to adapt to the alterations to the organisation can be too much to take at all once. For this reason an organisation may wish to address one aspect of the organisation at a time by adopting smaller systems.

⁵ Picas website - http://www.issol.com/products.htm
⁶ FlowersOnTime - http://msue.anr.msu.edu/news/flowers_on_time_a_new_free_greenhouse_decision_support_tool
2.2.4.1 - Inventory Management Systems

Online inventory solutions such as Unleashed 8, Stitch Labs 9 and inFlow 10, could be adopted by the organisation to begin capturing the production cycle information. The most suitable of these would be Unleashed as it is specifically designed of SMEs and possess functionality that could bring substantial value to the organisation. Unleashed provided a visually appealing and user friendly interface that enables the user to manage their inventory. In addition to this, the system provided the user with valuable information regarding the expenses of the product, and the sales of the organisation. One of the most valuable aspects to of Unleashed is the production module functionality. As many different components go into producing an potted herb, it can be difficult to accurately calculate the cost of producing an individual plant. Unleashed provides this information to the user, thus enabling them to make informed decisions regarding which herbs are going to bring them the most amount of value for the lowest expense.

Despite the positive aspects of these systems, they lack much of the capabilities that this industry requires. Firstly, they possess inadequate functionality for tracking the sprays being used on the herbs and since the repercussion of selling sprayed herb is so severe, it is an aspect that needs to be addresses in a more direct way. Secondly, these systems record the location of stock based on the layout of a typical warehouse, not a glasshouse. However, this is a minor issue. Lastly, these systems possess little functionality for tracking of the age of the product. Although this too is a minor issue, it does mean that these solutions are not optimised to the organisation need.

2.2.4.3 - Non-Technical Solutions

For some organisations, technology is seen as the panacea to all business problems, however, when developing a system one much consider whether technology is an appropriate solution for the problem. A non-technical solution that was considered by the author was for the office staff to physically communicate with the glasshouse employees more frequently. It was thought that if office staff were to spend more time working within the glasshouse and

---

8 Unleased website - http://www.unleashedsoftware.com/
9 Stitchlab website - http://www.stitchlabs.com/
10 InFlow website - http://www.inflowinventory.com/
communicating with the production team, then this could potentially solve the organization's problems. However, this is just a variation of the system that the organisation currently employes where the effort of the glasshouse staff is replaced by effort from the office staff.

Although a technical solution many not always be what is required, it was agreed upon by the author and the client that an attempt to address the businesses need should be made. The investigation into whether the author could address these needs was beneficial to both parties in that the author could investigate his potential solution with the added insight of the stakeholder, and the client stood a chance of acquiring a valuable piece of software, having only invested their time and knowledge.
Chapter 3. Design

In this chapter the design decisions of the system will be discussed, along with the methodology used during development, the requirements of the system, the systems architecture and any other considerations taken during the design process. The chapter provides the reader with an understanding of how the final system was developed and the reasoning behind this decision.

3.1 - Methodology

Prior to beginning the design phase of this project, the author came to the realisation that some form of development methodology needed to be adopted to ensure that the system being built met the requirements of the client. From previous experience developing online systems of this size, the author decided that a combination of both waterfall (figure 5) and an agile life cycle would be most appropriate. Waterfall is a linear-sequential life cycle which is composed of very specific phases which need to be completed prior to progressing onto the next. It is useful for small to medium sized projects where the requirements and the technology being used are both very well understood.

However, there are disadvantages to using the waterfall life cycle which needed to be addressed. Waterfall is a high risk cycle as the development and testing of the finished system only take place in the later stages. To lessen these risks, the specifications and

Fig. 5 - Waterfall Development Lifecycle
requirements of the system needed to be fully understood and agreed upon by all of the stakeholder prior to development. Another disadvantage of waterfall is that, throughout there is a lack of client interaction, necessary for feedback and decision making. This is why the author chose to alter these stages to allow for a more agile means of development.

Agile development frameworks reduce the risk involved in the design and testing phases of a project as they ensure that value is being brought to the client in iterative and incremental stages. Development practices such as test-driven development, one to two week development interactions and evolutionary prototyping were adopted into the waterfall life cycle to alleviate these problems.

As presented in the diagram above (figure 6), the implementation and testing stages have been altered to accommodate test driven development and additional client interaction. By turning the implementation and testing stages into a development stage, the cycle becomes much more dynamic. Additional testing within this stage ensures that the features being developed are correct and have been thoroughly tested individually and collectively.
3.2 - System Requirements

As stated in the revised waterfall lifecycle, we begin by specifying the requirements of the system being developed. These requirements come from three different sources including the author’s own experience, the employees within the case organisation and from analysing the alternative solutions being provided on the market. These requirements will be used to determine how the following sections of chapter 3 will be composed and to investigate whether the project objective can be achieved.

3.2.1 - Source 1: Author

As stated in chapter one and two, the author gained a knowledge of the herb production industry from his experience working in an industrial glasshouse during 2014 and the beginning of 2015. From this, a list of fundamental requirements, necessary for the system to be a proof-of-concept prototype, was developed. These fundamental requirements are:

- Users should be able to register an account with the website. This should involve providing the system with their details and receiving email confirmation of these details. They should then be able to sign into the website using their individual username and password.

- Users should be able to communicate with each other through the website. There should be a group chat wall where all members of staff can post messages that everyone can see. They should also be able to create a private conversation with an individual and have their messages only be visible to the members of that conversation.

- Users should be able to view the stock available in the glasshouse and the details of each individual bench. They should be able to create new benches and alter the details of each bench when necessary. Users should also be able to space benches and record the details of the spray used on each bench and the system should automatically record the benches spray date.
• The system should be accessible from both inside and outside of the organisation. Once the user has been registered and provides a valid username and password they should be able to access the website’s functionality.

3.2.2 - Source 2: The Client & Staff
Some of the systems requirements were highlighted in conversations with members of staff within the organisation. These staff derived requirements are:

• Members of both the production and dispatch teams have stated that the ability to view the benches available on each line would enable faster and easier searching of the glasshouse.

• The management staff stated that they would like to be able to write comments on a bench. This would enable them to set aside specific benches for clients and to mark benches that they want to be used first, thrown out or left untouched.

3.2.3 - Source 3: Domain Analysis
Lastly, from analysing the alternative solutions available on the market, the author developed requirements that would make the system more user friendly and professional. These included:

• The system requires the functionality to selvedge an account that the user has forgotten the username or password to. Through email, they should be able to receive a reminder of their username and be provided with access to functionality that allows them to change their password.

• Users should be able to harvest benches and move them around the glasshouse without having to navigate through the website multiple times. This makes the harvesting and moving functionality more user friendly as they do not have to move between pages unnecessarily.
3.3 - System Architecture

A systems architecture is the foundations on which everything is based. It defines the structure and behaviour of the system and if done correctly can lead to a system that balances short-term needs with long-term quality. From previous experience developing systems of this nature the author understood that the architecture needs to conform to the needs and requirements of each of the stakeholders. From the requirements, there is a need for the system to communicate with a centralised server and database. For this reason a solution stack is necessary (figure 7).

A solution stack is an ordered collection of software that makes it possible to complete a particular task. Each layer of the stack connects only to the layer immediately above and below it. A layer is composed of many different technologies meaning that regardless of the composition of the stack, as long as a layer can connect with the technologies that are immediately above and below it, it will not affect the other technologies within the stack.
The solution stack being used for this project is the open source development platform LAMP (figure 8). LAMP is an archetypal model of web services solution stacks which consists of the Linux operating system, the Apache HTTP Server, the MySQL relational database management system, and the PHP programming language. LAMP is the most common solution stack used for developing dynamic websites and as a result has become the de facto development standard.
3.3.1.1 - How LAMP works

When a user enters a web address in a browser, it sends a HTTP request to a web server. A server is a type of computer which has all the LAMP components installed and running on it and like any computer it requires an operating system. For LAMP, the operating system is Linux. Linux provides a base foundation on which all the other components can run and is known for being very stable and secure as it was designed to handle demanding business needs such as network and system administration, and database management.

Apache is the web server software that handles the HTTP request from the visitors. When the visitor makes a request for a HTML or JPG file Apache searches the server, generates the HTML to represent the requested information and sends it back to the user’s browser.

However, for dynamic websites, it is not that simple. If the user requests access to a PHP file then, before Apache sends it back to the visitor, it gives the file to PHP which executes the code and generates a HTML file to represent the dynamic functionality requested. This is then returned to Apache which in turn sends it back to the user.

Often the data needed to generate a PHP file is specific to a user, function or time. This data is stored in a MySQL database, which allows the data to persist over time. PHP retrieves this data from the database, uses it to generate the HTML page, which then gets sent back by Apache to the visitor’s browser. (Figure 9)
3.3.1.2 - Why LAMP was selected

Some of the reasons why LAMP was selected for this project was its ease of development, deployment and use. However, there are three primary advantages to LAMP that made it the ideal architecture for this system.  

Cost

As an example of Free or Open Source Software (FOSS) LAMP requires no payment and so the software can be developed at no expense to the author or to the organisations. Although there are some hidden costs of support, this is addressed by the second primary advantage.

Support

LAMP was selected in part for its extensive support community. Technologies such as Apache, MySQL and PHP have vast amounts of documentation and training material available online. However, for problems that go beyond a developers capabilities, the Open-Source community consists of many reputable specialists who can provide ongoing support at a reasonable cost.

Flexibility

Lastly, the flexibility of the LAMP technologies and licensing restrictions, such as the General Public License (GNU) enables users to develop and deploy applications in a method that suits them. Combining a web server (Apache), dynamic components (PHP) and a relational database management system (MySQL) means that a website can be more dynamic than a

---

11 What lamp is and why every blogger and website operator should know about it - http://catholictechtalk.com/2011/05/18/what-the-lamp-stack-is-and-why-every-blogger-and-website-operator-should-know-about-it/
12 Understanding LAMP and its effects on web developing http://www.webopedia.com/DidYouKnow/Computer_Science/LAMP.asp
13 Top 20 open source licences - https://www.blackducksoftware.com/resources/data/top-20-open-source-licenses
WISA (Windows Server - Internet Information Services - Microsoft SQL Server - ASP.NET) \(^{14}\) system or than an Oracle’s Integrated Stack system \(^{15}\).

### 3.3.2 - Model-View-Controller & 3-Tier Architecture

The previous sections provided a description of the Architecture of the full system. However, the architecture of the code also needs to be taken into consideration. The way in which the PHP and MySQL are structured can be understood in two ways. The first is Model-View-Controller (MVC) (figure 10). MVC comes from human-computer interaction studies and is a way of understanding how a system should be designed in order to increase usability. Model represents the functionality of the system. It’s where the processing of information happens and is behind the scenes as far as the user is concerned, as is the case for PHP and MySQL in LAMP. View is the information being presented on the screen. This is the contents of the HTTP response Apache sends back to the user. Controller is how the user interacts with the system and is made up of text boxes, hyperlinks, submit buttons, etc. The controller is represented on the view and is how the user makes requests for functionality from the model.

---


The second way of understanding how the system is structured is the 3-tier architecture. This is how the code is actually structured. Code is broken into 3 different components based on it's function within the system. Tier one is the presentation layer. This includes the View and the Controller. It turns the functionality of the system into a form the users can understand and interact with. Tier 2 is the business or logic layer. This includes the functionality of the system or in other words, the PHP in LAMP. Tier 3 is the Data layer. This is the MySQL of LAMP and is where the data is stored and manipulated.

Breaking the code into 3 tiers has its advantages. It means that code is less complicated and additional components of each tier can be added without having any effect on the system as a whole, meaning that the system can grown as the organisation does. It also means that the different components of the code only have specific functions thus making a more efficient and secure system. In the following sections, each tier of the proposed system will be discussed in more detail.

3.4 - Tier 1 - User Interface Design

How a system looks is an important aspect of the design process, however, a system can look beautiful but if it’s poorly laid out and the features are hard to find and understand, it is less likely that it will be adopted. Industrial software, such as SAP Business One (figure 11), generally has a very basic and functional UI. An abundance of features means that the system has a large amount information to present on the screen.

![User Interface of SAP Business One](image.png)
This can cause the UI to become cluttered and difficult to navigate. When you have fewer features, the system is easier to use but can lack the functionality the user requires. This is a balance that the organisation and the developer needs to agree on prior to development.

When you have experience using a certain style of website, it is much easier to use again. This is why the website is being developed in a similar style to a blog, or a social network. Figure 12 below is Facebook’s login page and figure 13 below belongs to the proposed system. They both look and feel similar, they both have similar functionality and they both clearly display information on what each component does. Designing in this way means that the user doesn’t have to learn many new things when they adopt the new system.

![Fig. 12 - Login & Registration functionality of www.facebook.com](image1)

![Fig. 13 - Login & Registration functionality of developed system - Design](image2)

The website’s layout will be similar to that of a blog or an e-Commerce website (figure 14). It will be broken into different components which are designed and developed separately from the functionality so that they can be altered easily at a later stage.

![Fig. 14 - UI components of the system](image3)

One problem that many of the existing systems (figure 15) share is difficulty
with displaying large quantities of data in a means that is understandable to the user. As there could be as many as 600 benches in the glasshouse at one time this was a problem that would need to be addressed by this system as well. This information needed to be displayed in such a way that the viewer can understand what they are seeing and be able to find a specific bench quickly and easily. As can be seen in figure 16, displaying the data with enough space between each row, makes for a more visually appealing interface. In addition to this, by providing the users with the options to query the data by which line it’s in (figure 16) they can view each line individually, thus meeting one of the requirements of the staff.

Fig. 15 - Representing data in a table
To develop a visually appealing and functional UI, the author drew up Low Fidelity Prototypes of each page (figure 17). This is where an element that the system required are drawn on paper in numerous different compositions and rearranged until the combination of elements looks well together. By drawing out many different versions of the same page, the author could choose the most functional and visually appealing version. This made for a simplistic yet functional view (figure 18).

Fig. 16 - Representing data in a table in developed system
Another aspect that needed to be taken into consideration is that users may wish to access the site on their mobile devices. Since the system was designed in a three-tier architecture this is a possibility as mobile versions of the site can be added using the same tier 2 and 3 as the desktop website however, the time constraints of the Final Year Project meant that the author only had time to develop a website intended for desktops. Despite this, the system was designed with functionality that ensured that it could still be used on mobile devices. One of the design decisions made to enhance mobile use is that the system has no drop down menus. This means that it can be navigated on a mobile device without complications. The system was also designed so that functionality could be used with fewer clicks and with less loading of new pages thus reducing bandwidth usage.
3.5 - Tier 2 - Functional Design

PHP is an Object-Oriented (O-O) language meaning that a problem is broken into many sub-problems, PHP functions are developed once to solve these problems, and then they are available to use throughout the system. The major benefit of O-O programming is that the code can be maintained more efficiently as any alterations that are made to the function will take effect throughout the system. This means that the system can grow and new functionality can be added as new needs arise. Another benefit of O-O programming is having a single location to review buggy functionality, making identifying the location of bugs and errors less time consuming. PHP uses the ‘require’ function to become object oriented (figure 19).

For the proposed system, the functionality and business logic of the system is stored in the ‘function’ folder (figure 20). The function folder contains the different functionality files, one file for each major functionality set. It is kept separate from the other files but is always available to the system by ‘requiring’ the init.php file. In this file there is a require call to each of the function files. This means that anywhere within the system, the init.php file can be ‘required’, thus making all of the systems functions available to use. By having the functionality designed in this way it ensures that every aspect of the code is designed to work to its fullest potential with no errors, flaws, or unwanted side effects.
3.6 - Tier 3 - Database Design

MySQL is a relational database management system which represents a collection of data items as a set of formally described tables. Each table is a means of representing a real-world entity in such a way that it can be codified into MySQL. Databases that are designed in this way are easier to maintain, update and alter. Although MySQL offers fewer features than other major database competitors, such as Sybase and Oracle, it possesses all the functionality that is required for this system.

Four tables were developed for the proposed system:

- user
- conversation
- message
- bench

These tables were designed using an entity relationship diagram (ERD) (figure 21). This is a representation of the different entities within the herb production industry and the attributes that describe each entity. These attributes form the columns of each table and each version of the entity forms the rows.

![Entity Relationship Diagram (ERD) for database design](image)

Fig. 21 - Entity Relationship Diagram (ERD) for database design
Once the ERD has been created, it was mapped to a relational schema and normalised (figure 22). This is the process by which the tables in the database are broken down to reduce data redundancy, duplication and inconsistency.

---

<table>
<thead>
<tr>
<th>user</th>
</tr>
</thead>
<tbody>
<tr>
<td>user_id</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bench</th>
</tr>
</thead>
<tbody>
<tr>
<td>bench_id</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>conversation</th>
</tr>
</thead>
<tbody>
<tr>
<td>conversation_id</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>message</th>
</tr>
</thead>
<tbody>
<tr>
<td>message_id</td>
</tr>
</tbody>
</table>

Fig. 22 - Mapping of data to relational schema and normalising

The above relational schema is only normalised to second normal form as there is a transitive dependency between spray and spray date in the bench table. Although the database could have been normalised further to third normal form, having multiple tables would mean that complex queries to join data scattered across multiple tables would have to be created. As having the tables normalised to second normal form would have no effect on the functionality of the system, the tables were normalised to the author’s convenience. As the organisation is a SME and their data needs are modest and so the system will not be affected. Although not optimised, the database is functional and further improvements can be made in this area in the future. These improvements will be discussed further in later chapters.

3.7 - Security Considerations

Websites built using the three tier architecture can be vulnerable to attack if the necessary precautions are not taken. Whether by malicious means or user mistake, a system can be disrupted by simple inputs. A system should be designed in such a way that it can’t be damaged. This is especially the case when the information moving
through the proposed system is valuable. This information needs to be handled in a secure way for the system to be a viable option for the business. For this reason, some security considerations were taken during the design process.

3.7.1 - Password Protection
An essential part of this website’s security is the password protection functionality. When a new user is creating their profile they provide a password which must be more than 6 characters long. A six characters password can be as much as 26 times more secure than one with 5 characters \(^\text{16}\). Once a valid password has been provided, it is passed through the md5 cryptographic hash function which ensures that the password is stored in the database in an encrypted form and can't be used to access the system if stolen.

3.7.2 - Email Verification
To ensure that the user registering the profile is not a malicious computer, the system uses email verification. The user much have a valid email address, which they supply during registration. Once the system takes their details, it automatically emails them a validation link which, if followed, activates their profile. This email service is also utilized when the user has forgotten their username or password. This ensures that only the user who owns the profile can access their account.

3.7.3 - Data Sanitisation
In a system where the user is entering in information to be stored in the database there is always a chance of mistakes or attacks that could disrupt the system. Consider the following MySQL query:

```
"SELECT * FROM table WHERE column = ' " + parameter + " ' ; "
```

\(^{16}\) How secure is my password website - https://howsecureismypassword.net/
This query pulls data from the database based on specific criteria, which are provided by the user. So if the user was to enter:

‘ OR ‘1’ = ‘1"

It would cause the statement to query the database with the following:

"SELECT * FROM table WHERE column = '' OR '1' = '1';"

This query is dangerous to the system as the parameter the user provided results in the first test of the database column = ‘’ to return false, but the second test ‘1’ = ‘1’ to return true. This means that the query will always return true regardless of any other tests within the query. For this reason, prior to sending any information into the database, it is sanitized. This removes any HTML, JavaScript and malicious MySQL components that could disrupt the system or corrupt the database.

3.7.4 - Error Messages

Lastly, wherever the user interacts with a system by entering in information, the system must ensure that no error messages are displayed to the user. The system must be designed in such a way that it catches any errors or mistakes the user makes, and only returns error messages that inform them of these mistake. This ensures that no valuable information is displayed to the user as a result of system failure.
Chapter 4. Implementation

This chapter will illustrate how the requirements of the system were implemented, the rationale behind the development decisions made, and the complications which arose during the implementation process. This chapter discusses the six different aspects that were addressed during the implementation process:

- The structure of the system
- The user interface
- Interacting with the Database
- The login and registration functionality
- The public and private communication functionality
- The stock management functionality

Each of these components will be discussed in detail along with the functionality used and the challenges faced during this period.

4.1 - Establishing the structure of the system

Prior to developing any of the requirements, the author had to establish a structure in the system which ensured that the 3-Tier architecture could be maintained. As the requirements were being developed in an iterative and agile way, the system was built up feature by feature, with each new feature being tested after its implementation. The structure of the system assisted this development style as previously developed functionality could be reused throughout the system.
Figure 23 shows the structure of the website and how this establishes a 3-Tier architecture along with the MVC framework. These folder act as a central resource to the system and ensure that new components can be added with the least amount of effort. As the ‘controller & view’ label at the bottom of Fig 23 shows, as new pages of the website are needed, they can simply be added to this folder and only need to contain ‘require’ calls to the functional and user interface files.

The only file created at this stage of development was the init.php file. This stands for initialization and is where the connections to the database and to the second tier functionality files are established. Also within the init.php file is the functionality that ensures that the user is logged in. The central aspect of this website is the user functionality and developing in this ways ensures that their details are accessible throughout the website.

4.2 - Implementing the User Interface

The interface was build using HTML syntax encoded into PHP files, and CSS to format the HTML. Taking advantage of the operations of LAMP, whereby PHP and Apache return HTML to the user’s browser, the user interface of the website is structured in the same way as a
typical HTML file. However, to increase scalability and adaptability, its components are divided up into the different files of the ‘includes’ folder, shown in figure 24 below.

![HTML code and diagram]

Fig. 24 - Components of the 'includes' folder

The visual aspects of the interface are composed of two PHP files, overall-header and overall-footer, which are both in turn made up of many other PHP files. By feeding each component into one of these files, new pages can be created by calling a php ‘require’ statement on overall-header and overall-footer. This ensures that the user interface is constant regardless of the functionality on the page.

As stated in chapter 3, the user interface of the system was developed in a similar style to that of a blog, an e-Commerce website, or a social network. This ensures that it feels familiar to the end users and can be adopted with fewer complications. The styling of the website was kept minimal for two main reasons. Firstly, this is a prototype and therefore is about functionality rather than visual appeal. Secondly, this is commercial software and therefore the visual aspects are less important. For this project, the functionality of the system was the
priority to ensure that the user received a system that met their requirements, rather than just being beautiful.

4.3 - Interacting with the Database

The connection with the database was established in the connect.php file stored in the database folder. This connection is established using the mysql_connect() statement which takes the location of the database, the database username, the database password and the name of the database as parameters. The system interacts with the database with mysql_query() commands:

The update function alters the data of a row already in the database based on a specified condition.

"UPDATE `table` SET ($fields) WHERE ($condition)"

The insert function adds a new row into the database.

"INSERT INTO `table` ($fields) VALUES ($data)"

The delete function removes a row from the database based on a specified condition.

" DELETE FROM `table` WHERE ($condition) "

The select function pulls information from the database based on a specified condition.

"SELECT ($fields) FROM `table` WHERE ($condition)"

These queries are used throughout the system to alter the database and to provide the system with user, function and time specific data. This collective datastore enables all of the functions of the system to work together.
4.4 - Implementing the Requirements

As stated in the methodology section of the previous chapter, once the system’s requirements and design specifications had been confirmed, the project’s methodology changed to a more iterative and agile form (figure 25) to increase the level of client interaction and testing of new features. The author worked in 2 week development iterations, addressing just one requirement at a time, during this period. As the foundations of the system had been laid, the author could begin developing new features without unnecessary complications. The three tiers for each of these sets was only developed at the beginning of each iteration. Prior to beginning development two lists of functions were created, the first being what needed to be developed and the second, what could be reused.

Fig. 25 - Agile development iterations
The first set of requirements to be developed were the login and registration functionality set (figure 26). Much of this functionality was inspired by social networks and other password protected services. Functionality such as email verification and editing profile settings are familiar to many of the stakeholder and by developing the system in this way, it increases the systems learnability and adoptability. As the functionality and information within this system is valuable, it is important that users only have access to it when they have verified their membership by providing a valid username and password. By default, users are restricted from the communication and stock management functionality for security reasons.

4.4.1.1 - Login

When the visitor enters their username and password into the login form (figure 27), the system derives the visitors user_id from the username and calls the md5 hash function on their password. These details are then added to a `mysql_query()` (figure 28) to be compared against the contents of the database. This query looks to verify that the
combination of the user_id and, now encrypted, password is correct by comparing it against the existing records in the database. If the query returns true, then the user exists and the system can continue, however, if it returns false then the system notifies the user that their details are incorrect and they must either register and account, verify their account via the email sent to them or attempt to enter their details again.

```
return (mysql_result(mysql_query("SELECT COUNT(`user_id`) FROM `users` WHERE `user_name` = "+$username+" AND `password` = "+$password+"), 0) == 1) ? $user_id : false;
```

Fig. 28 - Login Querying the Database to validate visitor username and password

4.4.1.1.1 - $_SESSION

Once the users details have been verified by the system, a $_SESSION is started. Created by calling the PHP session_start() function, a session is an array where variables can be stored for as long as the user remains on the site or until the variables are changed. In the login.php file, when the user has provided a valid username and password, a new variable 'user_id' is created in the $_SESSION array. Once this is set, the user is logged in and can access the functionality of the website without having to verify their account again. However, when the user exits the website, or logs out, the session is ended and verification must be provided again.

4.4.1.2 - Register

When the user enters the registration page, they are presented with a form (figure 29) which specifies the details required from them in order to create a profile. As depicted in figure 26 the registration process has certain criteria that must be fulfilled before the system can progress. This information necessary for the system to function and must be present and valid.

```
$required_fields = array('user_name', 'password', 'password_again', 'first_name', 'email');
foreach($POST as $key=>$value)
{
    if(!empty($value) && in_array($key, $required_fields) === true)
    {
        $errors[] = 'Fields marked with an asterisk * are required';
        break 1;
    }
}
```

Fig. 30 - Code for verifying required fields are valid
To ensure that these details are provided, following code (figure 30) runs through each of the fields marked with an asterix and checks that valid information has been entered. When these tests have been passed, the system enters the new user details into the database.

4.4.1.2.1 - Email Functionality

When a new users information is entered into the database, the system calls on the PHP mail() function to automatically send a verification email to the email address provided. This email contains a welcome message and a URL (figure 31) that brings the user back to the website. This URL contains the user’s email address and their ‘email_code’.

![Fig. 31 - Verification URL with email address and email_code](image)

The email_code is a md5 hash of a combination of their username and the microtime at the point of registration. This results in a 32 character string which is then stored in the database. The account is verified when the user follows the link and the system confirms that the details in the database and the details in the email URL both match. If the user were to try and login to the system prior to verifying their account they would be rejected. This is because of the ‘active’ column in the database which is only updated when the account have been verified.

4.4.1.3 - Retrieving Username & Password

Although not depicted in the activity diagram (figure 26), another use of the PHP email() function within the system is the username and password retrieval functionality. If the visitor has forgotten their username, then the system takes their email address and sends them a reminder. However, when the user forgets their password, the system must generate a random replacement and email it to them, to ensure that only they can gain access to the system. This random password is generated using a random number between 999 and 999999 which is passed through the md5 hash function. The first 9 characters of that 32 character string are then set as the user’s temporary access password (figure 32). When the user access their account using this temporary password, they are then directed to the change password page of the system and can not access the system’s functionality until they replaced the random placeholder.

```
$generated_password = substr(md5(rand(999, 999999)), 0, 8);
```

![Fig. 32 - Random password generator](image)
4.4.1.4 - Once Logged In

Once the $_SESSION has been set and the user is logged in, the aside section of the interface can be updated to display theloggedin.php file (figure 33). This file included links which allow the user to update their profile picture, to log out of the system, to view their profile, to change their password and to update their settings. These additional functionalities help build the profile of the user and make the system more professional and customisable.

4.4.2 - The Public and Private Communications Functionality

Once the users are in the system, they needed to be able to communicate with one another. At this stage the author began the development iteration that addresses the communication requirements of the system. This was developed in a similar manner to the login and registration functionality, in that a list of new and reusable functions was written prior to any development. The public and private messaging services share much of the same functionality only with additional permissions on who can view what private messages.

4.4.2.1 - Public Communication

The public communication page is composed of two sections, the text box for creating new messages and the list of posted messages. The text box uses similar functionality to the registration process in that the new message is added to the database using a MySQL UPDATE command. The users_id is added as a foreign key to the message and the datetime is recorded so that the messages can be ordered and viewed newest first. Each message has a conversation_id. This is used during private messaging and so, for public conversations, is
set to the default of zero. This way public messages can be distinguished from private ones. Once the message had been posted to the database, the page would call a header function back to the same page, effectively refreshing the page automatically (figure 35).

```
header('Location: group_chat.php');
exit();
```

Fig. 35 - header functionality

4.4.2.2 - Conversations & Private Communication

Figure 34 shows how the private messaging functionality draws in two tables from the database, message and conversation. When a private conversation is established, the user who created it is inserted into conversation as user1 and the user they choose to have the conversation with is user2. When viewing the private message page, only the conversations that include the user as either user1 or user2 will be displayed. This ensures that only the users within the conversation have access to its messages. Figure 36 is an activity diagram for the private messaging service. It displays how the system restricts the user to selecting a recipient and how they must enter in required information, similar to registration page. The system captures this information using a HTML form, composed of multiple input fields. When the submit button is pressed a HTTP post method is sent to the Apache server. This information is then available to access through the $_POST array, if the information is posted from the current page, or the $_GET array, if the information is provided in the URL (figure 37).

```
niallhunt.com/read_pm.php?id=1&title=Test+Conversation+1
```

Fig. 37 - Passing of information from one page to another via the URL
This means that data specific pages can be generated based on the information being passed into that page. For the private messaging, when the user selects a recipient, the recipient’s ‘user_id’ is passed into the new_pm.php file. This file can $_GET the passed in value and insert it into the of the recipient input box. This functionality is used throughout the system whenever an entity's details need to be altered or automatically entered.

4.4.3 - The Stock Management Functionality

The final requirements to be implemented were the stock management functionality set. This is the true value of the system and was the primary requirement of the client. As a result, additional development time was allotted to ensure that the requirements were developed to the users specifications and that their needs were being fully addressed. As was the case with the other two functionality sets, a list of functions was developed to identify which functions could be used throughout the system and which were specific to certain requirements.

4.4.3.1 - Creating a bench

Developed with similar functionality to that of the registration and private messaging pages, the creation of a bench uses a form that captures the necessary data on the new stock item. This page omits some information, such as the spray and the line number, as this is either automatically entered by the system (line) or is unnecessary at this stage in the benches life (spray). The information is then added to the database. Figure 38 shows the ability for staff to add a comment to a bench thus fulfilling another of the client’s requirements. This bench is automatically entered into the incubator, or line 0 of the system, until it is ready to be assigned to a line in the glasshouse.
4.4.3.2 - Viewing Benches & Lines

Each benches information needed to be represented in such a way that it was understandable to the user. As was the case with representing the public and private messages, the system prints out the benches information in a HTML table (figure 39).

<table>
<thead>
<tr>
<th>ID</th>
<th>Bench Ref</th>
<th>Herb</th>
<th>Quantity</th>
<th>Spray</th>
<th>Spray Date</th>
<th>Line</th>
<th>Days Old</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>426</td>
<td>074</td>
<td>coriander</td>
<td>300</td>
<td>spray2</td>
<td>28/04/2015</td>
<td>15</td>
<td>&gt; 51</td>
<td>Harvest Bench</td>
</tr>
<tr>
<td>428</td>
<td>074</td>
<td>coriander</td>
<td>300</td>
<td></td>
<td></td>
<td>15</td>
<td>&gt; 51</td>
<td>Harvest Bench</td>
</tr>
<tr>
<td>204</td>
<td>074</td>
<td>thyme</td>
<td>900</td>
<td></td>
<td></td>
<td>10</td>
<td>&gt; 51</td>
<td>Harvest Bench</td>
</tr>
<tr>
<td>248</td>
<td>074</td>
<td>thyme</td>
<td>900</td>
<td></td>
<td></td>
<td>13</td>
<td>&gt; 51</td>
<td>Harvest Bench</td>
</tr>
<tr>
<td>73</td>
<td>074</td>
<td>coriander</td>
<td>900</td>
<td></td>
<td></td>
<td>3</td>
<td>&gt; 51</td>
<td>Harvest Bench</td>
</tr>
<tr>
<td>304</td>
<td>074</td>
<td>thyme</td>
<td>900</td>
<td></td>
<td></td>
<td>16</td>
<td>&gt; 51</td>
<td>Harvest Bench</td>
</tr>
<tr>
<td>180</td>
<td>074</td>
<td>thyme</td>
<td>900</td>
<td></td>
<td></td>
<td>7</td>
<td>&gt; 51</td>
<td>Harvest Bench</td>
</tr>
<tr>
<td>238</td>
<td>074</td>
<td>basil</td>
<td>900</td>
<td></td>
<td></td>
<td>11</td>
<td>&gt; 51</td>
<td>Harvest Bench</td>
</tr>
<tr>
<td>42</td>
<td>075</td>
<td>coriander</td>
<td>900</td>
<td></td>
<td></td>
<td>2</td>
<td>&gt; 51</td>
<td>Harvest Bench</td>
</tr>
</tbody>
</table>

Fig. 39 - Change line functionality and viewing stock by line

```
if(isset($_POST['change_line'])) {
    header('location: change_line.php?bench_id=' . $_POST['bench_id'] . '&from_line=' . $_POST['from_line'] . '&change_line=' . $_POST['change_line']);
}
```

Fig. 40 - Change line code within table

The ‘line’ column of the table contains an input form (figure 39 & 41) which uses similar functionality to that of the group messaging text box. This enables the user to change the line of multiple benches without having to leave their current page. Within the form are two ‘hidden’ variables, ‘bench_id’ and ‘from_line’. Bench_id is used to identify the specific bench being moves and from_line is used to identify the line currently selected to ensure that the system returns the user to the same locations when the header function is called (figure 40).

Visible at the top of figure 39 is a list of numbers from 1 to 21, omitting the number 12. These represent the lines of the glasshouse, with line 12 being inside the dispatch area. Selecting one of these lines filter the data to only display the benches stored on that line (figure 41).
This enables the user to search the glasshouse and to edit multiple benches from a line at once. Being able to query the data in this way satisfies the user requirement of easier searching of the glasshouse.

Representing all of the data at once was a challenge the author faces during this stage of development. As the glasshouse could be holding as many as 600 benches at one time, this information needed to be presented on the screen in such a way that an individual bench could be found without any complications to the user. The temporary solution is a HTML div, with a restricted height of 500px and an overflow that enabled a scroll bar whenever the contents extended beyond this set height (figure 42).

```
<div style="height:500px; overflow:auto">

Fig. 42 - Representing large amounts of stock within a div with a restricted height

4.4.3.3 - Editing the Benches Details

Spraying, spacing and throwing out stock are activities that happen on a daily basis within the organisation and so it was especially important that these maintenance functions were developed into the system. Employing the use of the MySQL queries, these functions update and delete the information in the database to emulate the operations being taken on the physical bench itself. As the types organisation only uses specific spray in their herbs, they could be hard coded into the system along with the duration of time until the spray_date. This means that any employees can update the system without needing to know the specific duration required.

The spacing functionality requests a number from the user as to how many benches the current bench is being spaced on. The functionality currently divided the quantity on the bench by this provided number to quantify the new quantities of the benches. Although not entirely accurate, this is industry specific data could be update with the correct quantities in future development.
Finally, the disposing or deleting of stock, is currently is currently done by simply deleting the record from the database. This will need to be altered in the future to enable the tracking of stock disposal and in order to ensure that the company is not incurring unnecessary expenses as a result of production inefficiencies.

4.4.3.4 - Harvesting

Each bench has a ‘days old’ counter that uses the date the bench was planted to calculate its age. From this the user can derive whether the bench is ready for harvesting or not. The harvesting functionality is available from the stock management page and operates in a similar manner to the functionality that moves the bench from one line to another, in that when a bench is harvested, the system calls a header function to the same page and it updates the ‘used’ column of the database automatically. As multiple benches could be harvested at once, it was important that this operation could be done quickly and without unnecessary complications.
Chapter 5. Evaluation & Conclusion

The objective of this project was to investigate whether a prototype platform could be designed and developed that satisfies the communication and stock management needs of herb growing companies. The process by which this was attempted began by creating a list of requirements to represent the project objective and developing functionality that satisfied these requirements. However, when developing software for a client, the list of requirements is never fully set as the needs of the organisation and the staff within it are constantly changing. It was for this reason that the system was developed in an iterative and agile manner, ensuring that the client could communicate any necessary changes prior to them being developed.

The biggest challenge faced during this project was interacting with a client that possessed little knowledge of software development. This left them naive as to the amount of time and effort required to develop functionality and so the list of requirements had to be restricted to only encapsulate what could be developed in the limited timeframe of the project. Once a requirement had been implemented, it were presented to the client for approval. This ensured that the needs of the organisation were being met as each new functionality set was added. Although this client interaction was beneficial to the end result, it was the cause of many additional complications. Despite these complications, the finished system addresses the needs of the herb production industry, and has been developed in such a way that additional functionality can be added when necessary. This chapter aims to address some of the successful aspects of the system along potential improvements that could be addressed in the future.

5.1 - System Efficiency

There were several tests conducted on the prototype which will be discussed here. These tests were run by online technologies to inquire as to the efficiency and simplicity of the code. The metric by which efficiency was be measured was the load time of the site. The table below presents the data from 3 different speed testing websites, GTmetrix, WebPageTest and pingdom. The system was tested on each of these websites at five different time intervals throughout the 4th of April, 2015. The average of all of these times was then calculated to
develop a single figure that represented the efficiency of the system. This testing was conducted in this manner to remove any time or technology specific advantages and to make for more accurate results.

<table>
<thead>
<tr>
<th></th>
<th>Test 1 (6:00)</th>
<th>Test 2 (9:00)</th>
<th>Test 3 (12:00)</th>
<th>Test 4 (15:00)</th>
<th>Test 5 (18:00)</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTmetrix 18</td>
<td>0.46</td>
<td>0.49</td>
<td>0.51</td>
<td>0.61</td>
<td>0.41</td>
<td>0.496</td>
</tr>
<tr>
<td>WebPageTest 19</td>
<td>0.546</td>
<td>0.618</td>
<td>0.614</td>
<td>0.627</td>
<td>0.591</td>
<td>0.599</td>
</tr>
<tr>
<td>pingdom 20</td>
<td>0.47</td>
<td>0.435</td>
<td>0.295</td>
<td>0.402</td>
<td>0.655</td>
<td>0.451</td>
</tr>
<tr>
<td>Mean</td>
<td>0.492</td>
<td>0.514</td>
<td>0.473</td>
<td>0.546</td>
<td>0.552</td>
<td>0.515</td>
</tr>
</tbody>
</table>

Fig. 43 - Efficiency testing results and average load time (Red)

As is visible in figure 43, there can be a substantial difference in the results of each of the websites and so it was important that an average of results was calculated from multiple sources and multiple times. From these multiple sources the author arrived at a final results of 0.515 seconds. According to pingdom.com this makes the system faster than 97% of the other websites tested on their service (figure 44). This is an significant result as load time is an important aspect when it come to the adoption of the system. According to a study conducted by Akamai and Gomez.com, almost 50% of users expect a web page to load within 2 seconds or less and the system developed meets this criteria.

Fig. 44 - pingdom places developed system within top 3%

---

17 GTmetrix - http://gtmetrix.com/
18 WebPageTest - http://www.webpagetest.org/
19 PingDom - https://www.pingdom.com/
20 How loading time affects your bottom line - https://blog.kissmetrics.com/loading-time/
5.2 - Further Development

The 3-Tier architecture enables new functionality to be added to the system with fewer complications. Although the functionality developed does currently meet the users requirements, a system such as this is never truly complete as new functions can be added as the organisation develops new needs. It is important when developing the future vision of the system, that it remains relevant to the user.

As this system is addressing a real business problem, it also presents a real business opportunity. The main aspects of the system that could be developed into the further can be broken into four different categories:

5.2.1 - Administrative Functionality

Email all users

Currently the administrative functionality within the system amounts to the ability to email all of the users in the system from the [http://niallhunt.com/mail.php](http://niallhunt.com/mail.php) page (figure 45).

However the system should enable the administrator to add and remove employees as they join and leave the organisation. In addition to this it would be beneficial to the management of the organisation if they were presented with information regarding the quantity of herbs available to sell, what was being harvested and the quantity of waste being produced.

The future vision of this system is that it would expand beyond an individual organisation to being able to accommodate SME’s in other crop and plant production industries. This system would be customisable to the user so that they could add and remove the details which are currently fixed within the system such as number of lines and the species of plants being produced.
5.2.2 - Industry Specific Data
Currently the system uses placeholders to demonstrate the spraying functionality of the system. This industry specific data will need to be added in to ensure that the correct information is captured by the system. Building on this industry specific data, the vision of the system is that it will encapsulate much of the growth pattern tracking functionality as discussed in the domain analysis of chapter 2. Integrating a simulation of plant’s growth rates based on the specifics of the environment would enable the system to notify the user as to when the herbs should be produced, maintained and harvested.

5.2.3 - System Optimisation
As discussed during the database design section of chapter 3, the tables of the database were not normalised to third normal form out of convenience to the author during implementation. However, without the time constraints of the final year project, the database could be optimised in order to reduce data redundancy. Figure 46 shows the database in third normal form. Although currently having no effect on the functionality of the system, as the organization grows, their data needs will increase and so an optimised database is more favourable.
As the system was developed in a 3-Tier architecture, much of the visual components are reused on every page. However, because of the way the system is currently developed, the same functionality is being called multiple times as the user navigates throughout the system. Caching is the process by which the system stores files that are being regularly used so that they can be recalled without having to access them from the server. The introduction of this functionality would greatly improve the load time of the system and reduce the bandwidth usage on mobile devices.

5.2.4 - Improving Usability

Improvements could be made to the communication and stock management functionality with the implementation of new technologies such as Ajax. Ajax is a subset of JavaScript that asynchronously queries the database for new entries and updates the display to represent this new information. This technology is used heavily in social networks to add new posts to the users' news feed. Implementing this technology would improve the usability of the system.
by notifying the user as to when a new message has been posted or when stock had been sprayed. Currently this functionality does not exist within the system and would greatly improve the likelihood of its adoption.

Currently the system is accessible to mobile devices via their web browser (figure 47) but it is not optimised for mobile use. Much of the text is too small and the hyperlinks are too close together. However, in the future, a mobile specific Tier-one or a mobile application could be developed. This would greatly increase the reach of the product and would improved usability on mobile devices. As many of the employees will be accessing the website on a mobile device, a mobile optimisation would be necessary if the system was to be a viable option into the future.

Finally, visually representing the stock in the system was a challenge the author faced throughout the development process. The system requires a more visual representation of the glasshouse such as a floor plan or through querying the database based on other criteria than just line number.

5.3 - Client Adoption

As the system was designed to be a proof-of-concept prototype, its adoption was never a practical option for the case organisation. However, on Thursday the 2nd of April, 2015, the author presented the final system to the client to obtain their opinion as to whether they would consider adopting a future version of it into their operations and whether they felt their needs had been addressed by the functionality developed. From this meeting it was discovered that
although more work is needed, there was value within the system and it is addressing many of the needs specified. The positives that were taken from this meeting came from the owner of the case organisation who stated “I can see how this could become something or real value” and from a member of the glasshouse staff who stated “I’d say using it might require a little more work but I can see how it could be useful to us”. However, the office staff found that the functionality was not addressing their particular needs and that “it still needs work”. Understand which of the stakeholder feel that their needs have been addressed is an important aspect to the future development of the system. Know the weaknesses of the system is important because it help identify what functionality needs to be addressed first.

5.4 - Final Remarks

This report has outlined the successful implementation of an online platform that enables users to communicate publicly and privately within an organisation and to track the creation, maintenance and harvesting of herbs are they pass through the production cycle. This is not to say that the proposed solution is complete; it is simply a successful prototype.

This report began by providing a background on the herb production cycle to give the reader an understanding of the terminology used within the industry and the processes that were being addressed by the system. A domain analysis was then conducted to investigate the alternatives available to the organisation. These alternatives were found to be inadequate but they provided the author with inspiration in developing the system’s requirements and how to address certain problems. Additional requirements were developed from sources including the authors own experience and the insights of the case organisation.

In undertaking this project, a more rounded knowledge of not just computer science and software development was attained but also how to address business needs with a technical solution. Having the opportunity to interact with a client and develop a solution with their insight was invaluable to the author. Developing for a client requires not just a knowledge of the technical and business aspects of a system, but also how these two aspects can affect one another, and how to manage this aspect of the project.
It is likely that the author will continue developing functionality for this system in the future. If not to develop it into a business, than to provide the client with a solution that they would feel comfortable adopting into their operations. Considering the value that has already been developed within the system, this is by no means a goal that would be beyond achieving.
Bibliography

   Accessed at: 01-04-2015

   Accessed at: 01-04-2015

   Accessed at: 28-03-2015


   Accessed at: 03-04-2015

   Accessed at: 04-04-2014

   Accessed at: 04-04-2014


10. InFlow Inventory Management. Available at: http://www.inflowinventory.com/

11. What the lamp stack is and why every blogger and website operator should know about it. Available at:
    Accessed at: 25-03-2015


