

# Determining Optimal Water Usage for Small Agriculture Sites with Finite Water Resources

By Stephen Jackson

Student ID: 1729735

Supervisor: Dr Morteza Azad



UNIVERSITY OF  
BIRMINGHAM

MSc Project Submitted in conformity with the requirements for the degree of MSc  
Computer Science

School of Computer Science

University of Birmingham

## Abstract

Chesterfield Inspire Community Garden is a community garden that aims to grow a range of fruit and vegetables for local school and charities to encourage healthy eating in Chesterfield's most deprived areas. The problem that they have is their site does not have access to a water supply and can only use the water that it stored on site. While this water is replenished every 2 months the garden has the problem of working out how they allocate their available water to maximise crop growth within that period.

This report details how this problem can be solved using Linear Programming and optimization techniques and gives details of the project to build a software solution that would use these techniques to solve Inspire Community Gardens problem but can also solve the problem for other sites in similar circumstances.

The report starts by describing the motivation for the project and how a solution may be useful to a wider range of sites. It then researches the factors that must be taken into account in plant water requirements, details how a solution may be formulated and then documents the development of an application to implement that solution.

The report concludes with an evaluation of the project which looks at what went well and what improvements could be made in the way the project was undertaken. While also detailing what further features could be added to the software solution if more time and resources had been available to the project.

## Acknowledgements

I would like to thank my supervisor Dr Morteza Azad for his help and support in this project, Dr. Nabi Omidvar for his help in developing an objective function for the problem and the trustees of Chesterfield Inspire Community Garden for taking the time to look at the prototypes of the application and give such useful feedback.

## Contents

1. Introduction.....	5
1.1 Review of Report.....	5
2. Background.....	7
2.1 Review of Literature.....	7
2.2 Factors Affecting Plant Water Requirements.....	8
2.3 Optimization.....	12
3. Development of a Solution.....	14
3.1 Objective Function.....	17
4. Review of Similar Software Applications.....	20
5. Project Management.....	23
6. Application Development.....	27
6.1 Application Specification.....	27
6.2 Application Design & Implementation.....	31
6.3 Application Validation & Testing.....	40
7. Evaluation.....	45
7.1 Results of Project.....	45
7.2 Evaluation of Process.....	49
7.3 Evaluation of Application Design Choices.....	50
7.4 Lessons Learned.....	51
7.5 Further Developments.....	52
8. Conclusion.....	53
9. Appendix 1 – How to Run Application.....	54
10. Appendix 2 – Bibliography & References.....	55
11. Appendix 3 – User Interface Design.....	57
12. Appendix 4 – Database Entity Relationship Diagram.....	58

## 1. Introduction

In deciding what project, I wanted to undertake as part of my Msc in Computer Science I looked at problems that I was encountering in my hobbies and activities to see if I could come up with a software based solution for them.

An activity that I am involved with is volunteering at the Chesterfield Inspire Community Garden (<http://www.inspirecommunitygarden.org.uk/>). This is a community charity that is aiming to turn an abandoned wasteland into a fruit and vegetable garden with the overall objective of providing fresh produce to local schools and other community groups as well as promoting the health benefits of gardening.

The project has many committed volunteers and through donations and grants can source equipment and seeds, however its greatest challenge in achieving its objective of producing high crop yields is water. Due to the site being situated on a former wasteland it does not have access to mains water supply or any other source of water. This means that the site must rely on the water that it can store in water containers on the site. These containers are filled up either by rain water or at set periods by the local fire brigade. To ensure that the garden is successful the community garden wants to see the plants in the garden get as close to the amounts of water they require while making sure that water available is not depleted before the water is replenished.

Water management and optimisation systems have been in use by big agriculture for a number year and will be examined in later chapters of this report. This problem however will relate to small agriculture sites and the challenges that they face in allocating their water resources

While this application is being developed to provide a solution to a specific problem. It was the intention that the application being designed in such a way that it could have uses beyond this site-specific problem and be used by other small land holdings, allotments or farms that need to optimize their water usage. This could be for many reasons not just the one identified in the Inspire Community Garden. Other reasons could include sites that want to limit their water extraction for environmental reasons or even for sites that have access to a mains water supply, but due to the water being metered the site can only afford a set amount for water each month, they may need a system that would help them make the most of the water they can afford. Therefore, the system has been designed with these wider users in mind.

### 1.1 Overview of Report

The report follows on from this introduction with a background section that will review relevant research to identify the factors that must be considered when defining water requirements of sites. Following this the report will then seek to reduce this problem to a

mathematical formula and identify relevant techniques that can be used to solve the problem.

Once the report had identified the solution it then goes on to document the development of a software application that can implement the solution in a real-world context. It will do this by first reviewing what current software solutions are available and seeing if any lessons can be learned from them.

After reviewing similar software the report outlines the project management methodology used in the development of the application and the reasoning behind that approach.

Following this the report will then outline the development of the software application through first identifying the requirements of the system and then documenting the design decisions made by the developer to meet those requirements. This chapter will also detail how the system was validated and tested and how the development of the system was affected by the results of the tests.

The report concludes with an evaluation of the project and the application. This section will seek to look back over the project see what worked well and what did not. It will see if the software application that was produced met the aims of the project and evaluate some of the design decisions made along the way. This section will conclude with identifying how the application could be taken forward and what additional features could be added in future releases of the application.

## 2 Background

Having identified the problem the next stage was to develop the necessary background knowledge needed to develop a solution. The first section of this chapter will review the literature used in the research of this problem. The next part will give the conclusions from that research on what variables impact a plants water requirements and how an optimization problem can be defined.

### 2.1 Review of Literature

The research which took place for this project can be grouped into 4 broad subject areas. The use of technology in agriculture, factors affecting a plants water requirement, optimization methods and theory and how those methods are used in water management systems.

In researching the topic of technology in agriculture it was found many articles had been written by the technology press on this subject. Examples were The Future of Agriculture (Carr 2016) where he describes how technologies from cloud computing to robotics are changing agriculture. The article Dig Gardening? Plant Some Connected Tech this Spring by (Glaser 2016) looks at similar themes for home gardening. The conclusion from this part of the research was that technology is having a greater role in how people grow food and manage their gardens.

The second part of the projects research focussed on factors that impacted on plants water requirements since knowledge of these factors would be needed to develop a solution to the problem. Research found that there were numerous information sources and books giving advice on amounts of water to give to plants. This advice ranged from very vague statement such as those contained within The Farmers Almanac website(, The Farmers Almanac 2017) “Give lots of water in first 2 weeks”, to reports such as Water Requirements of Major Crops for Different Agro-Climatic Zones of Balochistan(Ashraf, Majeed 2006) which looked in extreme detail at water requirements for a small number of plants in a specific location. The most useful sources came from government organisations such as the United Nations Food and Agriculture Organisation. Reports they produced such as Irrigation Water Management(Brouwer, HeiBloum 1986) gave a good overview of factors that impact water usage of plants. Another text of note was The Complete Know and Grow (Bleasdale, Slater 1991). This text gave data on plant water requirements that was collated from results of other studies as well as their own experimentation. The conclusions reached from this research are developed further later in this chapter.

Having identified that the problem this project was seeking to solve was one of optimization, research was undertaken on optimization theory and methods. Since optimization has been an area of academic study some of the most useful and concise sources were produced by universities to go along with under graduate courses such as Maths 407 – Linear Optimization(Burke 2016). With further indepth examples provided by

text such as An Introduction to Optimization (Chong, Zak 2001). This part of the research gave a knowledge base to make judgements on how the problem should be solved.

The last section of the research looked for examples where these methods had been used in water management systems to see if those studies could be translated to this project. Several papers were found that detailed how particular algorithms had been used in water management systems. A report titled Management of Water Resources using Improve Genetic Algorithms (Chen 1997) looked at how genetic algorithms are used to develop water management for large scale water management systems. A further example was an (Wang, Yang et al. 2015) that looked at Water Management optimization in a water catchment area in China. Through research articles like these it showed that optimization algorithms had been used in water management, however the research could not find examples where they had been used in the context of this project.

## 2.2 Factors Affecting Plant Water Requirements

This section summarises the research on factors that impact plant water requirements.

### *Plant Types*

The primary variable that affects water requirements is the type of plant being grown. Each plant has its own total water requirement that it needs throughout its life cycle, with plants which are bigger or have more leaves requiring more water than smaller plants. (Brouwer, HeiBloum 1986) Research has shown that all plants have a desired or optimal amount of water that they need. If a plant has this level of water they will be able to produce the maximum amount of crops dependent on other factors not related to water requirements. It is however important not to go over this desired level since after that the plant will no longer take in the water. This would waste resources but also giving a water allocation over this desired amount could harm the plant through over watering which is a leading cause of plant death in gardens (Glaser 2016).

*“The belief that all vegetables will always benefit from watering, and the more water the better, is not so. It is very easy to give too much water too often.”*  
(Bleasdale, Slater 1991) p86

While all crops have this desired amount, they also have a basic amount of water they need to survive. If the plants get water that is less than this basic amount they will fail to produce crops and if they continue to not get the basic amount of water required they will die. (Bleasdale, Slater 1991).

This research has shown that for any water management system to be effective it must consider the type of plant when allocating water and try to give each plant an amount of water that is as close to its desired amount as possible while ensuring all plants are given their basic amount. In this it must also factor in the number of that kind of plant each plot contains to enable it to make a reliable allocation of water resources.



### *Plant Growth Stage*

If each plant had a set amount of water needed then it would be a simple model of adding up the water amounts for each plant and seeing if there was enough water for each plant. This is not the case, research has shown that all plants go through distinct growth stages as they develop with each plants water requirement changing as it moves from one stage to another.

Studies have been done for individual plant types to define their growth stages for example fig 8 of Chapter 2 (Brouwer, HeiBloum 1986) identifies 7 different aspects in the growth of wheat but still separates them into 4 distinct growth phases called initial stage, crop development, mid season and late season . The growth of most cropping plants can be broken down into 4 distinct stages and although they are named differently in individual studies these stages can be describe with the following broad characteristics.

The first stage of the cycle is the germination stage where the plant will develop from the initial seed and begin to form roots and leaves. The second stage, sometimes called the vegetative stage is where the plant establishes itself and grows big enough to support crops. The 3rd stage the flowering or crop development stage will see the plant develop its crops. The growth cycle is completed by the final stages called the harvest stage where the crops are fully matured and are ready to harvest. (Bleasdale, Slater 1991)

As a plant moves through these stages its water requirements will change to reflect its needs for that stage. Fruit plants for example will need far more water when developing crops then they do when they are in their vegetative stage due to the high amounts of water that are used in the creation of each fruit.(Allen, Periera et al. 1998) A water management system must be aware of the growth stage a plant is in and adjust water accordingly.

For optimization problems such as ours a factor to take account of is research that shows that to maximise crop production it is more important for plants to get their desired amount at certain stages of their growth. The most important stages being stage 1, since that is when the plant is more vulnerable and most likely to die if its needs are not met and in the 3rd since that is when the plant produces the crops.(Brouwer, HeiBloum 1986) A water management system using a finite water source would seek to prioritise water to plants in those particular stages.

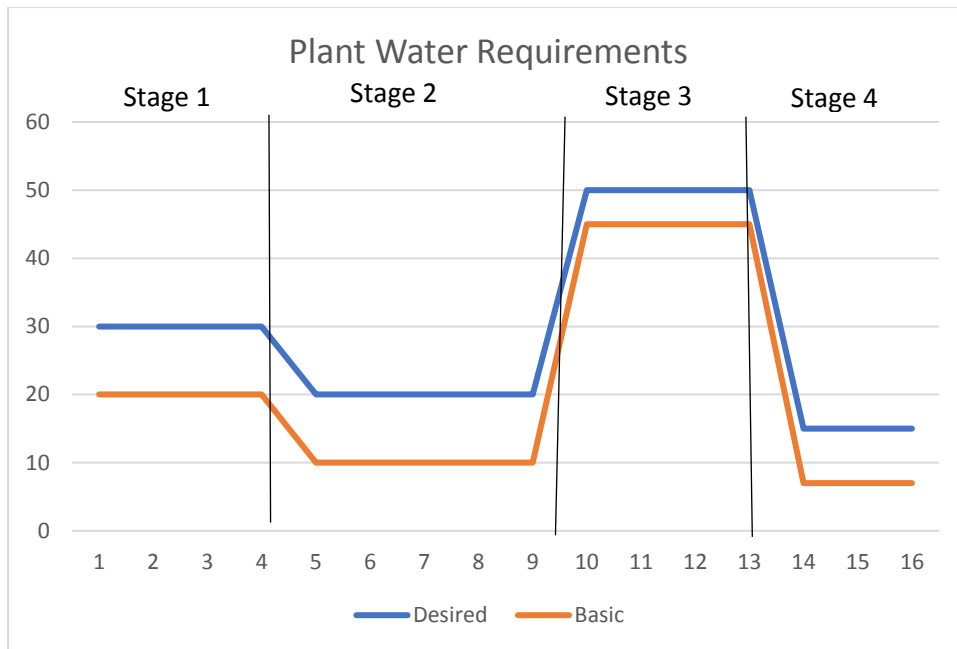


Figure 1 Example of how Desired and Basic Water Requirement change through growth stages of a plant.

### *Drainage*

One of the main impacts on a plant's water requirement is the soil that it is planted in. If a plant is planted in a soil which is sandier in nature then any watering done to that plant will drain away quicker leaving less for the plant to use. The other extreme to this if the plant is in clay soil which holds water. The danger is that the ground will hold more water than the plant needs leading to the problem of over watering identified earlier in this chapter.

In p90 (Bleasdale, Slater 1991) it shows that if 30cm depth of different soils is given 0.3ltrs of water for every square meter of soils a loam soil which is between sandy and clay soil in consistency will hold approximately  $5\text{cm}^3$  of water compared to  $6.5\text{cm}^3$  in clay soils and  $4.5\text{cm}^3$  in sandy soils. This shows that there is a 40% range within the different types of soils in the water that they retain.

A further factor to consider in drainage is the environment of the plant. It is common in small sites to plant crops in plots that are built higher than ground level called raised beds. Being higher than ground level these plots have a higher water requirement than plants at ground level.(Bleasdale, Slater 1991)

The results of these experiments show that when managing water soil type and drainage must be considered when allocating water.

### *Evaporation and Transpiration*

A term that is used in irrigation planning and environmental study is evapotranspiration which means the sum of the water lost from the land through direct evaporation

(Evaporation) and from the plants expelling water to regulate their temperature (Transpiration).

The rate of transpiration is different for each plant with plants with bigger leaves having a greater rate of transpiration than those plants with smaller leaves and flowers. (Brouwer, HeiBloum 1986)

The impact of evapotranspiration on water requirements is looked at in detail in (Allen, Periera et al. 1998). In the paper, they propose the Penman-Monteith Equation to calculate water needs based on temperature and humidity. This equation uses data collected for at least a month on humidity, daily high and low temperature to calculate the rate of evapotranspiration from the site. Using this the farmer can work out how much extra water to give plants based on the output of the equation.

While this method gives an accurate measurement for it to be effective it needs monitoring data collected over a set time. This project aims to produce a solution to be aimed at small plots that would not be able to afford the sensors necessary to collect this data or have the expertise on how to use it.

Therefore the more general approach taken in (Brouwer, HeiBloum 1986) was looked at. In this approach, they ran an experiment to see impact on water usage on a grass plant. They used grass because it was representative of how a range of plants water requirements changed with temperature.

The experiment showed that in a range of climatic regions the plants water need would increase by roughly 1mm per day for every 5 degrees centigrade that the temperature rose. While not as accurate as the equation used by (Allen, Periera et al. 1998) it does show a trend in correlation between rise of temperature and a plants water requirement that can be estimated in any water management system.

Climatic zone	Mean daily temperature		
	Low	medium	High
	(less than 15°C)	(15-25°C)	(more than 25°C)
Desert/arid	4-6	7-8	9-10
Semi arid	4-5	6-7	8-9
Sub-humid	3-4	5-6	7-8
Humid	1-2	3-4	5-6

*Reproduction of Table 2 from Chapter 2 (Brouwer, HeiBloum 1986)*

As with soil type the environment the plants are grown in impacts values associated with temperature and humidity. A way to grow a range of plants that require warmer climates is the construction of a polytunnel. A polytunnel is the construction of a frame covered with plastic sheeting that covers a plot. Water requirements of plants within the poly tunnel can be increased by anything up to 15% due to the higher heat and humidity,(Bleasdale, Slater 1991) which is significant enough that it must be considered by any water management system.

### *Rainfall*

Rainfall will have 2 impacts on the water requirements of plants and on any solution, that is found for the projects problem. The first will be that if the site is able to collect rain water using on site water storage, rainfall would increase the overall water available to the remaining part of the optimization period, therefore any solution must give the ability to recalculate an optimal water regime at any time throughout the optimization period.

The second impact is that rainfall may satisfy some if not all of the plant's water requirement for the day without the need to use any of the water stored on site. A solution to the problem must take this into account and if rain is forecast reduce water requirements accordingly.

“If the plant can obtain sufficient water from the soil to sustain water loss through the leaves at a maximum rate, then watering is unlikely to benefit the growing plant”.(Bleasdale, Slater 1991) p88

In assessing the impact of rainfall a solution must take into account plots that do not get rainfall. The last section described polytunnels, since those sites are covered they will not benefit from any rainfall that occurs and so a premium must be added to those sites to compensate for this.

## 2.3 Optimization

As stated in the introduction this project seeks to use a resource, in this case water, in the most optimal way that gives the greatest return. This makes this project problem an optimization problem. This section gives background information on what an optimization problem is and how it is defined.

Optimization is described by (Boyd, Vandenberghe 2009) as follows “Optimization theory and methods deal with selecting the best alternative in the sense of the given objective function.”

All optimization problems will have some sort of an objective function. An objective function is a function of variables and coefficients that we look to either minimize or maximize.

The standard mathematical form for an optimization problem is described in the Introduction to Convex Optimization by Boyd & Vandenberghe 2009 and is shown below.

“A mathematical optimization problem, or just optimization problem, has the form

minimize  $f_0(x)$   
subject to  $f_i(x) \leq b_i, i = 1, \dots, m$ .

Here the vector  $x = (x_1, \dots, x_n)$  is the optimization variable of the problem, the function  $f_0 : \mathbb{R}^n \rightarrow \mathbb{R}$  is the objective function, the functions  $f_i : \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, \dots, m$ , are the (inequality) constraint functions, and the constants  $b_1, \dots, b_m$  are the limits, or bounds, for the constraints. A vector  $x^*$  is called optimal, or a solution of the problem (1.1), if it has the smallest objective value among all vectors that satisfy the constraints: for any  $z$  with  $f_1(z) \leq b_1, \dots, f_m(z) \leq b_m$ , we have  $f_0(z) \geq f_0(x^*)$ .”

Examples of optimization problems are wide but can include problems where a financial company seek to maximize their profit through investing in a certain number of funds. Another example would be an electronic factory looking to find the best size to build circuit boards to minimize the cost of each board.(Boyd, Vandenberghe 2009)

The field of optimization is broad with many ways that can be used to solve optimization problems. In deciding which of these methods would be best for this projects problem we must develop our optimization problems model and objective function. To do this the next chapter takes the variables set out in this chapter and develops them into an optimization problem format as specified above.

### 3 Development of a Solution

Having now researched and identified the key variables that must be taken into account when estimating water requirements, we can now look to how we can construct a formula to solve this problem that can then be implemented programmatically.

The research detailed in the last chapter gives us the key variables that must be considered when developing a water management system. It also showed that a range of external factors impact on a plants water requirement that any such system must consider.

This chapter will now go through the process of creating a model and optimization function for the projects problem that will give a robust solution. From our problem description in chapter 1 and the conclusions of research described in chapter 2 we can state 3 key requirements needed from any solution. It must not use any more water than is available on the site but be able to deal with changes in that amount due to rainfall. It must be able to adjust a plants desired and basic amounts based on external factors such as weather, soil type environment. It must try to get the amount allocated to each plant to be as close to the desired amount as possible. In addition to this if it is not possible to give all plants their desired amount the solution must prioritises those plots where the plants are in their more critical growth phases.

To develop our model, we must first construct matrices showing these upper and lower bounds that will make up the feasible region for the solution. In these matrices, the rows would represent each plot within the garden and the columns would represent each day within the period that we are looking to allocate the water available in.

The first matrix to be constructed would be a matrix to show the desired water requirement for the plants planted in each plot based on where they are within their growth cycle on each particular day of the period of optimization

Next, we would need to identify the numbers of plants that we are calculating water requirements for to give us a total amount of water requirement for each plot. This would be calculated by using the recommended number of plants that should be planted per square meters and multiplying that by the size of the plot. How this information was sourced is discussed further in Chapter 6.2 Project Design

Matrix NP = NP = (np<sub>ij</sub>) = ij

j = plants per meter for Plots Plant type

i = size of plot

Again using data we create a matrix showing the desired water required (DWR) and basic water required(BWR) based on the plots plant type and where they are within their growth cycle. This would be calculated through ascertaining how many days had passed from when the plants were planted to the period that is being covered by this optimization problem.

Having established matrices showing desired and basic water needs we must then create matrices that reflect the variables that impact on those values giving us a final matrix that gives the bounds of the problem.

To construct this final matrix we would create a matrix of values representing the soil type of each plot, the environment of each plot and also matrices showing weather variables represented as values that can be input to a calculation.

Research detailed in the last chapter showed that there was a 40% difference in the amount of water retained from clay soil to sandy soils. Those results can be used to give values to build up a matrix for soil type.

S = soil

- Clay = 0.8
- Part Clay = 0.9
- Normal = 1.0
- part Sandy = 1.1
- Sandy = 1.2

The same methodology can be used to create a matrix to give values that represent the environment variables. From research detailed in the last section of this report we know how different environments can give an increased demand for water by plants. This lets us give values to the environment matrix using the research undertaken. Research showed a significant higher amount of water is needed in a polytunnel due to high temperature and that it is not able to benefit from rainfall. Research also showed that there is a higher need for water in raised beds due to their drainage. From that research we can give values to construct a matrices of environment values.

E = Environment

- Polytunnel = 1.25
- Raised bed = 1.05
- normal bed = 1.0

Following the creation of these matrices we must then create a matrix to give values that can be used in a calculation to express how temperature impacts a plant's water requirements. The last chapter detailed an experiment which showed a gradual rise in a plants water requirements for every 5 degrees Celsius increase in temperature. We can approximate that conclusion in our model by adding a small percentage increase in water requirement for each raise in 5<sup>0</sup>C beyond normal temperatures. In creating a temperature matrix, we must accept that temperature can only be estimated any degree of accurately up to 10 days beyond the current date. In constructing a matrix that involves any weather element factor we must accept that this array may be sparse with values only being able to be set for a maximum of the first 10 days of the period if the optimization period starts from the current date.

T= Average Daytime Temperature

- < 20°C = 1.0
- 20°C > & < 25°C = 1.05

- $25^{\circ}\text{C} > \& < 30^{\circ}\text{C} = 1.10$
- $> 30^{\circ}\text{C} = 1.15$

The final variable we must create is a matrix for the expected rainfall within the optimization period. Again like the temperature matrix this matrix may be sparse with values only being available for the first 10 days of the optimization period. Since rainfall will go to meet the water requirements of each plant that if a rainfall value is above zero it will always lower the water requirement calculation we are doing since less water will need to be used from a sites water storage.

$R$  = rainfall value forecast per day

Having now created our base data matrix we then used those data matrix to create matrices for our objective function that give the desired water requirement and basic water requirement for each plot on a day by day basis using the following formulas.

Desired Matrix  $D = (d_{ij}) = DWR_{ij} * NP_{ij} * S_{ij} * E_{ij} * T_{ij} - R_{ij}$

Basic Matrix  $B = (b_{ij}) = BWR_{ij} * NP_{ij} * S_{ij} * E_{ij} * T_{ij} - R_{ij}$

These 2 matrices will allow us to give the objective function lower and upper bound constraints and a desired value for the decision value to aim for. To ensure that the optimization function returns results that will give most benefit we must add weightings to the objective function so that it prioritizes water allocation to the plots where having an amount of water closer to the desired amount is more beneficial.

To add these weightings to the objective function we create a matrix of priority values which is made up of 2 data matrices User Priority and Stage Priority.

The user priority will give values that will be based on a priority rating given by the user of the system to show the importance they place on that plot producing the maximum amount of crops possible.

UserPriority  $UP =$

- High = 3
- Normal = 2
- Low = 1

The base matrix will be used to construct a priority matrix showing priority values based on which of the growth stages the plants in each plot are on, at each day of the optimization period, calculated by comparing the dates of the optimization period and the date a plot was planted and the length of each of its plants growth stages.

Stage Priority  $SP =$

- Stage 1 = 4



- Stage 2 = 2
- Stage 3 = 3
- Stage 4 = 1

From these 2 matrices a priority matrix can be constructed to give weightings for our objective function using the following formulae.

Priority Matrix  $P = (p_{ij}) = UP_{ij} + SP$

### 3.1 Objective Function

Having now developed a model through matrices to create bounds on our optimization problem and weightings to give periodisation the next stage is to develop an objective function that will take these into account while meeting the requirements detailed earlier.

To meet these aims an appropriate objective function would create a result matrix  $X$ .

To give a starting point matrix  $X$  would be given values using the formula

$$X = (x_{ij}) = (D_{ij} - B_{ij}) \setminus 2 + B_{ij}$$

Using this starting point an objective function that would meet the set out requirements could be:

$$\text{Minimize } Z = \sum_{i=1}^n (D - X) \cdot P$$

Subject to:

$$\sum_{i=1}^n X < \text{Water Available On Site}$$

$$X_i > B_i$$

$$X_i < D_i$$

In this objective function we look to minimize the sum of values representing the difference between the desired amounts and the decision variable multiplied by the priority weightings. This gives a function that seeks to minimize the difference between the matrix of the decision variables ( $X$ ) and desired amounts ( $D$ ) while giving a function that makes it more advantageous to have a lower difference for elements which have a higher priority value.

The limits of this function is that if not enough water was available to meet all basic water needs, then due to the constraints of this objective function it would not be able to find a solution in that scenario.

To solve this problem the objective function can be further refined to move the matrix of values showing basic requirements per plot from a constraint to form part of the objective function. This will be done through creating a further variable Below Basic (BB) which would be a non negative value representing the difference between the basic level and the proposed optimal matrix.

$$\text{Below Basic(BB)} = \sum_{i=1}^n B - X$$

Through adding this variable to our objective function we no longer make the basic water requirement a hard constraint and allow the objective function to find a solution that could go below basic levels. Instead this objective function gives a penalty whenever the decision variable goes below the basic required amount to encourage the objective function to avoid this if possible

$$\text{Minimize } Z = \sum_{i=1}^n (D - X) \cdot P + \text{BB}$$

Subject to:

$$\sum_{i=1}^n X < \text{Water Available On Site}$$

$$X_i < D_i$$

Having developed an objective function and identified the constraints we can now see more clearly what type of optimization problem we are constructing and can decide on appropriate techniques to solve it.

The objective function created will mean that the feasible region for the solution will be a subset of  $\mathbb{R}^n$  and the objective function is a function from  $\mathbb{R}^n$  to  $\mathbb{R}$ . This will mean that the problem can be defined as a convex optimization problem. This is a common type of optimization problem and consequently there are a range of methods to solve them.

One of the most common approaches used is called Linear Programming. A Linear Programming Problem is an optimization problem where the objective function is a linear function, that has the form  $c_1x_1 + c_2x_2 + \dots + c_nx_n$  for some  $C_i \in \mathbb{R}_i = 1 \dots n$  and the feasible region is the set of solutions to a finite number of inequality and equality constraints. (Burke 2016).

Since our objective function meets this criteria this approach would be suitable. Using Linear Programming has the advantage of being a widely used technique with multiple solver and algorithms created to implement it. Some of the most common algorithms to solve linear programming problems are:

**The Simplex Method:** Developed by George Dantzing in 1947, the Simplex method works in two phases. In the first phase it uses an auxiliary method to find a feasible solution for the first point. In the second phase it uses the “slack” version of the linear problem to move from corner to corner along the edges of the feasible zone. When it reaches a local maximum it returns this as the optimal. (Burke 2016).

A development of this algorithm is the Dual Simplex Algorithm. This algorithm starts with an optimal solution for a point on the vector and then looks for a solution that is feasible while keeping the optimality.

Another common set of algorithms and methods for solving optimization problem is known as the Interior Point Methods. While the Simplex methods look at corner points to find the

optimal point of the feasible zone the interior point methods look within the feasible zone. In doing this it will work to complementarity while using primal and dual feasibility to keep solution within bounds. (Ye 2015)

These algorithms are used by solvers which are application that allow users to input data and run the algorithms. To be able to use these algorithms effectively a relevant solver would need to be chosen.

This would be done as part of wider application that would look to implement the solution that has been defined in this chapter. The next chapters detail how that application was developed and implemented.

## 4. Review of Similar Applications

In developing an application to implement the optimization model described in the last chapter, a good first step would be to review similar applications to the one we are looking to build to see if they have functions that we would like to incorporate within our own application.

This section reviews the software packages aimed at supporting irrigation or gardening objectives since they have aims most in line with what we are seeking to achieve with this project. In reviewing existing software this section has grouped them into 4 broad groups of applications by their aims or targeted audience.

### Irrigation System Management Software

The agricultural industry is increasingly using technology to manage their irrigation systems. Irrigation systems used to be controlled by sets of timers and switches it is now more common for them to be controlled by software packages that take in real time data input from sensors and control the supply of water to plants in response.(Carr 2016)

Research for this project showed that a number of software packages are available to design and manage irrigation systems for big farms and smaller plots. The example of Hunters Irrigation Management & Monitoring Software(2017), shows typical features that are available from these software systems. These systems have the aim of giving the ability to control automated irrigation systems that are networked together. They use advanced mapping software that gives the user feedback of irrigation operations and once linked with sensors the system can adjust flow rate of sprinklers dependent on data it receives from those sensors. These applications have the ability to process large amounts of data including factors such as weather, soil type and plant type.

While applications of this type are advanced they would not be able to help with this project problem since none were found that had the function to optimize a finite amount of water for a defined time period. Software packages like these are aimed at sites that already have an irrigation system installed. Since Inspire Community Garden Site and other sites this project's application is aimed at do not have irrigation systems the usefulness of these applications is limited. In developing our application, we can take lessons from these systems in how they present data. In the examples that were tested, a high priority seemed to be placed in displaying the information in a graphical way. In many of those software packages they displayed the data either using mapping or using graphs and charts. The lesson that was taken from this is that displaying results in a graphical format makes for a better user experience.

## Lifestyle Gardening Apps

With gardening continuing to be a popular hobby many gardening applications have been created to help people manage their own individual gardens and plots. many of these applications are mobile apps that people can use on mobile phones and tablets. Testing these applications revealed they focused on provided functions to let users plan their gardens with advice given on when and where to plant particular crops and when they can be harvested. An example of this kind of app is Suttons Fruit and Veg Planting Guide(2017)

While these apps are aimed more at similar users which this project is looking to support, none of them offer a function to allow a site to optimize a finite water supply. While some of them do offer watering advice it is in a more general nature and does not give specifics for individual sites. This gives limited lessons that can be learned for this projects application.

## Water Management Application Aimed at Small Sites

Research has shown that there is a lack of applications that are aimed exclusively to support small sites with water management. The only example that could be found was CropWat that was produced by the United Nations Food and Agriculture Organisation. (2009)

This software is aimed at small farmers and allows them to calculate water needs based on plant types, soil type and other factors such as local weather patterns. This application would provide some of the features needed to solve this projects problem of managing water supply based on a range of inputs. It however does not contain a features that gives users a watering regime based on a finite amount of water. Its other disadvantages is that it is old software having been designed for the Windows XP system. This would mean that it would not be able to be used with a lot of modern computer and has the further limitation that it can only be used by a single user and has no way of sharing its results.

The lesson that can be learned from this application is how it calculates its results. It uses data that it has collated from multiple studies to make it relevant to a wide a range of sites as possible. This fits with the aim of the project of making the software as relevant to as many sites as possible and so how this software calculates its outputs could be used as an example for this project.

## Conclusions of Review of Similar Applications

This review has shown that there is not a software solution out there that meets the key objective of this project, which is to give an optimal watering solution for a site based on the water they have stored. The review has revealed some lessons that can be used to guide the application this project is developing. Those lessons are how best to display data to make it easy for users to interpret and how an application may calculate results to make them relevant to a range of sites.

## 5. Project Management

The completion of this project was a substantial piece of work that included researching the problem, developing a solution, coding an application to implement that solution, testing this application and then writing the report. To achieve this the project was planned and delivered using the recognised software development methodology of prototyping. This chapter describes what is meant by a prototype methodology and justifies its use for this project. It then states how it was implemented by this project and finally evaluates how its effectiveness helped this project achieve its required results.

The prototype software development methodology is based on an incremental approach to software development. Prototyping can come in various forms which implement this methodology in different ways. Examples of this are throw away prototyping where the prototype is discarded and the product is developed from the start, Evolutionary prototyping where successive prototypes are built on to create the final product but each prototype is a usable application and incremental prototyping where the prototypes are merged at the end to create the final product.(Somerville 2006)

For this project, an evolutionary prototyping approach was chosen. When using this methodology the developer will start off with a small list of loose requirements and develop a basic product to show potential features that could meet those requirements. This prototype would then be taken to users, clients and other stakeholders for feedback. This feedback would then be used to develop the requirements and used by the developer as a guide to what features and functionality they should prioritise their efforts to achieving. This process is then repeated until the software product is judged to have met the projects original aim and can be released.

The project implemented this methodology by first developing an initial set of requirements for the system based on developers knowledge of the Inspire Community Garden Project

After this the requirements were developed further through further research of the problem and the best way to solve it. The outputs to this are documented in chapter 2 of this report.

Following this research the project then worked on developing a mathematical solution to the problem(See Chapter 3) and started the development of the application concentrating on some of the initial requirements identified in the first stage. Through this the first

prototype was created which had basic functions and simulated how a final product would calculate an optimal solution. This first prototype was tested using observational testing and then presented to the Inspire Community Group on the 16<sup>th</sup> July. Details of the feedback given from that session and the following one held on the 2<sup>nd</sup> August can be found in chapter 6.3 of this report Application Validation and Testing.

Following this session the initial prototype was developed using feedback from the first session. This second prototype had more features and functionality and also had a full implementation of the optimization solver.

This second prototype was taken to the trustees on the 2<sup>nd</sup> August. This meeting served to validate the development that happened as well as agreeing on further features that could be added to the application.

The application was then developed and further testing done through observational testing and Junit testing of key aspects of the application in preparation for the final deadline when the application had to be completed.

A Gantt chart was developed to show key tasks that needed to be completed and the timescales they had to be done by. This gave a quick reference on what tasks should be completed by a certain date and let the developer know if there was any slippage in the timescale that meant action needed to be taken.

#### Why the Methodology was selected

An evolutionary prototyping methodology was selected due to the nature of the project and the demands put on the delivery of the project.

The nature of the project meant it started out with an overall aim and a problem to solve but did not have a detailed path showing the steps to solve the problem or a firm idea what a final solution could look like. In this situation, there is a high risk that the development will get off course and develop an application that does not solve the problem. The use of a prototype methodology meant that as each new iteration of the application was created it allowed checks to be made with perspective users to see if it was still meeting the core aims and requirements of the project.

It also gave the chance for the users to develop their thoughts on what was required and so give better feedback to the developer. While the users of the system knew what result they wanted they did not have firm ideas about how a piece of software would deliver that result. Through using prototypes and letting the users see those prototypes it let them develop their thinking about what features they would like to be included in the application. An example of this was adding the feature of being able to remove plots before their growth cycle has ended. This was suggested in the second feedback session held with the Trustees of Inspire Community Garden. After seeing the first prototype they were able to consider how they would use the application in their current activities. In this case the trustee had lost a plot of plants to disease and realised that the application as they had seen would not be able to deal with that situation. This was fed back to the developer at the second



feedback session and the feature to remove plots before growth cycle had finished was added to the application.

While the prototype methodology gave these advantages, it could be argued that other methodologies would give the same outputs. A traditional PRINCE2 methodology would have used a project executive to gain feedback from users and through the use of a business plan would have ensured that the project kept to the aims of the project. Methodologies of this kind were not used for this project because those methodologies deliver the whole product in stages with the product not complete until all the tasks in those stages have been completed. The risk in this project management methodology is that if there is a problem with implementing one single feature then the whole project could be delayed including the time when a working application is ready. This project had a hard deadline for when a working application was expected. By using a prototype methodology a working application was developed early on and then features added to the application as the project progressed, this lowered the risk of not delivering a working application to the deadline.

#### Evaluation of Chosen Project Management Methodology

Through choosing an evolutionary prototype methodology it allowed this project to lower the risks of non delivering and allowed the developer to develop an fuller understanding of project requirements by allowing users to see prototypes as they were developed and give feedback.

Given that the project started out with very basic requirements I believe that this was the correct approach since it allowed both the developer and users to solidify what they wanted from the application and build an application based on that better understanding.

The disadvantage of this approach was that a lot of the applications development was guided by user feedback. The challenge with that was since the users did not have technical expertise they were not able to comment on technical aspects of the application such as appropriateness of data structures and other technical design choices. This meant that a lot of the feedback was based on visual aspects of the application rather than the logic of the application.

Having a prototype methodology meant that there was more pressure for the developer to get on with the project and deliver a working prototype to meet the milestones of user feedback sessions. This meant that less time was available for research on what technologies to use and the best way to approach the applications development, leading to design decisions being taken on timing of delivering rather than what may be the best technology to use. This caused problems further into development where aspects of the application, in particular the implementation of the optimization algorithms proved harder on the chosen platform than first envisioned. This meant that this part of the project took longer than expected so timing of the project had to be altered to accommodate this.

The reasoning for these decisions and their impact on the project are discussed in more detail in chapters 6.2 and 7.2 of this report. While this project methodology could be blamed for some of the challenges faced by this project by forcing the developer to get on

with development rather than researching available technology. The methodology led to the delivery of a robust application that solved the problem set for it and reflected the needs of its targeted users.

## 6 Application Development

This chapter details the development of the application to implement the problems solution.

### 6.1 – Application Specification

Following the evolutionary prototype methodology development of the application started with some key requirements to guide the direction of the project. Some of these requirements will come from the variables needed for the problem solution detailed in chapter 3 and others came from the review of similar applications done in chapter 4. The rest of the key requirements came from using software engineering techniques to define requirements through envisioning how the application would be used by its targeted users.

The first step was the creation of use cases to imagine how a user would progress through the application, and what outputs would be expected based on the solution detailed in chapter 3 of this report.

#### Use Case Description - Application User gets Optimized Watering Regime

1. User logs into the system for first time and creates new user with a user name and password.
2. User creates new garden.
3. The user then adds plots to the garden with details on their size , plant growing and other variables required for optimization calculation
4. User enters the amount of water available on the site.
5. User enters the number of days they want water use to be optimized for.
6. Application retrieves data from database for plant water requirements.
7. Application uses databases data and user input to create matrices of variables.
8. Application retrieves weather forecast and adds that data to matrices
9. Application inputs matrices into optimization solver.
10. User Interface displays the results in graph format.
11. User chooses if they want to export results or adjust variables.
12. If user selects option to change variables process restarts from 3.
13. If user selects option to export results system to a text file.
14. User closes system.

#### Use Case Description - Edit Plot Detail

1. User logs in to system.
2. User selects garden that has already been created.
3. Applications user interface displays list of plots in that garden.
4. User selects a plot.
5. Application checks if user has permission to edit garden details.

6. If user does not have permission User Interface displays plot information but does not display user interface elements that allow the user to change any details.
7. If user has permission user interface displays elements that allow editing.
8. User makes changes to plot details using interface.
9. Application checks that new plot details are valid.
10. If new plot details not valid then application stops and warns user.
11. If new details are valid then system updates the relevant entries within the garden database.

#### Use Case Description - Add User to Garden

1. User/Garden Creator logs into application.
2. User/Garden Creator loads a garden that they have created.
3. User/Garden Creator selects options to give option to another user to view the garden.
4. The garden owner then enters the name of other Garden User who they want to give access to this garden and selects if they are to have edit rights.
5. Application checks to see if there is a user with the username given in the database.
6. If there is no record of a Garden User with the given username then the application does not proceed and shows a warning to the user.
7. If the user name exist the system then updates the user database to show that the Garden User now has access to that garden.
8. The Garden User will then see the garden when they next log into the application.

Following the creation of these Use Case Description we then identify key actors in the system and create a use case diagram to show which key actions they would be involved with.



Application to allow users to create a garden and then add plots specifying their size, soil type, environment, what is planted, when it was planted and its priority.	This is needed to allow the application to collect the variables identified in chapter 3 needed to solve the problem.
User Interface to display optimization results in graphical format such as graph	In looking at similar applications a lesson learned was that graphs were an effective way of displaying water information
Application to allow users to share results with those who do not have access to the application.	Identified in use case descriptions User gets Optimized Watering Regime
Application to be designed to allow multiple users to edit a single garden	Shown to be a feature lacking in similar applications but would make it more usable to targeted users

Non Functional Requirements	
Key Requirement	Why Key Requirement
People can access and use the application irrespective of what computer or operating system they are using	To meet a overall aim of the project to make a solution that can be used by as many sites as possible
Garden owner able to give other users different level of access rights to edit and view their garden	A standard security protocol that is expected in most applications
Application designed so that a user with limited computing and/or gardening knowledge can still use the application	Links with the overall aim of the project in making a solution that can help as many sites as possible.

## 6.2 Application Design & Implementation

This section discusses how the application was developed and implemented and why certain design choices were made. It should be noted that not all the features described here are to meet requirements identified in the last section. As mentioned this project used a prototyping methodology so some further requirements were developed from meeting with users detailed in chapter 6.3 of this report.

Based on the requirements and specifications detailed in the previous section of this report it was determined that the architecture of the application would be built of 3 main components. At the core of the application would be the optimizer which would take data and constraint parameters undertake an optimization calculation and return a result that gives an optimal watering regime based on plots within the garden and number of days in optimization period. The optimizer part of the application would be supported by a database. This database would hold key data needed to allow the optimizer to undertake its calculations. Finally the application would use a graphical user interface to interact with the users. This user interface would be used by the users to input the parameters of the problem they wanted the optimiser to produce a solution to. Once the calculation was completed the user interface would display its results to the user.

### Programming Language

In the initial design of the project it was decided that the application would be a JavaFX application programmed using the java programming language. This decision was taken to satisfy both system requirements and also project management requirements.

As stated in the project management section of this report it was decided that an evolutionally prototype methodology would be used to deliver the project. To achieve this it was necessary to be able to create a working prototype as quickly as possible to show to the users and gain their feedback. To meet this timescale it was decided to choose a language that the developer was experienced in rather than lose development time in learning a new language.

A further reason that the Java Programming Language was used was to satisfy the requirement that the application could be used by a wide a range of users as possible irrespective of their operating system. The Java Programming language allows this to be achieved through the Java Virtual Machine. This virtual machine can be installed without charge on all Windows, Mac and Linux operating system. Through using its own operating system the Java Runtime Environment to handle system calls to the native operating system, the Java Virtual Machine allows the same source code to run on multiple different operating system and operate in the same way on each one. (Lowe 2014)

JavaFX is a software platform for creating desktop applications as well as web applications that can run across multiple devices(Lowe 2014). This was chosen rather than using the native Java Swing library because it has been announced that it is the intention that JavaFX

will replace Swing as the primary way to create graphical user interfaces (Lowe 2014) therefore allowing the code written for this application to remain functional for future updates of the Java language. JavaFX was also chosen due to libraries for graph creation. Presenting results in graphical form was identified as a requirement of this application so being able to create graphs quickly and easier help meet this requirement, which would have been more difficult in swing where the graphs would have had to be plotted using x, y coordinates.

## Database

In this application the database would be used to store data on plants water requirements and the length of plants growth stages. Another part of the database would be used to store user details once they signed up to the system. Including details of their gardens and the plots in those gardens and details about when they were planted.

This data would then be retrieved by the application as part of the formulas to construct matrices of variables for the optimization function described in chapter 3.

For this application a Postgresql database was used. This was decided on because Postgresql is an open source product that is in wide use. Being widely used means that there is already a range of drivers and dependency software to allow connection through code to a postgresql database. Therefore it made it easier for the developer to connect the application to the database through JDBC libraries.

Postgresql is also available as an option for many cloud based database solution including those offered by Amazon Web Services and Google Cloud. In deployment, the database would be hosted on a server and be accessed through online connection made by the application when it needed to retrieve the information. By doing it this way rather than having the data as part of the application it would allow developers to update the core information used in the calculations without having to issue an update to the applications installed on users computers. This would make the application easier to maintain and easier for users, since they would not need to deal with software updates each time new data is added to the data set.

An Entity Relationship Diagram showing the design of the database is shown in appendix 4.

## Data

For this application to be effective the results of its calculations needed to be based on reliable datasets. As already stated in chapter 2 the problem was not that there was a lack of sources of data but that there was too much of it with much of it being either too broad or based solely on observation from one geographical area. This meant that the data was not in a format that could be input into the model created in chapter 3 or the data would be so geographical specific that the results produced would not be relevant to other sites.

Two sources were found however, where data could be extrapolated to create the data set on plant water requirement and plant growth stage length that was needed for this



application to produce relevant results. Those 2 sources were (Bleasdale, Slater 1991) & (Brouwer, HeiBloum 1986). These sources were used because the books although older were aimed at giving information to a wider variety of sites in a range of geographical areas. The datasets in those books were taken from observations done in multiple studies to give values that were relevant to a wide a range as possible. Through using these data sets it helped the application achieve one of its requirements of producing results that were relevant to as wide a range of users as possible.

### Optimizer

The 2<sup>nd</sup> component of the application the optimizer was to take in the given variables and compute a optimal watering schedule based on those variables and the constraints given.

### Optimization Solver

Part of the application would be an optimization solver that would take the variables and parameters and using the model and algorithms identified in chapter 3 test multiple solutions until an optimal result was found. Rather than recode these algorithms it was decided to use already constructed optimization solvers to implement those algorithms and return results.

These solvers had already been rigorously tested, with many having multiple releases where they had been improved on, based on feedback. They also had well developed API's which allowed developers to program their problem in simple readable code. Below is a review of solver considered and tried as part of this project.

Solver Name	Java Optimization Modeler Using The GNU Linear Programming Kit Solver (GLPK) <a href="http://www.net2plan.com">http://www.net2plan.com</a>
Description	A library created to allow java programmers to define an optimization problem and then call the GLPK solver that uses the Simplex Method
Advantages	Very easy to code and model your objective
Disadvantages	Was designed primary as a teaching tool so is limited in functionality
Results and Conclusions	The results of the test was while the library made programming easier since only one algorithm was used the results were limited

Solver Name	Choco Solver <a href="http://www.choco-solver.org/">http://www.choco-solver.org/</a>
Description	“Choco is a Free Open-Source Java library dedicated to Constraint Programming. The user models its problem in a declarative way by stating the set of constraints that need to be satisfied in every solution. Then, the problem is solved by alternating constraint filtering algorithms with a search mechanism.” Choco-Solver Website
Advantages	By using a constraint focused method it ensures that a solution is found that respects the constraints
Disadvantages	Only allows simple linear objective functions
Results and Conclusions	Since the objective function for this problem identified in chapter 3 of this report has few constraints this form of optimization programming was not considered appropriate.

Solver Name	Apache Maths Optim
Description	An open source Java Library that allows a range of mathematical functions. Allows user to program a objective function and then solve using the Simplex method
Advantages	A very large library which comes with methods to allow development of model through the manipulation of matrices
Disadvantages	For linear programming problems only uses the Simplex method
Results and Conclusions	While a big library of methods it again had the problem of some of the other solvers of only using a single algorithm that would not fully test all potential solutions.

Solver Name	Linprog
Description	A solver for minimization optimization problems it uses the dual Simplex algorithm(see chapter 2) and Interior Point Legacy Algorithm. This algorithm is similar to Interior Point Algorithm(see chapter 2) but undertakes steps to resolve the problem before running algorithm to lesson the iterations needed.
Advantages	Using 2 algorithms including the dual simple method makes this an extremely powerful solver.
Disadvantages	The disadvantage at the current time there is no Java library that implements the Linprog solver. It can be implemented either by using MatLab, Python or R platforms
Results and Conclusions	The Linprog server produced good results for a range of scenarios that gave a result array that reflected the priorities more

Having tested the above solvers it was determined that the Linprog solver gave a result that was most reflective of the priority weightings of the different plots. What was found as a result of these tests was as stated in chapter 3 the Simplex method is an edge based algorithm that finds optimal solutions on the edge of the feasible zone. What this meant for this projects problem was the solution given would move from the desired value to the basic value since they represented edges of the feasible solution zone. Solvers that used interior point method gave better results since they considered points within the feasible zone as well.

The challenge from this was that the Linprog solver did not have a Java language implementation, therefore the application would have to call another software platform to run the Linprog solver and return results. The Linprog server could be run on both Python and Matlab. The Python implementation of the Linprog solver was done through the python package Scipy. While it is possible to run python scripts through Jython an implementation of the Python language for the Java platform it does not run the Scipy package. (Hugunin 2017) MatLab is a commercial product but had a feature called Java builder that allowed the developer to create Java classes that implement the functionality of MatLab functions. Matlab then allows this to be bundled as a Jar file and along with the Java Builder run time Jar it allowed Java programs to run MatLab scripts. Even if a user does not have Matlab installed the program will run on a freely available MatLab runtime environment. (MathWorks 2017)

This allowed the application to implement a optimization solver by creating matrices to give bounds of the problem and then running the algorithms through the Linprog solver by calling MatLab scripts saved as dependencies of the application.

## Data Structures

The primary data structure used by the application is the TreeMap in the optimizer part of the application it is used to store input variables for the solver and the output results.

The Optimizer class of the application creates TreeMap data structures to hold variables showing basic and desired water requirements for the garden. Once results are returned from the optimization solver they are then collated as a TreeMap and returned. In each case the name of the plot acts as a key and the value is an ArrayList containing values for each day on the optimization period for that plot.

The TreeMap data structures was used because having a key value pair allowed each line of data being input or returned to be identified by plot name. This made it quicker and easier to be able to compare plots, identify which plot a vector referred to and give an efficient way to add new plots to the solver.

TreeMaps were also used to populate values of GUI elements. For example in constructing graphs series the key value was used and identified for each series. Rather than having to find the plot identify by index which would have required extra computation to identify which row of data referred to which plot. The TreeMap data structures also orders its entry by the key. This made it helpful in organising elements to be displayed logically in the user interface. A further advantage to this is that in a TreeMap a particular key can be searched for in  $O(\log n)$  time. This means that the application can retrieve and display the data to match the users selection quickly.

The trade off for using the Tree Map data structures is that it is not a form that can be passed to the Matlab Linprog solver, meaning that extra memory and computation resources would have to be used in adapting the data structures so it could be processed by the solver. This was accepted because in the majority of use cases the number of variables that would have to be extracted from the TreeMap and put into the data structure required for the solver would be less than 1000 so extra computational power required would not be high. The other tradeoff with using the TreeMap data structure to record plots with the plot name as key so all plot names had to be unique and could not be used more than once in each garden. This meant further code had to be written for the user interface to check and restrict plot names chosen by the user to ensure that they did not duplicate any other plot names in that garden.

### Getting Weather Data

As discussed in chapter 2 temperature and rainfall impact on a plants water requirement. This led to a requirement being identified, the application takes into account local weather conditions when determining water requirements.

To accomplish this the application makes use of the online weather data source Weather Underground to get forecast for 10 days from the current date. The application uses this data and service by first establishing a location where relevant weather data can be requested for.(The Weather Company 2017) This is done when the user first creates their garden. At this point the user is prompted to enter part of a location for their garden. The application uses this data to query the weather underground system through a http request. In response to this query Weather Underground sends back a JSON file containing possible location that match the term entered by the user. The application then uses methods within the JSONSimple library(Fang, Nokleberg, Hughes 2012) to construct a TreeMap that uses the text location as a key and unique identifier from weather Underground as the value. The applications User Interface shows a list of these potential locations to the user, who selects the most relevant one. The relevant entry from the TreeMap are then stored as part of the gardens entry in the applications database.

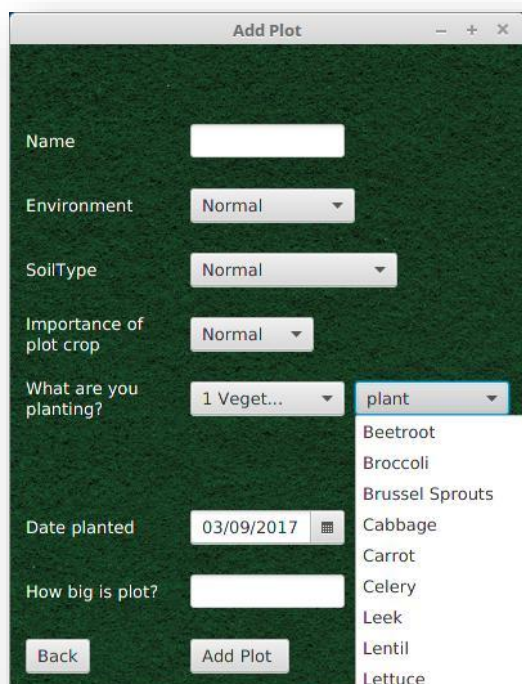
When the optimization function is undertaken the application then uses this location reference to send a further query to the Weather Underground API. This query returns another JSON file containing weather forecast for a 10 day period. The application then

again uses method from JsonSimple library to extract the weather values and input them into the model as described in chapter 3.

### User Interface Design

In designing the user interface for this application, a lot of attention was given to considering the kind of users that would be using this application. Since a requirement was for the application to be used by a wide a range of users as possible then it was assumed that the users of this application would be of different ages and have different levels of skill in IT.

For this reason an objective was to make the user interface as simple and as easy to use a possible. In reviewing similar application a conclusion was reached that these application were too complicated for this projects targeted users, expecting a level of knowledge in irrigation engineering to be able to enter the required values. Therefore the UI was designed to avoid this problem by being designed so the user had to type as little input as possible (see Appendix 3) and to where ever possible give the user a list of options and a default that they could choose if they were unsure of which option to pick. This initial design was developed in each prototype of the application created.



The screenshot shows a window titled "Add Plot" with a dark green background. It contains several input fields and dropdown menus. The fields are: "Name" (text input), "Environment" (dropdown menu with "Normal" selected), "SoilType" (dropdown menu with "Normal" selected), "Importance of plot crop" (dropdown menu with "Normal" selected), "What are you planting?" (dropdown menu with "1 Veget..." selected and a list of plants open), "Date planted" (text input with "03/09/2017" and a calendar icon), and "How big is plot?" (text input). At the bottom are "Back" and "Add Plot" buttons. The list of plants includes: Beetroot, Broccoli, Brussel Sprouts, Cabbage, Carrot, Celery, Leek, Lentil, and Lettuce.

### 2 - Screenshot showing UI element giving users list to choose from and default options

The user interface was also used to show results of the optimization part of the application to meet the requirement that results were shown in a way that the user could understand and use. Therefore the user interface was designed to display results from the optimization

solver in graph form to the user showing how much water is used on the garden over the time period and how it relates to the desired and basic amounts required by the garden.

The application through the UI also used the results to display the optimal watering for each plot on any given day. This is shown in screen shot 2 in chapter 7 of this report.

### Allowing Experimentation

Partway through the development process a further requirement was identified to allow the user to test the impact on water usage if they added more plots without having to commit those plots to the garden. An example would be if a user wanted to see how much adding a new plot of a particular plant type would reduce water allocations to other plots. The application delivers this feature once the initial optimization is calculated. Once the results are shown the user interface allows the user to enter further plots to the garden. These plots are then saved in temporary data structures that are used to undertake further optimization calculations using the optimization solver. If the user then chooses to save the new plots they are added to the users garden and recorded in the applications database, otherwise they are discarded when the user moves to another part of the application.

### Security and Sharing Data

The data that this application uses was not judged to be of a highly sensitive nature. It was however felt that users would want a basic level of access permission to allow them to stop other users changing their garden information either by accident or maliciously. To accomplish this a basic login functionality was designed to have users select username and password when first signed up and then for the application to check these each time the user logs in to the application.

One of the key requirements identified for this application was that users are able to share their results with other users of the garden both those who have the application installed and those who do not.

The application was designed to fulfil the 2 parts of this requirement in 2 different ways. To allow the user to share results with people who may not have access to the application, a feature was designed that allowed the user to save results of the optimization to a text file showing required amount of water needed to be given to each plot on each day of the optimization period.

The second way was that users could allow other users of the system access to their gardens. The application does this by allowing the user to enter another users username and assign them a permission level. When this happens the application confirms that the username entered is registered and updates the database accordingly. When that user next logs into the application they are given the option of loading the garden they have been given access to. If they have been given edit rights the application user interface will show elements that allow them to undertake edits.

## Coping with Failures and Bad Data

To ensure robustness of the application features were implemented that lets the application deal with failures and the input of wrong data and if possible still allow the user to complete their task. Below is a table showing some of the main anticipated issues and how the application was designed to cope with them.

Description of Failure	How application deals with failure
User fails to enter data in required format. Example: user enters plot size in word format rather than numeric	All input data is checked against regex expression to see if it is right type and values. If not application warns user and does not continue
User fails to select variable that is needed for calculations. Example user does not select number of days but still asks application to calculate optimal	Before optimization takes place application does check to ensure it has necessary values and there is no null entries
Application is unable to access the database	The application catches this throw an IO Exception. Application will warn user and if possible provide data that will allow application to continue. For example if cannot access database to get user details gives user a Guest User Name
Application unable to retrieve weather data from Weather Underground service	If and IO or JSON Parse exception are thrown the application will return matrix of ones so that calculation can still be undertaken without weather data.
Sudden crash of computer application in running on	Each time application creates new information such as a plot added to a garden this change is recorded in the database. This would allow the user to retrieve information quickly once they restarted the application.
Malicious attempts to get access to database through SQL Injection techniques	Access to database done through prepared statements to prevent SQL Injection attacks

### 6.3 - Application Validation & Testing

In developing the application a broad set of requirements were developed to give a general direction on how the system would solve the problem that has been described in chapter 1 of this report. Once this broad plan was in place a draft of the test plan was written to show how the iterative prototypes would be tested.

#### Test Environment

The requirements have identified that the final application will be used by a range of users using multiple computers with different operating systems. The application is a java application so should be able to run on any machine that has the java virtual machine. To make sure that all the dependencies still work all the tests were run on a Windows and a Unix based Ubuntu Linux Operating System.

#### Output Testing

These tests tested the core functionality of the application to show if it produces relevant output for a range of inputs. These tests were not there to test the algorithms used by the solvers used as part of the application. These tests looked to test that the applications is putting relevant variables into the optimization model that the solver is using and that results meet the aims of the application. This was tested using Junit test using test data.

Test description	Expected Results
Test 1 If $x_{1,1}$ $x_{1,2}$ have same optimal and basic values but $x_{1,1}$ priority value > $x_{1,2}$ priority value	Decision value for $x_{1,1} > x_{1,2}$
Test 2 Undertake optimization with garden where some of the plots have completed their growth cycle	Optimizer does not take into account those plots when calculating results.
Test 3 Plot in garden not set as planted until 5 days after optimization period starts	Optimizer should not take that plot as part of its calculations until after the 5 <sup>th</sup> day
Test 4 Tester runs optimization function with set number of days and water available with and without weather variables added	Different optimization results for the test which includes weather variables



## Error Checking

This part of the testing looked at how the systems will deal with system errors such as IO failures. It will also test how the system deals with errors in data entered by users. This will be done by tester trying the bad entries and observing how the system reacts.

Test Description	Expected Results
Test 1 - No Entry of data User leaves text box blank or does not select value on slider and pushes action button.	That the system will not continue until all required information is available. Application gives error message and outlines offending text box
Test 2 - wrong data types entered in text box. Example words put in box where plot size is required	System to not take wrong data and warn user that data type is wrong
Test 3 - Enter wrong username and password  User enters a user name with wrong password	System to not allow user any further into system and to let them know input has not been accepted.
Test 4 – User choses to delete plot in error	System warns user about action and only continues with deletion if user reconfirms their action.
Test 6 – User enters negative number for water available on site	System displays error message and does not proceed.
Test 7 – Tester shuts application down part way through optimization calculation	All relevant data should have been saved in database so can redo calculation when application restarted and user re-enters requirements

## User Prototype Testing

To test that the application met user requirements prototypes of the application were shown to trustees of the Chesterfield Inspire Community Garden who were asked to navigate the application and implement its functions.

Below is a summary of those sessions, the feedback given and conclusion on how this feedback was used to further develop the application.

## Test Session 1 - 16th July 2017

Testers: Inspire Community Garden Trustees Sharon Sutton, Chris Allen and Steve Samson.

Test Environment: The tests were done on the application being run on a laptop computer running the Linux Mint Operating System. Testers were given the use of the laptop keyboard and an external mouse to use.

Prototype: The prototype consisted of a database with a limited data set and a UI that allowed users to build their garden. A full optimization solution had not been coded at this point so the application was run using a basic algorithm that approximated the results.

### Observations:

Testers were able to navigate through the UI with little explanation and were able to create their own gardens with plots and set up an optimization problem.

What did you like about the application?

"I like how I can give plots priorities to ensure my favourite plants get the most water."

"This would be really useful and gives us good information about what water to do each day."

"I like the idea of showing the requirements in graphs rather than just numbers".

What did you not like about the application?

"It looks pretty bland reminds me of applications from the 90's"

"It is not good how you enter your water needs get your result and that is it. I may want to use this for my own garden as well and do not want to have to shutdown and log in each time".

What Improvements would you make?

"It would be great if this could be made into an [mobile] app so I could take it on to the garden and use it".

"I would want it to be able to show people what watering was required on the days I am not there."

## Response to Test Feedback

The tests showed that the basic concept of the application was supported and it was producing results that would be useful to the prospective users. The feedback on how they would use the system to develop watering regimes for multiple gardens in the same sessions had not been considered therefore it was resolved to allow users to navigate both backwards and forwards through the application once they were logged in.

While the testers idea that the application could be done as a mobile app was acknowledged it was decided not to act on this feedback. The reasons for this was that the application was developed as a Java application since that was a technology the developer understood and so would be able to devote more time to the applications development. To change the project at this stage would have involved more time to learn mobile application development and would give a weaker final product. Although this could be future development once the Java Application has been finalised.

### Test Session 2 - 3rd August 2017

Testers: Inspire Community Garden Trustees Sharon Sutton and Steve Samson

Environment: Environment was same as the first test session held on the 16th July.

Prototype: The iteration of the application that the testers used had been developed to include all elements of the UI including the ability to edit existing plots and now used the optimization solver that would be used in final product. The application was now also able to pull down weather data and integrate into water optimization recommendations.

#### Observation

It was observed that the testers were able to navigate through the user interface create their own gardens and get optimization results for it. The only point of confusion was when testers were creating a garden and adding its location. At this point they were not able to determine how to get a list of possible locations to appear for them to select from.

What did you like about the application?

“I like the new feature that allows you to add users to your garden and that you can give them different levels of permissions”.

“The backgrounds look better now more colour is always better”.

What did you not like about the application?

“I don’t like how small the writing on some of the buttons is I need my glasses to see it”.

“I still think that this would be best suited as an app on my phone”.

What improvements would you make?

“I want the ability to remove plots before their growth cycle ends. Sometimes crops fail for reasons such as disease and I would want to be able to take them out of my water calculations”.

“I don’t trust weather forecasts can we make it so I can see the results without weather taken into account to then make my own judgements on if I should water that day.”

“I think displaying water requirements for plots in square inches is best. It is easier for people to visualize than the other water measurements. A lot of water storage containers have markings that are in inches<sup>3</sup> so those measurements can be related to”

#### Response to Test Feedback

The observation showed that users had a particular problem with entering a garden location. To address this the user interface was changed to give user instructions on how they should enter part of a location name.

In response to the suggested improvements a feature that allowed users to delete plots from their garden was added. Along with this further development of the user interface was undertaken to make some of the button text easier to read. The feedback on the display of water requirements in inches was in agreement with the developers reasoning so was kept as part of this application. In response to the request that the users see the results without weather calculations the application was altered to allow the user to opt in to using the weather. If they were not the application would not send a query to the weather service and would just use a matrix of ones and a matrix of zeros to represent temperature and rainfall variables in the objective function described in chapter 3.

## 7.0 - Evaluation

Having detailed in the last chapter how the application was designed, developed and tested this chapter now evaluates the output of the project to see if it met the original aims stated in the introduction to this report.

This chapter undertakes this task by first reviewing if the application produced meets the specified requirements. The chapter then goes on to evaluate the project as a whole and see if any lessons can be learned from how the project was delivered. The final section of this chapter will look at potential future developments that could be added to the application in any future iterations.

### 7.1 Results of Project

The success of any application must be judged by how well it met its original requirements. Below is a table showing how the finalised application met those requirements.

Functional Requirements	
Key Requirements	Was Requirement Met?
Application to produce a watering regime for the plants that will maximize crop production by allocating water to plots based on priority while not using any more water than is available	As can be seen on screen shot 1 the application does produce a water regime based on the water available
Application can retrieve data on plants water requirement and stage lengths	This requirement was met by the development of a database using sets of data described in chapter 6.2
Application to be able to get up to date weather information for the site	This was accomplished through the application sending a http requests to an online weather service. This request returned weather data for a 10 day period in a JSON file that the application parses and to get the variables it needs.
Application to allow users to create a garden and then add plots specifying their size, soil type, environment, what is planted, when it was planted and the its priority.	Screen shot 2 below shows that this functionality was included within the application.
User Interface to display optimization results in graphical format such as graph	This was completed using JavaFX libraries and is shown in screen shot 1
Application to allow users to share results with those who do not have access to the application.	The application allows users to export the results in a text format allowing them to be printed or emailed to other parties
Application to allow users to test impact of adding plots before they save them to their garden	Screenshot 1 shows this functionality. By using temporary data structures the application is able to undertake optimization calculation with test plots without having to save them to the users garden
Application to be designed to allow multiple users to edit a single garden	The application allows the users to give edit rights for their garden to other users. When a user loads another users garden the application checks their permission level and displays appropriate user interface elements.

Non Functional Requirements	
Key Requirement	Was Requirement Met?
People can access and use the application irrespective of what computer or operating system they are using	Developing this application in Java meant that it would be able to run on a wide a variety of computers as possible. Using a MatLab implementation meant that users would have to download the MatLab run time but that could be run on a range of operating systems as well.
Garden owner able to give other users different level of access rights to edit and view their garden	The application allows users to give access rights to their garden to other users.
Application designed so that a user with limited computing and/or gardening knowledge can still use the application	As shown in the screen shot in the next section of this report drop down menus were used to make data entry simple for users. An identified problem is that the applications users interface does not conform to Human Computer Interaction best practise therefore may be confusing for some users.

Since this project used a evolutionary prototype methodology for its development further requirements were developed in meetings with perspective users as detailed in chapter 6 of this report

Further Identified Requirements	
Identified Requirement	Was requirement met?
The application allows users to edit and get optimal water regime for multiple gardens in the same session.	After this requirement was requested the application was modified so that user could navigate back to garden selection page and load a different garden.
User able to delete a plot before its growth cycle has completed	Code developed that allows user to select plot and choose to remove it. Application then deletes that plot from the current data structures and removes its reference from the database.
That users be able to choose if weather elements are taken into account when calculating the optimal water usage	User Interface was altered to allow user to select if they want weather included or not. If the user does not select to use weather the application creates a matrix of ones for temperatures and zeroes for rainfalls to input in the objective function described in chapter 3. The application also does this if the user selects an optimization period that is more than 10 days in the future from the current day.
Application to allow users to test impact of adding plots before they save them to their garden	Identified in screenshot 1 on next page

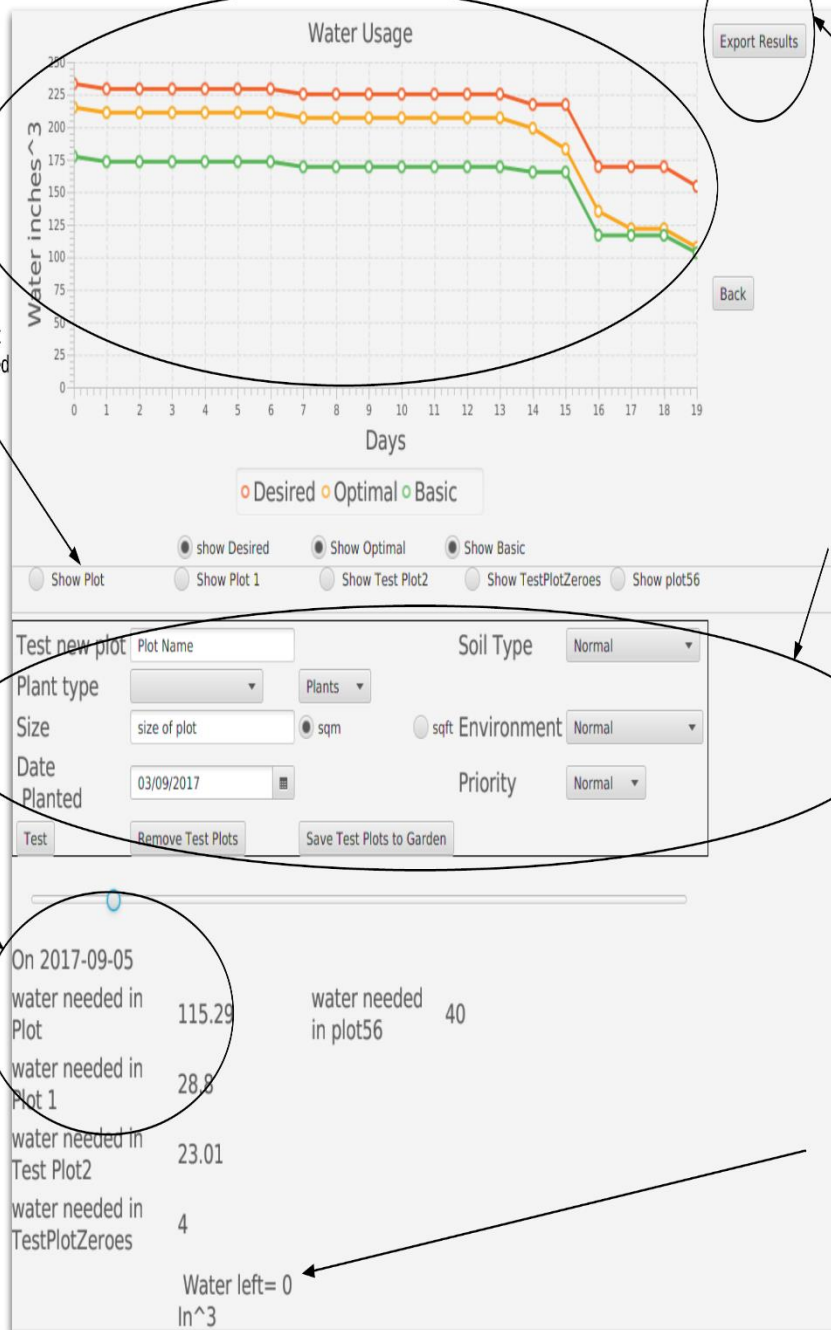
Screen Shot 1

Graph to show optimal water usage in relation to desired and basic amounts

User able to select information displayed

Export Results

Ability to export results for printing or email

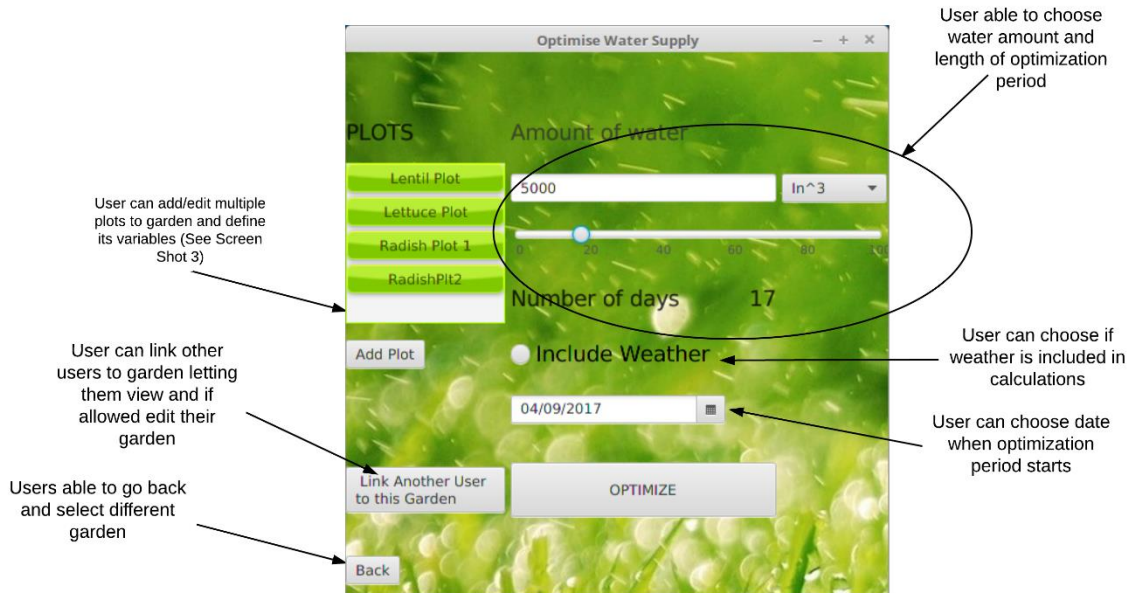


Shows water to be given to each plot on each day

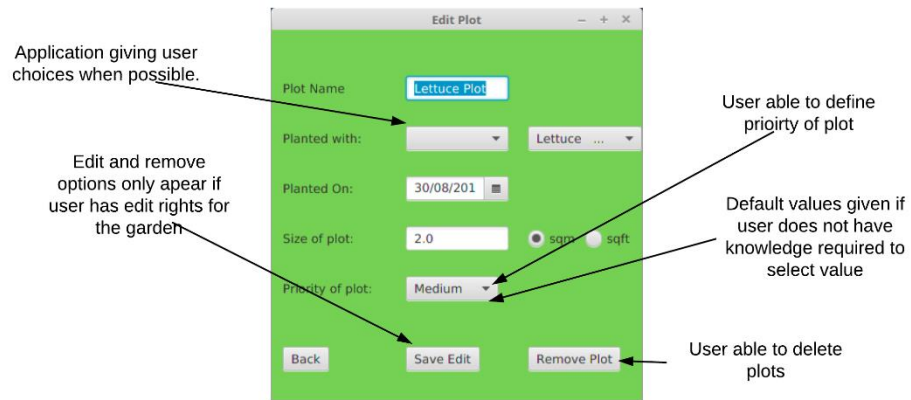
Ability to test different scenarios

Shows if any of the water stored is left over at end of optimization period

Screen Shot 2



Screen Shot 3





## 7.2 Evaluation of the Process

The development of this project used an evolutionary prototype development methodology. It was decided to use this methodology due to the high importance placed on how well targeted users could navigate and get results from the application.

This was done by developing a series of prototypes that were then demonstrated to the targeted users of Inspire Community Garden. These sessions and their impact on development of the application are detailed in chapter 6.3. This was a worthwhile exercise which helped show what needed to be developed in each iteration of the product to meet those user's needs.

These sessions were done on an informal basis with the developer in the room while the trustees were testing the system. While this does not meet full best practise for getting potential users feedback, it could be argued for instance, that the fact that the trustees knew the developer of the system would lead to feedback that was more positive than the application deserved. It did however give a way to get useful user feedback that was implemented into the application within the timescale set by the project.

This process had a positive impact and gave an application that was more in keeping with users needs. For example the application was originally designed so that the user would go through the steps of defining the optimization and getting their results and then shutting down the application as it was assumed by the developer that once they had their results they would be finished. In the first feedback session it became obvious that users would want to move back as well as forward through the stages of the process, changing the garden they were developing water requirements for, or even plot names they did not like. If this feedback session had not taken place and the developer had continued with the assumption the end result would have been a product that would have delivered a solution to the problem but not quite in the way the users wanted.

An example of a new function being added due to feedback sessions was the user's identifying that they may want to get rid of a plot before its plants growth cycle had finished. Having this process allowed the users to see the application working and then able to link it to how they would use it. This meant that new functions that were not initially thought about could be identified and that there was time left for them to be implemented before final deadline. One criticism of the prototype approach and one that was evident in this process was that the majority of users feedback was based on the design of the user interface rather than the core functionality of the application. While this feedback was useful it meant more time was spent on user interface rather than core functionality of the application.

Overall the process was well managed and most of the stages were delivered to their original set time scale. In retrospect looking back at the project more time should have been allocated to researching the technology to be used in the application. Not doing this led to challenges and problems detailed in the following section.

### 7.3 - Evaluation of Application Design Choices

It was decided very early on in the project that the application would be built using the java programming language. The overriding factor in choosing this was that the developer had the most experience in using the Java language. It was felt that there would have been more gains in the project for the developer to take time to expand their already existing knowledge in Java rather than having to spend time during the project learning a new language from a basic level.

This knowledge allowed the developer to produce working prototypes of the final application to show stakeholder in the process and get feedback. This familiarity also allowed further design decisions to be made that improved the usability of the application. The use of JavaFx to create the user interface was one such design decision that benefited from this knowledge. One of the requirements for the application was that output information being displayed in an easy to understand graphical way such as graphs. The developer's experience meant they were aware of the deficiencies of the Java Swing package and so was able to select JavaFX as a better option for the creation of user interfaces for displaying the results in graph form.

Using Java also met the objective of allowing as many users as possible to use the application and not be dependent on what computer they would be running it on.

While these were big advantages the use of Java did have a big disadvantage that led to problems with the project and timescales for parts of its development taking longer than first estimated. In the last section a criticism on the process was that not enough time had been taken to research the technology around the problem that was being solved.

The problem was one of optimizing a range of variables to give the best possible output. To do this it was required to use a third party optimization solver as part of the application. Research undertaken at the start of the project showed that java has optimization solver. Although there were not as many available for it as there are for other languages. In a survey of linear programming solvers undertaken in June 2017 it was estimated that there were 22 linear programming servers available to Java programmers lower than Python at 28 and C++ with 35.(Fourer 2017) While the Java language had less optimization solvers it could use it was still hoped that one would be able to give the solution the application needed.

During this project a range of java solvers were tested to see if they gave results that met the objectives of the project. This testing detailed in section 6.2 of this report showed that due to the algorithms that the Java solvers used the results they produced were limited. This led to the application having to call a solver from another software package MatLab. This meant that extra coding was required and extra dependency libraries were added to the application making it harder to deploy. This also meant that coding an optimization into the application took 4 weeks rather than the 2 weeks allocated in the original project plan. Had more time been allocated at the start of the project to research solvers this problem may have been identified and the design of the application could have been rethought to take account of this.

A further design decision that had to be made was regarding the data that the application would use in its calculations. To undertake successful calculations the application would need access to data that was accurate. The challenge was to decide which of these data sources was the most appropriate to be used by the application. Many of the data sources could be dismissed because they were either too vague and did not give the figures required or too specific to a geographical region.

The decision on what data to use is described in chapter 6.2 of this report. It is accepted that using this data would mean that due to this averaging out of the input data the output data may not give the exact optimal amounts for a site of a geographical location had it used data representative of that area. In developing this application it could have been based on studies that were conducted in geographical areas similar to that of the Inspire Community Garden. It was decided however, to use the selected data sources because of the aim that the solutions it provided by the application be relevant to a range of sites as possible. Through using data that was based on a range of sites the results that would be produced would be relevant to a greater number of sites and so meet that objective of the project.

#### 7.4 Lessons Learned

Despite the trade-offs and challenges that have just been specified the project did meet its objectives to provide a solution that allowed small sites to manage finite water supplies to optimize their crop production. The development of an application that allowed the user to enter a range of plots which the application would give an output detailing the amount of water that would be given to each of those plot on a day by day basis meant that users could implement the solution.

This was not without its challenges and from it a number of lessons have been learned that would be relevant to future software development projects.

The first lesson that can be taken from this project is the value of getting input from users on the application before it is complete. By using a prototype development methodology it allowed the developer to get feedback on the application during production. This was important because even though the developer knew the project and its objective from their involvement, they still made assumptions about how the users would use the system. As detailed in section 6.3 of this report the feedback session showed this assumption was not true. Had those sessions not taken place the developer would have continued to develop an application that had solved the problem but not reflect how users would use the application.

The second and most important lesson that was learned was the importance in giving enough time at the start of the project to research your proposed approach and see if it is the most efficient way to go about this. This includes technology you are intending to use, programming language used and the how you intend to deploy the application so that it can be accessed by the users. As already mentioned in the last 2 sections, not doing this enough and making design decisions without this information led to a lot of wasted efforts trying to

program a solution with Java Optimizer Solvers which would have been much more simple in other programming languages.

## 7.5 Further Developments

While this application meets the identified requirements, and solves the original problem. There were features that are not included due to time constraints that would make it a greater tool to help small sites optimize finite water resources.

The application as it was presented got the information to make up variables for the optimization problem from its own database and from user input. As stated in the section on design decisions the data sources were chosen so they would give variable values that were as close to the desired amount for a range of sites no matter what climatic area they were located in. A way to improve this application would be to allow it to receive real time data from monitoring sensors that could monitor weather conditions, soil moisture content and a range of other local factors. This would allow the application to create an optimal watering schedule based on the most relevant factors. A local weather station would mean that the application would be able to take into account other factors that may impact plant water need such as wind speed and direction. The application being able to get up to date information about soil moisture content would give real time feedback on plant water need and let the application monitor how effective its water optimization is. Giving an application that would be comparable to some of the irrigation management systems reviewed in chapter 4 while still remaining loyal to its objective of providing a solution to small sites.

Another direction the application could be taken is to allow the garden and its user to optimize more aspects of how it waters the garden. A common use for optimization solvers and linear programming is task scheduling and employee shift allocation. A module with similar function could be added to this application that takes the inputs about the availability of the gardeners and using common optimization solvers create a schedule that ensures that there is enough gardeners assigned to watering on each day to implement the given optimal watering regime.

As already mentioned in this chapter, one of the reasons chosen to code this project using the Java programming language was because Java is very portable so the application could be run on any computer with the Java Virtual Machine installed. While this helps meet the identified requirement of allowing a range of users to use the application this could be further refined. Gardening applications are increasing going down the route of using mobile apps to take in user input and give back results and this was suggested within the feedback session with users. This makes sense because these devices are more portable and therefore easier to use on sites. While this project did not go down this route due to reason discussed earlier it is something the application could be developed into the future.

## 8 Conclusion

This project was developed from a real life problem so has always had a clear end result in mind as it has progressed. Once the problem had been established and its key requirements identified, a lot of work then went into identifying the variables that impacted on the water requirements of plants.

Solving optimization through linear programming has been a big part of computer science and a growing part of commercial software development. Linear programming is used throughout a range of industries to make operations and processes deliver outputs in the most efficient way with the lowest costs. This is seen in agriculture where large farms are now investing in sensors that constantly feedback current conditions. Optimization programs then use this information to control irrigation systems to give water where it is needed but make sure none is wasted.

This project looked to using some of those techniques to give information to small sites that would not have access to sensors but still needed information that allowed them to most effectively use their finite water resources. This led to challenges in deciding how to create a solution that could be used on a wide a range of sites as possible without making the system too complicated.

To meet these goals an evolutionary prototype methodology was used allowing the optimization model and its variables to be built into an application that would give a solution to the problem.

This application was designed and built so that it could be used by a wide a range of users as possible and reflected how they would use the application. Using the process and methodology described in this report I believe that an application has been produced that does just that, and could very well be in use by Chesterfield Inspire Community Garden next summer.

## Appendix 1 – How to Run the Application

Download all files from <https://git-teaching.cs.bham.ac.uk/mod-msc-proj-2016/sxj638.git>

Contents of Repository

Optimizer, UI and Res directories containing application Java files to be compiled

Test directory containing tests

All other directories not relevant to this application

### Installing the Database

1. To run this application you must set up a postgresql database. If you do not have Postgresql installed follow the instructions here:  
<https://www.postgresql.org/download/>
2. Move the files plants.csv and mscDatabase.sql from the res folder and save in directory.
3. Type: psql CREATE NEW DATABASE [Name of Database]
4. Then type: psql -d [name of Database] -f mscDatabase.sql
5. Load the database and type the following command:  
COPY plants( name, type, st1\_days, st1\_ow, st1\_bw, st2\_days, st2\_ow, st2\_bw, st3\_days, st3\_ow, st3\_bw, st4\_days, st4\_ow, st4\_bw, plants\_persqm) FROM '[FilePath]/plants.csv'  
DELIMITER ',' CSV HEADER;
6. Open the file Database.java in the Optimizer directory
7. Change the static field variables username and password to your postgresql credentials

### Running the Application

1. Ensure that Jar files in folder Jar are within the class path
2. Move contents of the files Optimizer, UI and res to your working directory
3. Run the main method in file View.java contained in the Optimizer folder

## Appendix 2 – Bibliography & References

The Farmers Almanac 2017 [Homepage of Yankee Publishing], Available: <https://www.almanac.com/> [ Accessed: 6/15, 2017].

ALLEN, R.G., PERIERA, L.S., RAIES, D. and SMITH, M., 1998. *Crop evapotranspiration - Guidelines for computing crop water requirements - FAO Irrigation and drainage paper 56* . 1 edn. Rome: Food and Agriculture Organisation of the United Nations.

ASHRAF, M. and MAJEED, A., 2006. *Water Requirements of Major Crops for Different Agro-Climatic Zones of Balochistan*. Quetta, Pakistan: The World Conservation Union.

BLEASDALE, J.K.A. and SLATER, P.J., 1991. *The Complete Know and Grow*. 2 edn. Oxford: Oxford University Press.

BOYD, S. and VANDENBERGHE, L., 2009. *Convex Optimization*. 7 edn. Cambridge: Cambridge University Press.

BROUWER, C. and HEIBLOUM, M., 1986. *Irrigation Water Management : Irrigation Water Needs*. Rome Italy: Food & Agriculture Organisation of the United Nations.

BURKE, J., 2016-last update, Maths 407 - Linear Optimization University of Washington [Homepage of University of Washington], [Online]. Available: <https://sites.math.washington.edu/~burke/crs/407/notes/section1.pdf> [Accessed:15 July 2017] .

CARR, G., 2016. The Future of Agriculture. *Technology Quarterly*, 6/9/16, pp. Chapter 2 - 3.

CHEN, Y., 1997. Management of water resources using improved genetic algorithms. *Computers and Electronics in Agriculture*, 18(2-3), pp. 117-117-227.

CHONG, E. and ZAK, S., 2001. *An Introduction To Optimization*. 2 edn. New York: John Wiley & Sons Inc.

Fang, Y., Nokleberg, C., Hughes, D. 2012. *JSONSimple A Simple Java Toolkit for JSON*. Available: <https://github.com/fangyidong/json-simple>

FOOD AND AGRICULTURE ORGANIZATION OF THE UNITED NATIONS, 2009, *CropWat – Software for Calculation of Water Requirements*, Available: <http://www.fao.org/land-water/databases-and-software/cropwat/en/>

FOURER, R., 2017. *2017 LINEAR PROGRAMMING SOFTWARE SURVEY*. <http://www.orms-today.org/surveys/LP/LP-survey.html> . Institute of Operations Research And The Management Sciences.

GLASER, A., 2016. Dig Gardening? Plant Some Connected Tech This Spring. [Online]. Available: <https://www.wired.com/2016/04/connected-gardening-tech-iot/>

HUGUNIN, J., 2017-last update, Jython: Python for the Java Platform. Available: <http://www.jython.org> [Accessed:1/8, 2017].

HUNTER INDUSTRIES, 2017 last update, *IMMS®/ Hunter Industries*, Available: <https://www.hunterindustries.com/irrigation-product/water-management-software/immsr> [Accessed 26/5, 2017].

LOWE, D., 2014. *Java for Dummies*. 4 edn. Hoboken: John Wiley & Sons.

MATHWORKS, 2017-last update, *MATLAB Compiler SDK – MATLAB*, Available: <https://uk.mathworks.com/products/matlab-compiler-sdk.html> [Accessed 10/8, 2017].

PEDREGAL, P., 2004. *Introduction to Optimization*. 4 edn. New York: Springer Science and Business.

SOMERVILLE, I., 2006. *Software Engineering*. 9 edn. Boston: Addison-Wesley.

Suttons Consumer Products, 2017-last update, *Suttons Fruit & Veg Planting Guide for iPhone/iPad*, Available: <http://hub.suttons.co.uk/gardening-advice/apps> [Accessed 26/5, 2017].

THE WEATHER COMPANY, 2017-last update, *API | Weather Underground*, Available: <https://www.wunderground.com/weather/api/?MR=1> [Last Accessed 30/8, 2017].

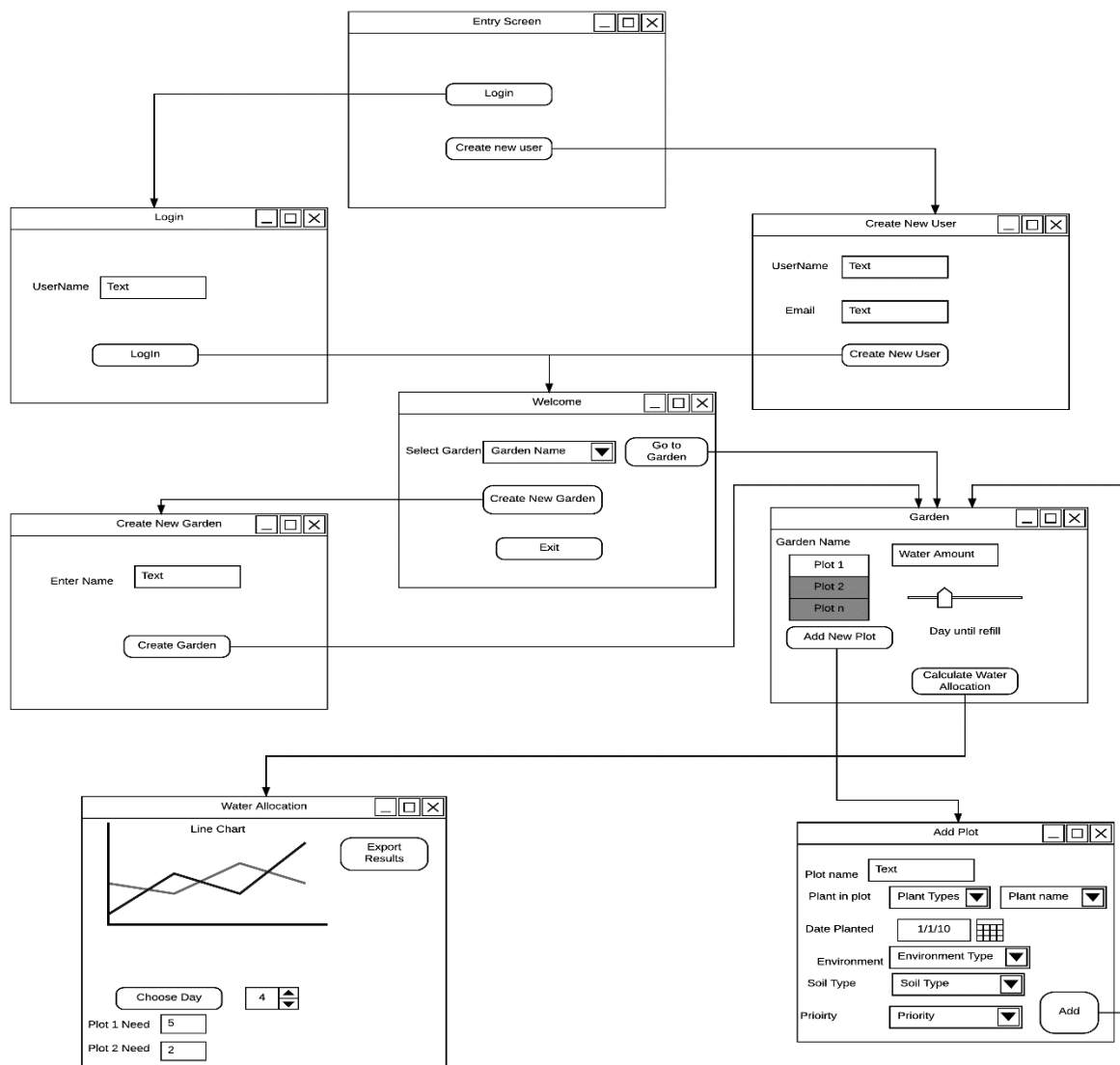
WANG, Z., YANG, J., DENG, X. and LAN, X., 2015. Optimal Water Resources Allocation under the Constraint of Land Use in the Heihe River basin China. *Sustainability*, **7**, pp. 1558.

YE, Y., 2015-last update, Interior Point Algorithms [Homepage of Stanford University], [Online]. Available: <https://web.stanford.edu/class/msande310/lecture11.pdf> [15/07, 2017].



## Appendix 3 – User Interface Design

Showing initial design done to outline structure of User Interface. That was developed through the prototypes.



## Appendix 4 – Database Design

