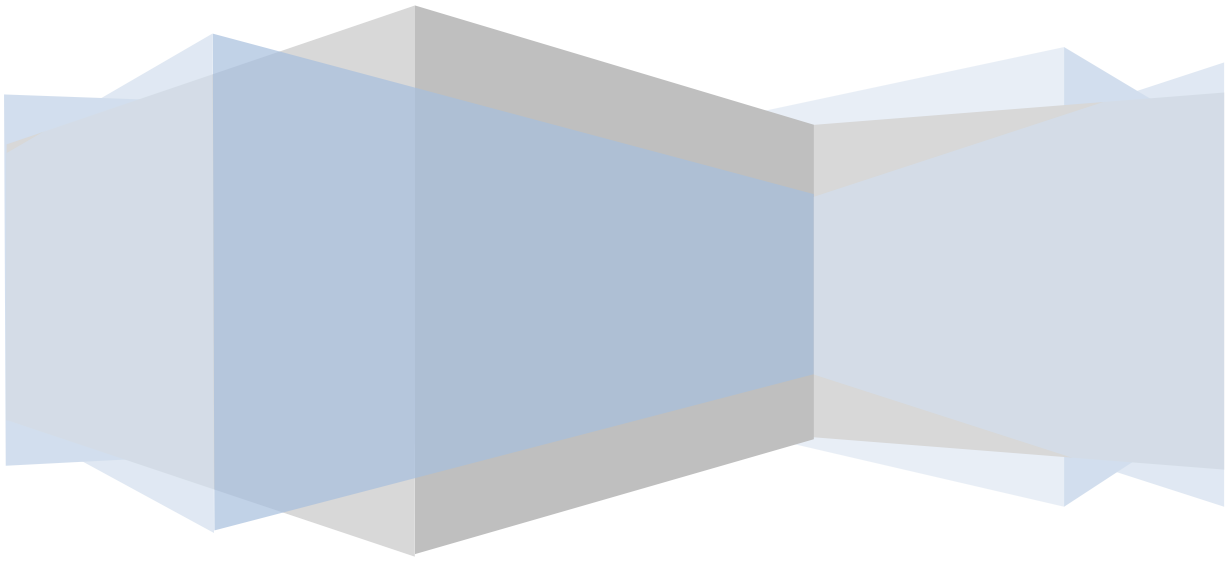
COM814: Project 2016

Dissertation

School of Computing & Information Engineering

Edward Mullan

Plantree; An app for planning and managing woodland plantations
Supervisor: Shuai Zhang
Second Marker: Giuseppe Trombino
1st Sept 2016



Contents

Plagiarism Statement 4
Acknowledgements 4
Abstract 4
1. Introduction 5
1.1 Introduction 5
1.2 The Problem 7
1.3 The Solution 9
2. Professional Issues 11
2.1 Existing Solutions 11
2.2 Current Use of Apps in Agriculture (case study) 14
2.3 Business Case 15
2.4 Risk analysis 16
2.4.1 Controlling risk through Agile Methodology 16
2.4.2 Controlling Risk with version Control 17
2.5 Ethical Considerations 17
3. Analysis 18
3.1 Questionnaire 18
3.2 User Stories 22
3.3 Platform 23
3.4 Functional and non-functional requirements 25
4. Design 26
4.1 Aesthetic 26
4.2 Interface 27
4.2.1 Activity layout 28
4.2.2 Error handling 29
4.3 Application Architecture 31
4.3.1 Activity lifecycle 31
4.4 Database Design 35
4.5 Design Summary 36
5. Implementation 36
5.1 Coding standards 36
5.2 Platform IDE 37
5.3 Version Control 38
5.4 User Guide 38
5.5 Home button 39

5.6 Google Maps API 39
5.7 Creating The Farm 40
5.8 Creating, Measuring, and Storing Plantations 41
5.8.1 Polygon Area 42
5.8.2 Encoding and Decoding polygons and viewing the plantations 43
5.8.3 Displaying the plantations in a list view 43
5.8.4 Passing plantation data to another activity 44
5.9 Adding trees to plantation 45
5.10 Database Implementation 46
5.10.1 Relational integrity in SQLite 47
5.9.2 Inserting data and updating tables 48
5.11 Future implementation 48
5.12 Implementation summary 48
6. Testing and Evaluation 48
6.1 introduction 48
6.2 Debugging 49
6.3 Resolving failed tests 49
6.4 Different devices 50
6.5 Testing 50
6.6 Focus group feedback 52
7. Conclusion and Recommendations 53
7.1 Appraisal 53
7.2 Final Summary 54
References 54
Appendices 59

Table of figures

Fig 1. Rich picture of problem 8
Fig 2. Rich picture of proposed solution 11
Fig 3 Screenshot of Fields Area Measure by Studio NoFrame 12
Fig. 4 Screenshots of Agri Precision by Leonardo Om 12
Fig 5. Screenshots of Farm Manager by TEICM-Agricultural lab of mobile phone apps EGEKT 13
Fig 6: Appearance of app 26
Fig 7: Final design for home screen 27
Fig 8.1: Error handling and displaying messages 29
Fig 8.2: Error handling and displaying messages 30

Fig 8.3: Error handling and displaying messages 30
Fig 9: Lifecycle of an activity 33
Fig 10: Application map 34
Fig 11: Plantree database ER diagram 36
Fig 12: Initialising GIT 38
Fig 13: Example of user guide popup. 39
Fig 15: Pre existing XML file within the project for the Google Maps API information 40
Fig 16: onMapLongCLick method from the CreateFarm class. 40
Fig 17: Method to draw plantation 42
Fig 18: Clearing the plantation 42
Fig 19: Listing plantations 44
Fig 20: An intent with a bundle of data 44
Fig 21: Receiving an Intent and Bundle 45
Fig 22: Evaluating number of trees planted 45
Fig 23: DbHelper create farm query 46
Fig 24: DbHelper onCreate using queries 46
Fig 25: Database tables viewed in DB Browser upon creation 46
Fig 26: Creating table with composite primary key 47

Plagiarism Statement

"When submitting your work you are agreeing with the following statement: I declare that this is all my own work and that any material I have referred to has been accurately referenced. I have read the University's policy on plagiarism and understand the definition of plagiarism. If it is shown that material has been plagiarised, or I have otherwise attempted to obtain an unfair advantage for myself or others, I understand that I may face sanctions in accordance with the policies and procedures of the University. A mark of zero may be awarded and the reason for that mark will be recorded on my file."

Signed –

Edward Mullan

Acknowledgements

I would like to thank my supervisors Shuai Zhang and Giuseppe Trombino for all their help and support on this project.

Abstract

The UK and Ireland are amongst the least wooded areas in Europe. With recent flooding events being exacerbated by water run-off from agricultural land, there is a growing understanding of the importance played by trees in agriculture. European Common Agricultural Policy has established incentives for farmers to plant trees in order to create a more balanced ecosystem and reverse some

of the damage done by intensive farming methods in recent history. Unfortunately, there is a lack of knowledge about planting trees and their benefits.

This dissertation describes the work undertaken in creating an app that will allow users to plan and manage the creation of woodlands.

'Plantree' was developed as an Android app developed for smartphone and tablets. Design and user requirements were identified by administering a questionnaire to a group involved in agriculture and horticulture.

A prototype app was developed and implemented, and the was tested by the same focus group and recommendations for future development of the application were made. The process of analysis, design, implementation, testing and final evaluation are described.

1. Introduction

1.1 Introduction

The UK has been cited as of the least wooded countries in Europe. Woodland comprises just 13% of the geographical area of the UK compared to 37% for the rest of the EU. Northern Ireland has the least woodland in the UK at just 6.5% of total geographical area (Woodland Trust 2011). The republic of Ireland fares slightly better than Northern Ireland at 10% coverage, (Wade 2012) but this figure is at odds with the general perception of Ireland having an unspoiled green landscape. This is the legacy of decades of attempted increases in efficiency in output and land use. Woodland coverage was at its lowest at the start of the twentieth century and plantation of publicly owned conifer forest in both countries has seen the figure rise from single figures in both countries, (Woodland Trust 2011) but conifers alone do not foster biodiversity, and do not permit other plants or trees to prosper in their forests. The perceived importance of trees to our ecosystem has increased hugely in the last decade in the UK and Ireland. There are several reasons for this, primarily the availability of grants under the EU Common Agricultural Policy and under various locally administered schemes has allowed farmers and landowners to receive remuneration for trees planted in marginal areas where more traditional forms or agriculture are impossible, or where existing woodlands need replacement, There are various schemes throughout the UK which can help farmers and landowners cover up to 75% of the costs of this activity at commercial rates, although if work is carried out by the claimant the amount awarded does not decrease. (Available grants, Woodlands.co.uk 2016). There are further grants available from the same sources for the ongoing maintenance of trees planted.

There has also been an increase in the awareness of how deforestation and intensive farming techniques have damaged the eco system by encouraging monocultures of crops and the most 'efficient' use of land by eradicating anything marginal ground at boundaries.

Recent flooding events in the UK and Ireland have demonstrated the need for the planting of more trees in order to prevent or lessen the amount of run-off of water from agricultural land, as the clearance of trees from higher ground has been linked to worsening flood

events at lower levels. Ground beneath trees has been shown to absorb 67 times more water than that beneath grass (Marshall et al. 2013). Trees also prevent the run off of sediment with water from land which in turn helps to keep water courses clear and allows soil on agricultural land to retain its nutrients. (Fazio 2010) Traditionally farmers have been encouraged to clear land in order to avail of the 'Single Farm Payment' and other subsidies, which require land to be in 'agricultural condition', (i.e. bare of trees and other vegetation), regardless of the suitability of that land to be used for agricultural purposes (Monbiot, 20/12/2015). This has led to an incorrect perception of what an ideal landscape should look like, open countryside should not be open and green, but rather should be broken up regularly by pockets of woodland. The detrimental effect on wildlife has been seen with the poor success rate of a project to reintroduce eagles to County Donegal in Ireland. The loss of tree cover has reduced habitats for prey which in turn reduces the amount of food available for indigenous predators. (Woodworth 2016).

Recent outbreaks of disease have decimated woodlands and forests in the UK and Ireland, with diseases such as ash dieback and Dutch Elm disease being of particular concern, a new threat contributing rapidly to generations of elimination of woodland by human activity. Ash dieback in particular has been gathering momentum, it is caused by a fungus and renders trees weaker to attack and is usually fatal for the trees it infects. Recent studies have shown the rate of this disease to be increasing at an alarming rate with the number of infected sites climbing exponentially. As ash trees make up approximately 13% of trees in the UK, this has already become a cause for urgent concern (Carrington 2012). Industrial conifer forests in across the UK and Ireland have been seriously damaged by the Phytophthorum ramorum pathogen, a disease which can kill many types of tree relatively quickly but thrives in the types of conifer most commonly planted in state owned forests. Attempts to contain the disease have led to hundreds of hectares of forest being felled in Northern Ireland alone. (Stewart 2103)

There is now growing awareness of the value of trees for agriculture. Trees and shrubs have been shown to increase the amount of nitrogen in the soil, and having trees on the same land as crops, although inconvenient for machinery, has been demonstrated through projects in developing countries to improve the productivity of the soil. (Van Vark 2013). Grants have been made available to help landowners cover the outlay of planting trees and the potential maintenance costs. Native species are encouraged as is variety, in order to avoid a monoculture (which does not foster biodiversity and leaves tree populations vulnerable to disease as discussed above). Trees can be harvested for firewood or biomass within relatively short period of time which can provide a further financial incentive to landowners, and trees can act as poor weather shelter for animals, reducing the need to provide buildings for use in poor weather, and by products such as woodchip can be used as bedding instead of having to buy in straw bedding.

There is also greater awareness amongst the public of the importance of wildlife and biodiversity. Organisations such as the RSPB, WWF, The Woodland Trust, and Wildlife Trusts, actively involve the public in conservation efforts. Creating new habitats and rejuvenating areas which have fallen foul of intensive agriculture or development. Tree planting is one area that is particularly encouraged, although the process of starting a tree plantation, large or small, can be daunting, and mistakes in planning and maintenance can lead to poor results in terms of growth and success of the planation. Whilst there are plenty of apps, games and online resources aimed at general farming, there are relatively few resources available to those wishing to plant trees. This dissertation will set out the problems faced by those wishing to plant trees on their land, with a solution being the design and realisation of an application that will assist users to plan, plant, and maintain their trees.

This dissertation aims to provide a solution to assist users in the creation of new woodland areas on their land, to provide the necessary information to encourage this activity and improve the confidence of the novice arborist.

The dissertation will include research into what solutions already exist, as well as questionnaire led research into what the market requires, using opinions from those who work in the field of forestry and plantation as well as potential users who may be interested in planting trees or who have already done so.

There will description design concepts for the user interface in terms of layout and functionality as well as graphical design considerations. There will be a discussion of the architecture of the application and how it will work on a software level, and there will be an outline of the testing and evaluation process.

1.2 The Problem

The problem to be addressed can be stated as follows;

"The importance of trees to our ecosystem is becoming more widely understood. There is a need for a straightforward application that would help novice planters plan, plant, and care for their trees."

Intensive farming methods have become the norm in Europe in the 20th and 21st centuries. Demand from supermarkets and downward pressure on overall food prices has meant that many farmers have had to take measures to extract the maximum yield of crop from their land. This has taken the form of eradicating all but the most necessary land boundaries and marginal areas, hedgerows and wooded areas have been decimated in favour of monocultures of a particular crop. Larger farms tend to prosper where crops are involved

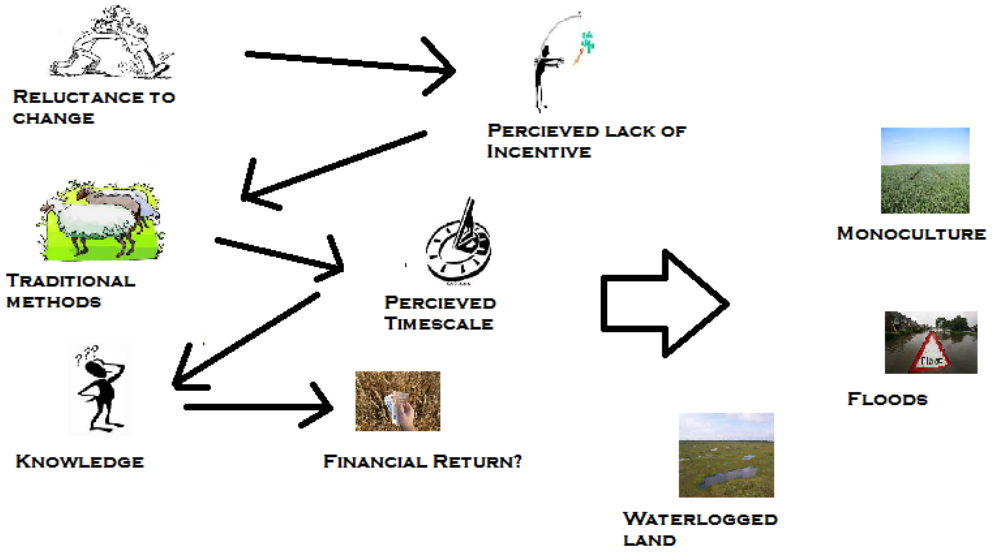
and these can become like deserts in terms of ecology. In the UK and Ireland, livestock farmers have been encouraged to keep land bare in order to avail of subsidies, (Monbiot, 20/12/2015) Increased development of suburban areas has led to woodlands being eradicated in favour of housing developments. The ecological impact of these changes over generations, combined with the effects of climate change in recent years has been demonstrative of the need for a rejuvenation of woodlands.

Whilst there are no overt barriers to this, (trees are not subject to planning restrictions) there are several reasons why landowners may not engage in tree planting. Farmers especially may not generally not view trees and woodland management as part their remit. There may be a perception that planting trees offers no financial return, (Osmond 2012), however there are ways in which trees can be beneficial to agriculture and can improve the overall profitability of a farm. Confidence is a factor that would effect even those willing to plant. Planting can be a long term commitment, there is a perception that a tree can take decades to reach maturity, and in that time the space it occupies cannot be used for any other purpose. It can be difficult to decide the type of tree to plant and for what purpose. It can also be difficult to justify if one is unsure of the benefits.

For the general public, who do not have large areas of land at their disposal, planting a tree can have some daunting considerations. Small gardens can be overwhelmed, light can be blocked out, and roots can damage foundations and sewers. These can put people off planting trees in their gardens.

The following rich picture diagram demonstrate the factors contributing to the problem,

Fig 1. Rich picture of problem



1.3 The Solution

The proposed solution to the problem outlined above is a mobile App that will help users to easily visualise, plan, and execute the creation of a woodland plantation. The key points of the solution are as follows;

Knowledge:

The main factor is knowledge, knowing where to get saplings, when to plant, how to identify trees, what trees flourish in which conditions? Should tree varieties be mixed? How far apart should they be spaced? How to care for trees? Not knowing the answer to any one of these questions can cause serious problems for a planation of trees. Planting at the wrong time can lead to saplings failing to flourish, trees appropriate to the locality should be planted to avoid the introduction of invasive species or disease. Some trees are better suited to wet ground, some do better at higher altitude, some grow faster for quicker harvesting of firewood or other by products, some have a negative effect on soil alkalinity, some block out too much light. Planting too close together will cause retardation of growth and not caring for trees will lead to misshapen trunks and low height or early death. The app should address all these concerns and give clear information and instruction.

Cost:

Individual saplings are rarely expensive, but to plant a large area can run into hundreds or thousands of pounds. There are various grants available to cover these costs as well as incentives to dedicate land to forestry. The app could contain information and links to sites where funding applications can be made.

Planning

One of the main uses of the app would be to help users plan an area for planting trees, helping to visualise what the plantation may look like can be a good motivator, also this function would ensure best practice when spacing trees and separating varieties, allowing a quick reference in the future and reminders for events such as pruning or clearing of vegetation.

Target user:

Although farmers may be primarily the group who would engage in tree planting under the various schemes in place, there may also be other potential users who own areas of land who wish to find out more about tree planting on their land. They may not be actively engaged in agriculture. The process of creating a large planation may be perceived to be beyond their means or time constraints may hamper them. The app must encompass anything from a few trees in the corner of a garden to a full field or multiple field area.

The use of Apps and mobile devices has exploded, Ofcom's 2016 Communications Market Report published in July of 2015 illustrates this. It describes a 10% increase in the amount of adults using a smartphone in the UK since 2014 bringing the figure to 71% of all adults, with the smartphone being on a par with the laptop as the most valued internet enabled device in UK households. (Ofcom 2015 p10)

Whilst older generations may still be reluctant to use mobile devices, their use amongst those aged 55-64, has more than doubled (19% in 2012 to 50% in 2015), whilst their use by those aged 65 and over has more than tripled (5% in 2012 to 18% in 2015). (Ofcom 2015 p69). When this is taken into account it is clear that the mobile app market offers the most potential for development, added to this is the ageing population of farmers in the UK and Ireland, it is envisaged that the development of smart agricultural technology should play an important role in encouraging younger generations to get involved in agriculture and to see it as a viable career in the 21st Century.

Further care:

Many people plant trees without realising that like plants, they require maintenance and care. The app should include simple instructions on how to care for young trees as they grow.

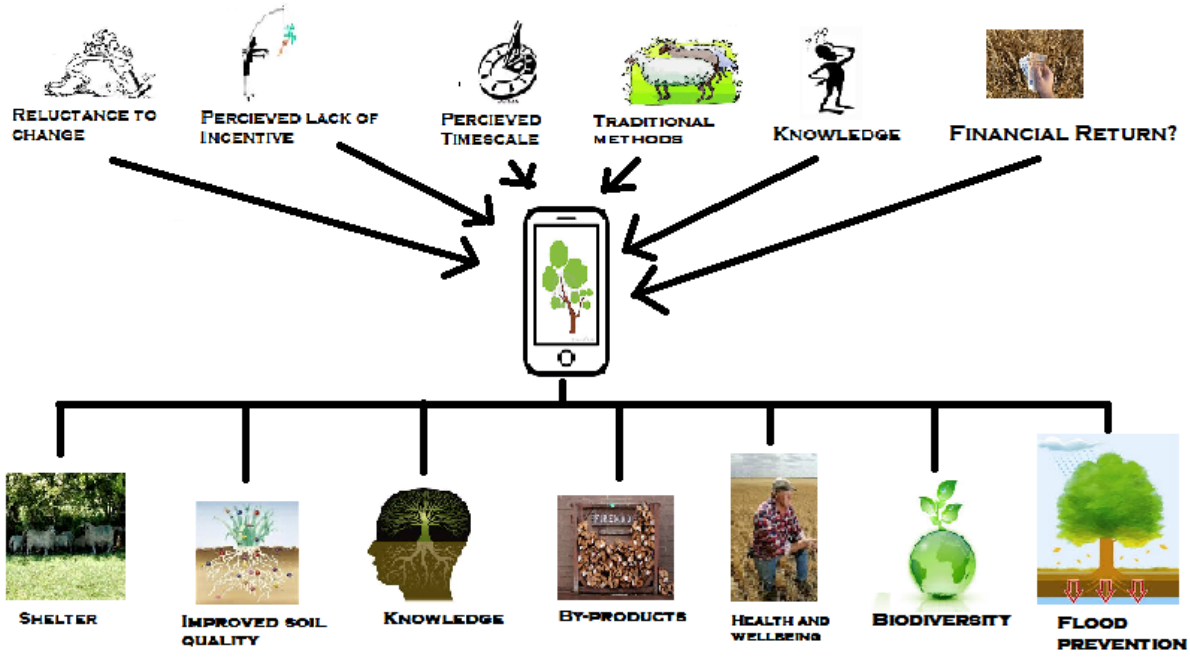
Interest:

One thing that has become more prevalent in the last few years is an interest in trees as a pastime. They have associations with mythology, outdoor pursuits, nature watching and conservation, as well as engaging people in outdoor activities beneficial to health. The act of planting, maintaining and eventually harvesting trees for wood can be deeply enjoyable and

invigorating. The app should holistically address these issues to encourage tree planting amongst users. Features such as identifying trees and information on their characteristics should be included.

The following is a rich picture diagram of the proposed solution:

Fig 2. Rich picture of proposed solution



2. Professional Issues

2.1 Existing Solutions

An investigation of the current marketplace of android mobile apps found that there are currently no apps specifically targeted at the creation of woodlands or the practicalities of planting trees in general. Having said this, there are some agricultural apps that incorporate some useful features that may be utilised in the proposed solution.

1. GPS Fields Area Measure by Studio Noframe

This is a measurement app for determining the area of any plot of land from the size of a garden to an entire farm. It incorporates GPS for automatic measuring which allows users to walk/drive a boundary or alternately to define an area on the device screen using flags.

Fig 3 Screenshot of Fields Area Measure by Studio NoFrame

Pros:

- It is quite simple to use and includes a tutorial for first time user.
- Measuring is accurate and measurements can be saved to the device
- Interface looks professional.

Cons:

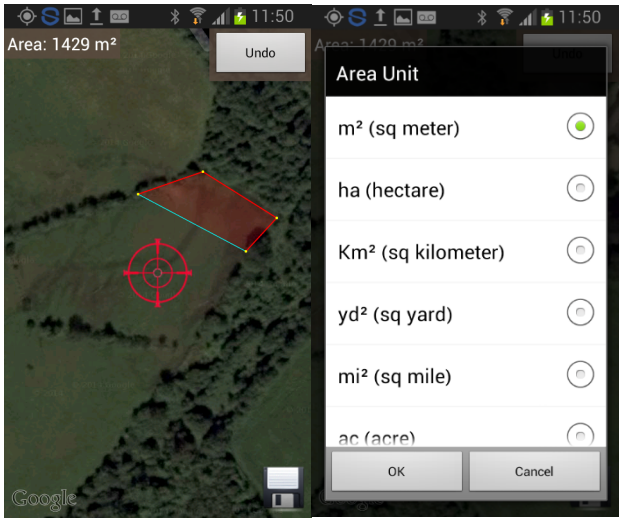
- The display is quite cluttered with icons, mostly from the google maps API that it uses, but the action bar also seems overloaded with icons. The effect of this is to make navigation difficult, it was not immediately obvious where to find settings for units of measurement.
- Screen space is further restricted by the use of an advert banner across the bottom of the screen on the free version. I may have been more prudent to employ a full screen advert that disappears completely once dismissed.

Overall this was a useful app and it is envisaged that the area measurement function may be a useful addition to this project

2. Agri Precision by Leonardo Om

This app uses GPS or manual point and click to define the area required by the user in the same way as the previous example.

Fig. 4 Screenshots of Agri Precision by Leonardo Om



Pros:

- The interface is very simple and intuitive.
- The manual measuring capability is much smoother than the previous example and the app has many more functions, allowing the definition of plots within a larger area and determining the contents of each area.
- No adverts.

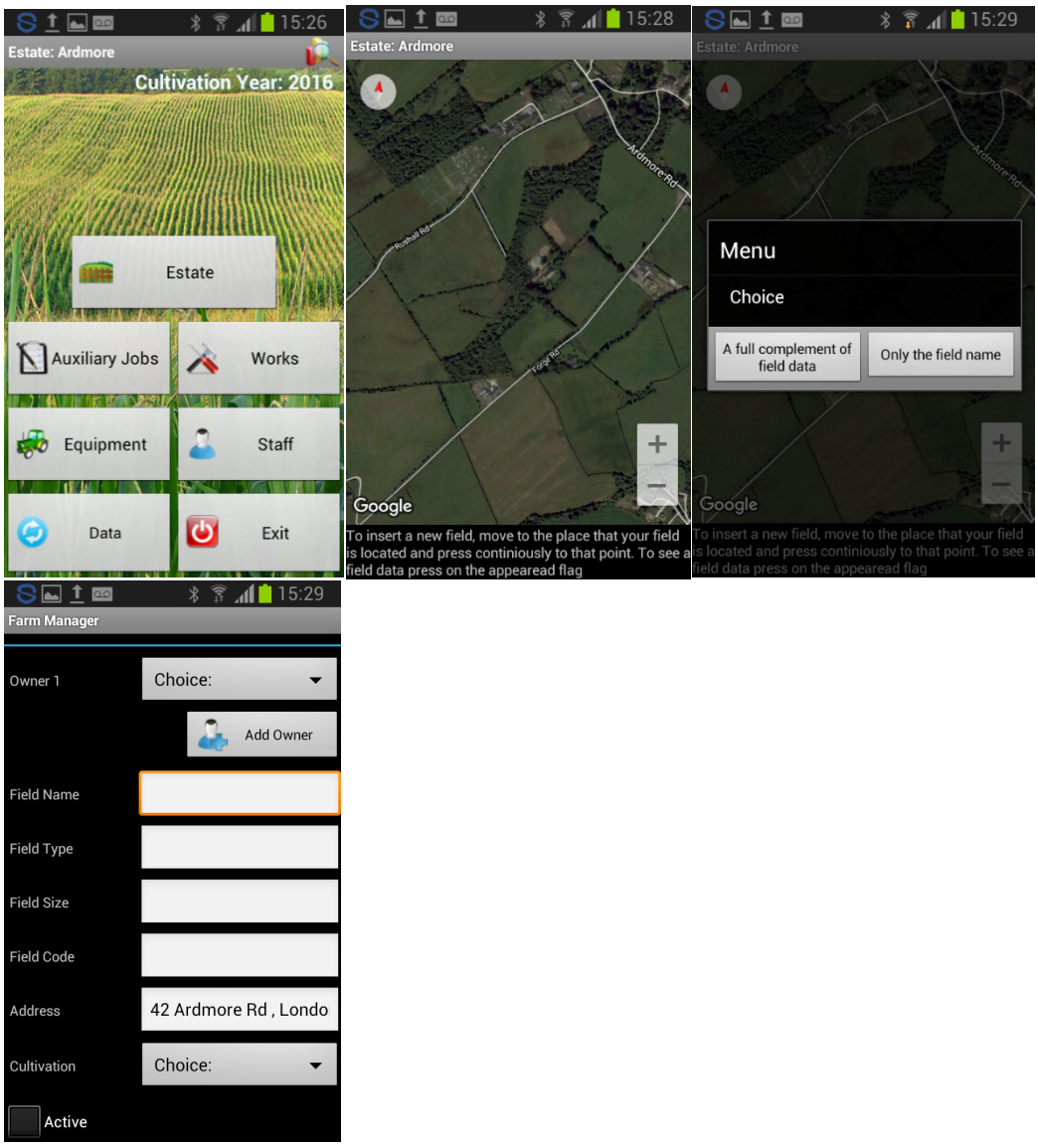
Cons:

- No user creation option
- Most options require in-app purchase.

3. Farm Manager by TEICM-Agricultural lab of mobile phone apps EGEKT

This is a more complete farm management app which includes several useful functions including works schedules, field planning, staff management, and equipment management.

Fig 5. Screenshots of Farm Manager by TEICM-Agricultural lab of mobile phone apps EGEKT



Pros:

- This app looks professional and allows you to select fields on a map and input data.
- It recognises postal addresses.

Cons:

- Does not allow measurement of fields or definition of boundaries, merely places a flag where you tap the screen and identifies this as the field. The field flag was not visible when trying to review the saved selection.
- No security options. (Username, password etc.).
- It does not appear that work reminders are alerted to the user without entering the app.
- There is no user guide and it appears that most of the functions on the app exist to record data and not to do anything with it.

All these apps demonstrate that the required functionality to provide a solution to this problem exists within the android environment. They also demonstrate that there are relatively few truly useful agriculture apps in the android marketplace. With this in mind it is surmised that a woodland plantation app would be filling a gap in the market for those who wish to get involved in tree planting.

2.2 Current Use of Apps in Agriculture (case study)

As in almost all areas of life, the use of mobile apps in agriculture is burgeoning. This has been recognised and there are various practical apps and games in the market place already. One example of the potential market for this type of app can be seen in the experience of Irish Developers Tommie Slattery and Ryan Garner. (Worral 2013) They produced a portfolio of applications called Smart Farm Apps to help farmers keep track of data relating to their business and the industry in general. They include;

- A dairy app that helps keep track of animal records, milking schedules, and other dairy related functions.
- A grass rotation app for silage production.
- A cattle breeding app that helps keep track of animal fertility over the year.
- A milk solids app which helps keep track of milk prices and plan production.

Their apps have been downloaded internationally in 30 countries and further growth is expected. The driver for their ideas was the knowledge that a lot of farmers keep track of information using traditional means such as diaries and when computers were used, it tended to be by entering data on a pc type machine. The Mobile phone was seen as a better option because farmers tend to have them on their person most of the time and thus there is less chance of something being overlooked and forgotten about.

These and other apps are part of an increasingly technical agricultural industry where the use of data and information gathering can increase productivity and decrease cost for the farmers. One very salient point is that although farmers are increasingly using technology, they do not want automation or decisions to be made by machines, what is required is help in decision making so the right choices can be made. (Vidal 2015).

With this in mind any solution to the need for a woodland planning app should be unobtrusive and easy to use. It should meet a defined need and actually make life easier for those who use it

2.3 Business Case

At present there are no apps on the market that are targeted at planning tree planation. Farming apps themselves have been becoming more widespread in recent years, and for the most part these tend to aimed at crop management. Their increasing use points to an increased acceptance of apps and websites as tools in agriculture. Web apps and associated technology can play a part in increasing the attractiveness of agriculture and forestry to younger people. This is important as there has been a steady increase in the average age of farmers since 2000. (National Statistics 2012). A simple app to aid visualisation and planning of woodlands is hypothesised to encouraged participation of members of farming families who would not be able to participate in more hazardous areas of agriculture. Tree planting cannot easily be carried out by machine and is therefore usually done by hand. This forms an opportunity for exercise and a social activity if family members are involved. It is relatively simple and safe to participate in.

Planting trees can foster a feeling of greater affinity with the ecosystem, and allow those who take part to make a real difference to the state of their land and how it supports agriculture and their communities.

Planting trees is recognised to be of benefit to agriculture in several ways;

- By helping farmland to retain water, which aids transportation of nutrients to crops. This is achieved in two ways:
    1. The windbreak effect of trees reduces air movement allowing humidity to remain in the air above crops which reduces evaporation from the soil. (Woodland trust)

    2. Trees allow water to penetrate deeper into the soil via root networks where it can be held for longer, resisting run off and evaporation (Marshall et al 2013).

- By providing shelter for grazing animals, reducing the need for buildings
- By providing habitat for pollinating species, thereby increasing crop yields
- Other financial benefits can be achieved when trees are planted in ground that is otherwise difficult to cultivate, thus returning a financial benefit through either grant funding or via using trees as a crop for fruit or biomass from otherwise unproductive land.
- Awareness of wood as a biofuel is increasing, and the use of wood or wood pellets as a fuel for home heating can insulate farms from the unpredictability of fossil fuel prices. A small plantation can provide fuel for many years and be ready for harvesting in less than a decade depending on the type of tree, PlanTree aims to help beginners approach this task and mange it through to completion. In this sense the App could conceivably help users to produce a saleable commodity in firewood or to reduce their reliance on fossil fuels for heating purposes.

2.4 Risk analysis

With the undertaking of a project such as this, there will be many potential risks that should be mitigated where possible. There follows a risk assessment to define the risks and the means of controlling them.

Table 1 Risk analysis

Risk -Technical considerations	Contingency
Corruption / loss of data	Regular backups to secondary storage/Version control
Theft / loss / malfunction of computer	Organisation of access to local UU campus labs (Magee)/version control
Loss of broadband service	Use of traditional research methods and use of campus labs
Third party / web hosting issues	Use of any web hosting services (if required) will be done on the basis of research into reliability.
Software issues	Management of delivery timescale to ensure early delivery so any issues may be overcome.
Coding issues	Use of efficient code and testing at every stage. Use of test driven development in order to develop robust code.

Risk- Project management considerations	Contingency
Developer Illness/Personal issues	Agile methodology to ensure ahead of schedule delivery
Timescale overrun	Regular meetings with project supervisor, setting achievable goals with timescales.
Other commitments / workload	Formulation of study timetables with clear timeslots for project and other study

2.4.1 Controlling risk through Agile Methodology

The use of agile methodology in the project will be an important factor in meeting deadlines and targets to ensure delivery. This methodology entails constant re-examination of the project by working in short cycles or iterations. This allows any changes to be made to the project to overcome issues and obstacles. Due to the fact that the project will be carried out by a novice developer, there will be a lot of knowledge development throughout its course, and there may be factors within it that will require reassessment as the developers skill and familiarity with the chosen programming environment increase. For this reason it is important that the project be managed in a flexible

manner, as opposed to a more rigid methodology such as the traditional 'waterfall' methodology. Waterfall methodology does not allow changes to the original design concept and therefore would not allow much flexibility to overcome obstacles.

The main feature of agile methodology is the 'sprint'. This is a relatively short period of time where a particular issue or problem is addressed and overcome. Sprints are punctuated by 'scrums' where the development team (in this case the project tutor and myself) will discuss and assess each iteration and the steps to follow.

Another useful feature of agile methodology is the time management process it uses. Tasks are grouped according to the time they are expected to take, not necessarily in order of importance to the project as a whole. The purpose of this is to increase the predictability of the timescale for development.

The main advantage of agile is that it allows suitable time for contingency plans to be put in place should any changes in technology occur, such as updates of the operating system in use or other changes beyond the control of the project team (Abraham 2015 p.33). This methodology also allows regular feedback to the project leader so problems can be addressed rather than allowing them to snowball out of control and threaten the whole project. If insurmountable problems with a particular aspect of the program are met, then there will be ample time to reassess and change direction. All these concepts together allow the project to remain relevant throughout its development. (agilemethodology.org 2008)

2.4.2 Controlling Risk with version Control

Version control is the most important means of controlling the risks that may threaten the project's completion in terms of technical issues. It will allow complete backups and rollbacks and is much easier to manage than simply making copies to external storage. It is essential that some form of version control be employed in order to mitigate against loss of data or unforeseen circumstances. For example, in the unlikely event of accidental deletion of some or all of the project, previous versions can be recovered easily if version control is used, furthermore, should primary device being used to build the app become inoperative or be stolen, the most recent version of the project can be recovered if a remote service is employed in addition to the local 'node'. It is likely that a service such as GIT will be employed. GIT is chosen for the following reasons:

- Ease of use – git can be controlled easily with bash scripting and user advice is readily available from a variety of sources.
- Open source – Originally developed to enable version control for the development of the Linux system, it is free to use and fits the philosophy of that OS.
- Remote servers – GIT HUB allows the storage of versions remotely, data stored here will be public unless a paid account is taken. Fortunately, there is also a GIT LAB server available to University of Ulster students free of charge, which will be a private repository.
- Distributed development – Although this particular project will be an individual effort, GIT is based on the principle of distribution, i.e., each developer working on a project has their own branch repository with a history of commits. This means that there is no dependence on a central repository (Atlassian), which further mitigates risk.
- Scalability – GIT can also be easily scaled with the addition of further developer branches, existing versions can simply be cloned for new developers who can then start to work on their own version, building on what has already been done with a full view of the project.

2.5 Ethical Considerations

This project is expected to be classified as category Z by the University Ethics Committee. The research methods used are non-invasive and do not involve children or vulnerable adults. Feedback

and testing will be carried out by myself or adult volunteers and should not involve any risk to participants.

3. Analysis

The following chapter will detail how the final concept was reached in terms of platform, system requirements, and functions of the finished solution.

3.1 Questionnaire

In order to fully identify the requirements for the app, a research questionnaire was formulated to ascertain the feasibility of the proposed app and to gain some user input into the features and functions it should contain. This was distributed to a focus group of ten people. Two of the people are involved in a garden landscaping business which include the sale of and planting of trees. Two are involved in the running of a nursery and garden centre which sells large amounts of trees for planting on agricultural land and in gardens. The rest are people who have planted trees on their farmland or in their gardens or who intend to do so in the future.

The questionnaire was broken down into 9 sections:

1. Section one was designed solely to establish that the participant was suitable to be involved.
2. Section two was designed to identify whether there was a market for an app to assist in the planning of plantations and woods.
3. Section three was designed to identify a suitable name for the app from five prospective choices.
   It also allowed the participant to suggest their own (no suggestions were forwarded in this case)
4. Section four was designed to identify the suitability of various functions for a tree planting app.
5. Section five was designed to identify the relevance for certain areas to be covered by the app as knowledge tools
6. Section six was designed to identify suitable technical features for the app (such as GPS tracking and website links) The participant was also allowed to submit their own suggestions, (again none were forwarded)
7. Section seven dealt with what data should be stored and transferred assuming the app was to have user accounts.
8. Section eight dealt with the appearance of the app presenting participants with 6 prospective background types / colour schemes.
9. Section nine dealt with testing and asked participants if they would like to take part in future test of the product.

The information gathered from the questionnaires is shown in the tables below

Table 2
Questionnaire Section 2

Question	Response
Planting trees benefits agriculture	8 /10 participants agree

	2/10 don't know
Planting trees benefits biodiversity	10/10 participants strongly agree
Planting trees reduces water run-off from land	7/10 participants agree 1/10 strongly agrees 2/10 don't know
Planting trees is a big commitment	3/10 participants neutral 4/10 disagree 1/10 don't know
Planting trees is of no financial benefit	8/10 participants disagree 2 /10 don't know
Information on planting trees is difficult to find	5/10 participants neutral 3/10 disagree 2/10 don't know
Planting trees can help prevent flooding	1/10 participants strongly agree 5/10 agree 2/10 don't know 1/10 disagree
There is a market for apps and resources to help people plant trees	5/10 participants strongly agree 3/10 agree 2/10 don't know

Table 3
Questionnaire Section 3 (app name)

Name	Response
PlanTree	9/10 Participants chose very suitable 1/10 quite suitable
Forestr	7/10 Participants chose quite suitable 3/10 not very suitable
ArborApp	6/10 Participants chose not at all suitable 4/10 don't know
WoodU	5/10 Participants chose very suitable 3/10 quite suitable 1/10 not very suitable
Tree Oracle	3/10 Participants chose quite suitable 3/10 neither suitable nor unsuitable 4/10 not very suitable
Participant suggestions	No suggestions

The name 'PlanTree' was deemed the most suitable so this was chosen as the name of the app. One word incorporating Plan, Plant and Tree.

Table 4
Section 4 (functions)

Function	Response
GPS/geo-fencing for planning plantation areas	8/10 participants chose very important 2/10 quite important
Alerts for due maintenance activity	10/10 Participants chose very important
Optimum /maximum tree numbers calculation	7/10 Participants chose very important 2/10 quite important 1/10 not very important
Capable of storing multiple plantation areas	7/10 chose quite important 2/10 neither important nor unimportant 1/10 don't know
Tree counter (user)	7/10 participants chose very important 2/10 don't know 1/10 not very important
Tree counter (all users)	8/10 participants chose not important at all 2/10 not very important
Links to funding websites	4/10 participants chose not very important 6/10 quite important
Disease and pest geo-flagging	7/10 participants chose quite important 2/10 don't know 1/10 neither important nor unimportant
Image geo-tagging	6/10 participants chose quite important 4/10 neither important nor unimportant
GPS record of undertaken maintenance work	7/10 participants chose quite important 1/10 very important 1/10 not very important 1/10 not important at all
Choice of measurement units (imperial/ metric)	5/10 participant chose quite important 4/10 neither important nor unimportant 1/10 don't know
Social media interface	7/10 participants chose not at all important 2/10 neither important nor unimportant 1/10 quite important

All of the above features were seen to be suitable for the finished app except for a tree counter for all users and a social media interface. These were both seen as unnecessary by the participants. Choice of measurement units was seen as quite important by 50% of the participants, but as most of the calculations involved will be carried out by the app itself this will not be included.

Table 5
Section 5 Knowledge tools

Knowledge area	Response
How to choose an area to plant	5/10 participants chose neither suitable nor unsuitable 3/10 quite suitable 2/10 don't know
Safety tips	7/10 participants chose quite suitable 3/10 neither suitable nor unsuitable
How to plant a sapling	9/10 participants chose very suitable 1/10 quite suitable
When to plant	6/10 participants chose very suitable 4/10 quite suitable
Plantation type selector (eg orchard, woodland, biomass)	5/10 participants chose neither suitable nor unsuitable 4/10 not very suitable 1/10 don't know
Species info	10/10 participants chose very suitable
Pruning guide	8/10 participants chose very suitable 2/10 quite suitable
Firewood suitability guide	4/10 participants chose very suitable 3/10 quite suitable 2/10 not very suitable

All the knowledge tools proposed above were seen to be suitable except the plantation type selector.

Table 6
Section 6 Technical features

Feature	Response
User account with log in and password	9/10 participants chose very suitable 1/10 not very suitable
GPS	7/10 participants chose very suitable 2/10 don't know 1/10 quite suitable
Reviews/feedback section	5/10 participants chose quite suitable 3/10 don't know 1/10 neither suitable nor unsuitable 1/10 very suitable
Accessible on multiple devices	6/10 quite suitable 4/10 participants chose neither suitable nor unsuitable
App designer and maintenance contact	8/10 participants chose neither suitable nor unsuitable 2/10 don't know
Suggested features	None suggested

All of the technical features above were seen as suitable except an app designer and maintenance contact. This was seen to be unnecessary if there was to be a feedback section.

Table 7
Section 7 Data recording and transfer

Recording tree numbers and species	8/10 participants chose very suitable 2/10 quite suitable
Recording different plantation areas	8/10 participants chose very suitable 1/10 quite suitable 1/10 Neither suitable nor unsuitable
Ranking against other users for planting	7/10 participants chose not very suitable 2/10 quite suitable 1/10 don't know
Recording instances of disease	6/10 participants chose very suitable 4/10 quite suitable
Recording maintenance activity	6/10 participants chose quite suitable 4/10 neither suitable nor unsuitable 2/10 not very suitable
Maintenance reminders	6/10 participants chose very suitable 4/10 quite suitable
Twitter updates of trees planted	8/10 participants chose neither suitable nor unsuitable 2/10 quite suitable
Participant suggestions	No suggestions

All of the above uses of data were seen to be suitable for the app, except twitter updates and ranking against other users of the app for trees planted.

In section 8 participants were given a choice of 6 background/colour schemes. Overwhelmingly they chose the simpler single colour background as being the most suitable with 6 preferring the green background and 4 preferring the white with green accents. This was mainly due to the desire for a clean and simple interface with minimal distraction. Photographic backgrounds were deemed to cause difficulty with reading text.

The app will include a basic logo to create an identity and some symbols to improve the appearance of individual screens and buttons. Overall the consensus was that the app should be easy to sue and intuitive. Older respondents bemoaned the lack of instruction that comes with most modern technology, and that whilst something may seem intuitive to younger users, it is not always the case for older users. With this in mind it has been determined that there should be some kind of user guide included in the app.

3.2 User Stories

Whilst carrying out the surveys, participants were asked what they, as a user would require or expect from the app. The purpose of this was to get a greater understanding of importance of the functional requirements. There was also consideration of the needs of the developer in terms of maintenance of the app.

Table 8 User stories

As a user, I want to be able to involve my children in the running of the farm, they can find it boring helping out.	As a user I don't want to have to rely on civil servants to tell me how many trees I should plant and how to plant them.	As a user I want to be reminded in good time if there is something to be done so I can plan my time.	As a user I don't want to spend more time messing with the phone than actually getting things done
As a user I don't want to have to remember more usernames and passwords	As a user I don't want everyone knowing my business on the farm	As a user, I want to be able to look and see exactly when things like maintenance and planting were done and not have to go through loads of options	As a user I want to be able to change things later, like adding more trees or planting in stages
As a user I only have an old nokia phone, but I have a tablet. It would need to work on that too	As a gardening expert I would like people to have the confidence to try planting themselves. It is easy and good exercise.	As a garden centre owner I would welcome anything that will get people planting more	As a developer I want to receive feedback from users so any issues can be fixed and the app can be improved

These user stories are very useful in determining the required functional and non-functional requirements of the PlanTree app, as they reflect what users will expect and appreciate within it.

3.3 Platform

In building a mobile application there are several choices in terms of what platform to use. The first two that spring to mind due to their ubiquity are Apple's IOS and the Android operating system. The windows phone operating system can be considered as well as creating a cross-platform app that would work on multiple platforms. The benefits of each of these solutions must be explored, with the intention of getting the widest possible audience with the first release of the app.

- IOS:
  Apple's platform is sometimes mistakenly attributed to be the most widely used, but in fact is less widely used than the Android platform. This may be due to the brand's market position as a premium or luxury brand. It is true however that IOS has been a trendsetter in terms of innovative UI elements and mobile device technology. Apple devices do tend to more expensive than their rivals, with iPhone and iPad costing several hundred pounds outright. With this in mind it is conceivable that cheaper devices would be more favoured by those working in the outdoors due to the risk of damage. Apple's app store is very closely controlled and curated, with many very strict guidelines about app performance and content. Whilst it is not expected that this app will in any way be questionable in this

respect, such considerations could require a lot of management in the deployment stages of a project.

- Android:
  Android is by far and away the most widely used platform worldwide, with some 80% of mobile devices running it. Devices can be much more affordable than apple devices. Android is open source and gives easy visibility of the underlying operating system architecture to the developer, it is royalty free and easily accessible via Android Studio. The android studio IDE is based upon the proven platform of the Intellij IDE and as such should be robust and user friendly. The Google play store is not as restrictive as apple's app store, and whilst increasing accessibility can allow substandard apps to be present in the store, a clear review section allows users to make an informed decision when downloading apps. From the developer's point of view, the android studio IDE is a reliable and proven development environment and allows quick and easy testing on emulators or real devices, file structure is transparent and primarily using Java and XML it is an accessible and versatile platform.
- Windows Phone
  At first glance does not compete well with the Android and IOS platforms, with worldwide sales of devices in the single figures of overall mobile sales percentages, but as an operating system it has received much praise for its ease of use and clarity. It is also directly compatible with Microsoft office software, which would be very useful for the business minded user. There is also a reassuring consistency in its interfaces, unlike those of Android which can be overwhelming to inexperienced or older users.
- Cross platform/web/hybrid
  Web apps are becoming more widely used and offer some advantages worth considering when developing an application. The functional abilities of the browser in use by the device are exploited by the app in order to achieve its purpose, and with enhancements brought by HTML5, the abilities of web apps are approaching those of the traditional platform specific app. Due to the fact that they are programmed in HTML and CSS, web app development can be done with ease by those who have a working knowledge of web design. As this is common to browsers on all devices, the need to develop separately for different platforms is eliminated. Thus the time required for deployment on all platforms is much shorter. Hybrid elements allow certain functionalities of the device to be exploited, such as accelerometers and camera. For this project a web based approach may not be appropriate due to the fact that the device may be in use in areas where connectivity is poor and as such may limit the abilities of the app.

Having taken all the above platforms into account it was decided to proceed with development of an app on the Android platform. This offers an accessible development environment, best potential market exposure and good offline functionality. It is envisaged that future developments will include porting to Windows and IOS Operating Systems

3.4 Functional and non-functional requirements

Functional requirements:

1. **Users**: The questionnaire and user stories showed a reluctance from users to have to input or remember passwords. With this in mind it is envisaged that the initial implementation will not require the creation of any user accounts. Data will be stored locally on the device, and the nature of the work to be undertaken will not require any sensitive data to be stored. Further development may include an online backup facility to allow device changes. This would entail some kind of user identification such as via email address. An immediate alternative to this would be to allow the app and its associated data to be stored on an SD card by default
2. **Farm definition**: The users' farm will be defined by its postal address. This will allow the general area to be located by the app. The user will then be able to more specifically choose a location on the map to be the centre of the farm. This will be useful in areas such as the Republic of Ireland where there are no postcodes in use and the area defined by a postal address can be vague
3. **Plantation definition**: The user's plantations will be defined topographically either through use of the UI to 'draw' on a map or by the use of some variant of geolocation so a user may define an area by walking around it. Although one or the other will be implemented in the primary case, it is envisaged that both will be implemented in further development, in order that the user have a choice of methods.
4. **Tree capacity calculation**: The user will be able to use the app to calculate the maximum number of trees that a plantation can support, in order to prevent overcrowding and to make best use of the allocated land.
5. **Knowledge tools**: Advice on fundamentals of planting, spacing of trees, upkeep, tree varieties and species characteristics.
6. **Editing**: A sizeable plantation can take some time to plant out, so plantations must be editable once created, e.g. adding more trees, or indeed, harvesting or thinning a plantation.
7. **Reminders**: User should be alerted when maintenance tasks are due

Non-functional requirements

1. Error handling: If a user inputs invalid address info, this should be handed and produce an error message, not causing a system failure.
2. CPU usage: The app should not cause the mobile device to run slowly or freeze. Code should be efficient.
3. API: It is envisaged that the app will use the Google maps API for location services and mapping.
4. Security: User data should be stored locally and not be shared without the user's express permission.
5. Priority: The app must allow higher priority device functions to interrupt it, such as phone calls or alarms.
6. Device scaling: The app user interface must display appropriately on various sizes of device, with avoidance of fouling by keyboards and other system tools.
7. App scalability: The app should be capable of further development and use by large numbers of users.
8. Data integrity: The app should be able to save its data securely and correctly without corruption or loss.

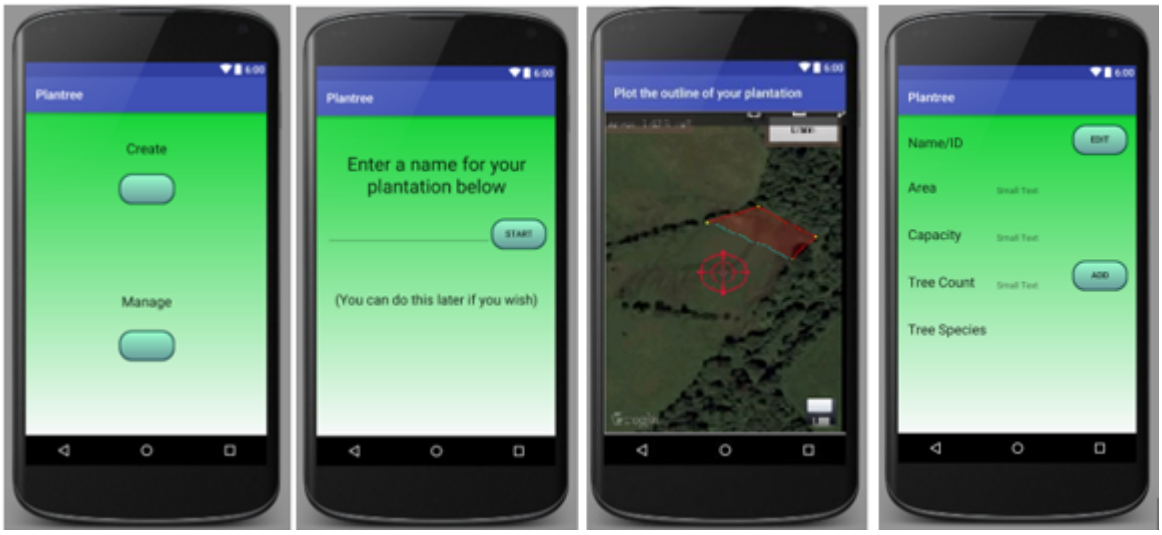
9. Offline and online functionality: There should be some level of offline functionality to allow users to review or amend certain data held within the device. This is due to the fact that users may be in a countryside location that does not have access to a network connection. This being the case, the app should not be totally limited to having a connection to carry out its functions. Also the app should not be tied to the immediate geographical area of reference to perform its functions. A user should be able to plan, view and amend details of their plantations wherever they are.

4. Design

4.1 Aesthetic

As established by the focus group questionnaire the aesthetic appearance of the app is to be as minimalist as possible with simple steps and clearly visible buttons. It is envisaged that the app may be used in the field so functions should be quickly available with relatively few steps between start up and beginning work. The app will have an appearance which will be evocative of nature, with green tones being used widely. The app will also be intuitive to android users with functions performed by menu bar items etc. The following are initial design studies to show what the anticipated finished interface may look like:

Fig 6: Appearance of app



From the focus group questionnaire it was established that a simple green and white colour scheme is the most desirable for the app so this will be retained. One development from the initial design was the use of sequential numbering on the home screen buttons as illustrated in figure 7. This is to give the user a greater sense of direction in the app and of the sequence they should follow. Allied to this is the user guide button which is given a high prominence in the design.

Fig 7: Final design for home screen

4.2 Interface

The user interface should be as intuitive as possible, and the user should be at any stage able to navigate back to the home screen. A user guide will be implemented which will provide clear instructions at every stage of the process of using the app. This will take the form of an overlay or popup and will avoid long sections of text where possible, descriptive icons will be used to demonstrate interface use. Whilst there is a school of thought that a good interface should not require a user guide, it is true that older users may struggle with user interface concepts and systems that younger generations have been acclimatised to from an early age. As it is envisaged that this app will be used by at least as many older users as younger users, accessibility and empathy for the user is paramount.

When designing and building the app care will be taken to adhere to best practice in terms of user interface design where appropriate. This can be defined by Mandel's 'Golden rules of Interface design'. Broadly these fall into three main categories:

1. Place the user in control
2. Reduce the user's memory load
3. Make the interface consistent

(Mandel, p5.3)

- Ways of placing the user in control include making the app easily navigable, allowing interruptions and corrections, displaying helpful messages and text to guide the user during operation.

  In the main, when the user is using the app it should be easy to understand. As many Android users will be used to the common behaviours of Android apps this is something that should be used. The back button should take them back to the previous screen or process without corrupting information, the app should interrupt to allow phone calls or messages to be delivered, the app should make use of things like action bars and toast displays.
- Reducing the user memory load will be achieved by reducing the amount of text the user should digest. Instructions will be graphical where possible, and the interface will be visually simple. The user will have options to undo and restart before committing to changes or

additions to the database. Wherever possible the user will be able to directly interact with the UI elements, an example being the ability to touch a plantation area on a map to add trees to it.

- Interface consistency will be achieved by creating page designs that fit together, and that are clearly designed to be part of the same app. Whilst it should be clear that is an Android app, it should not be mistaken for any other app for the same platform. Actions on one part of the app should be the same on others, e.g. pressing the help button or the back button and at no time should the user feel like they have made a mistake that they cannot easily fix.

At all stages the app should give clear instruction to the user if an incorrect input or action is made, this will be in the form of on-screen dialogues such as toasts. Through the employment of error handling the app will resist crashing and will display an adequately informative message as to why it cannot continue so the user can take remedial action, either by re-entering data or connecting to the internet for example. If the user decides to press the back button, this should not then take them to an activity where information could be accidentally re-entered to the database in error.
It was decided that the amount of individual buttons on the various screens of the activity should be kept to a minimum and that they should only be utilised for performing specific tasks such as drawing the plantation or searching for a farm address. Where possible the actual elements of the app should be interactive. This is demonstrated in the ability to mark locations on the map by touch, or viewing the map of a particular plantation by scrolling through a list of plantations and touching the one you wish to view. The maps used in the application are easily manipulated by hand and create a satisfying tactile experience for the user.

It was however necessary to create some custom buttons for the app. These are detailed below:

Table 9 icons

	The home button (for display in the action bar of each activities).
	The home screen task buttons, numbered to help the user understand the correct sequence of operations.
	The user guide button, this was not created but sourced from the web. (credit: Sebastian Rubio)

4.2.1 Activity layout

It is important that the app will perform well on whatever size of device it will be used on. This can be a challenging aspect of android development as layouts can appear sparse on larger devices. Several of the layouts use fragments to display information and receive user input, which allows scale considerations to be easily overcome. A map fragment for example can be specified to fill the entire screen and it will automatically scale to do so. Other considerations include device orientation changes, which must be handled in order to prevent loss of data (a rotation to landscape view will restart the activity) and possible fouling by pop-up keyboards which may cover important areas of the activity at inconvenient times. This must be anticipated and managed carefully through to the final design. The XML layout files for each activity allow a great deal of control by the developer over

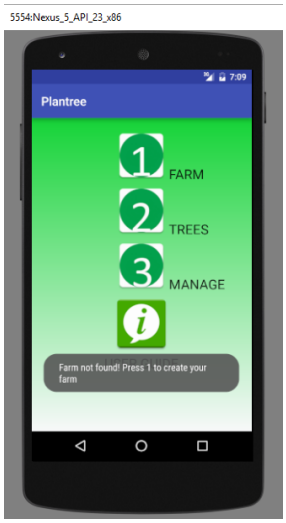
the appearance and functionality of the app, allowing the allocation of resources such as strings and images to user interface elements like buttons and text views. It is important that string resources be referenced rather than hard coded to the layout as this allows easy updating or language changes in future implementations.

4.2.2 Error handling

There are several places in the app where user input is required. This varies from choosing which activity to access, to inputting string information or creating polygons on a map. In the case of user miss-operation it is important that the app does not crash or freeze and that the user is alerted to what they have done incorrectly. The following screenshots show some of the places in the app where such error handling is to be used. This should contribute to a sense of robustness experienced by the user, and is in line with good practice in terms of user interface design.

The activity where the user creates a farm is especially important in this respect as the user is allowed to freely enter a postal address. Thankfully, the geocoder implemented in the design is relatively robust and will return a location if an attempt is made to enter a legible address, but if complete nonsense is input it will cause an exception that must be caught and handled. The use of try-catches around blocks of code is a very useful tool in this respect as it specifies the action that should be taken should the result of a call to a particular method fall outside of an expected range. Without this, the app would not 'know' what to do next and would hang or crash. The following emulator screenshots show the proposed responses to a few scenarios where user input falls outside of required parameters.

Fig 8.1: Error handling and displaying messages

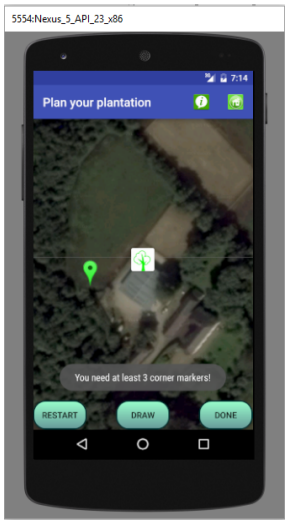


In this case the user has tried to create a plantation before defining their farm. The exception is caught and the user is displayed a message directing them where to go next.

Fig 8.2: Error handling and displaying messages

In this case the user has entered a nonsense address before pressing search. This is recognised and they are advised to try again. (This also works if no address is input before pressing search).

Fig 8.3: Error handling and displaying messages



In this case the user has tried to draw a plantation after only placing one corner marker. This is recognised and the user is advised to place at least 3 markers.

In some cases, the external resources required for the application to function such as the Google maps API may be unavailable, such as where internet connectivity is not available for example. This is another instance of where an exception must be handled. The following is a demonstration of a try-catch in action, where an action is tried and if the attempt throws an exception, this is caught and another action is taken instead, such as displaying a message to the user:

```
public void findFarm(View v) throws IOException{
    hideSoftKeyboard(v);
    try{
         /*main body of code for method*/
    }catch (IndexOutOfBoundsException e){
        e.printStackTrace();
        Toast.makeText(this, "Invalid address! Try again.",
Toast.LENGTH_LONG).show();
    }

        }
```

In this case the exception is caught and the user is advised that they have entered an invalid address.

Another method of error handling is to specify conditions that must be met before a method may operate. A simple example of how this is to be achieved is demonstrated in the draw plantation method from the create plantation class:

```
private void drawPlantation(){
    if(markers.size()<3){
        Toast.makeText(this,"You need at least 3 corner
markers!",Toast.LENGTH_SHORT ).show();
    }else { //MAIN BODY OF METHOD}
```

The condition of the main body of the method running is that the array of markers that the user places on the map has a size greater than three. Otherwise the method will only display a toast to the user. It is possible to create custom toasts with greater visibility and incorporating images etc, but this was not deemed to be a priority for this stage of development. The methods to be used for error handling will be decided on an ongoing basis dependent on the requirements of the code.

4.3 Application Architecture

4.3.1 Activity lifecycle

Although the main programming language used in Android development is Java, Android does not operate in the same manner as a standard java program. There is no 'main' method to speak of and work is divided between the activities of the app.

The activity is defined as a 'single focused thing that the user can do' by the Android developer documentation, and in the case of the Plantree app this is clearly demonstrable.

The first implementation and core functionality of the app will allow users to define a farm, define plantations within the farm, add trees to the plantations and view the plantation on a map along with its details. This was achieved through the creation of several activities, consisting of an XML layout file and a Java program file which would control the application logic for each activity. Java files that were not linked to layouts were also required to carry out database operations and other functions where the user input is not required. It is important that all elements are named appropriately according to their functions. As a result of this philosophy, Android development is very much object orientated. The following tables list the main file resources and the functions they carry out:

Table 10: Project files

Java Activity classes	Purpose
MainActivity	Starting point of the application
CreateFarm	User defines farm and saves to database
NamePlantation	User assigns name to plantation
CreatePlantation	User creates plantation and saves to database
DisplayPlantationsList	Displays a list of plantations created
ShowPlantationMap	Displays chosen plantation details
AddTrees	User adds tree numbers and varieties to plantations

Database operation classes	Function
DBHelper	Carries out all queries on database
FarmTableData	Holds info used to build farm table
PlantationTableData	Holds info used to create plantation table
SpeciesTableData	Holds all info used to create and populate species table
PlantedTableData	Holds all info used to create planted table

Misc classes	Function
PlantationListDataAdapter	Listview adapter class forDisplayPlantaionsList
DataProvider	Provides data objects for the ListView adapter

XML layout files	Function
activity_main	Used to display MainActivity
activity_create_farm	Used to display CreateFarm
activity_name_plantation	Used to display NamePlantation
activity_create_plantation	Used to Display CreatePlantation
activity_list_plantations	ListView to hold all plantations
activity_display_planation_info	Layout to hold ListView
display_plantation_row	Custom layout for rows in ListView
activity_show_plantation_map	Used to display plantation details
activity_variety	Used to display AddTrees
Popups	Used to display user guide info

Additional to these there are many more files dealing with the underlying operations of the app, and many resource files that hold data relating to strings and image resources for instance, but these are far too numerous to list.

Each activity has a lifecycle with various methods carrying out work to create and destroy it as required. In general the developer overrides these methods to achieve desired results as well as creating bespoke methods.

Arguably the most important method in any activity is the onCreate method. This is responsible for the instantiation of everything in the activity, preparing the activity to be functional when it comes to the fore of the app and is visible to the user. This is where the layout for the activity is identified and assigned as well as any other resources needed, such as Google maps and UI elements such as buttons etc. The onCreate will only be called once during the lifetime of an activity.

When the user navigates away from the activity by choice or by design of the app, the onStop method will be called to allow its processes to cease but be available for use again if required. It goes to the bottom of a 'stack' of activities. If onDestroy is ever called the activity will no longer be available and must be rebuilt by another call to onCreate. Each activity which the user interacts must be declared in a file called the android manifest.

The developer can make use of the logcat output of the Android Studio IDE to keep track of what methods are running. A tag is created in each activity which will identify it and allow filtering of messages, and a string message will produce an output relevant to the method the developer is tracking. The following is a log message statement from the onCreate method of the CreatePlantation class:

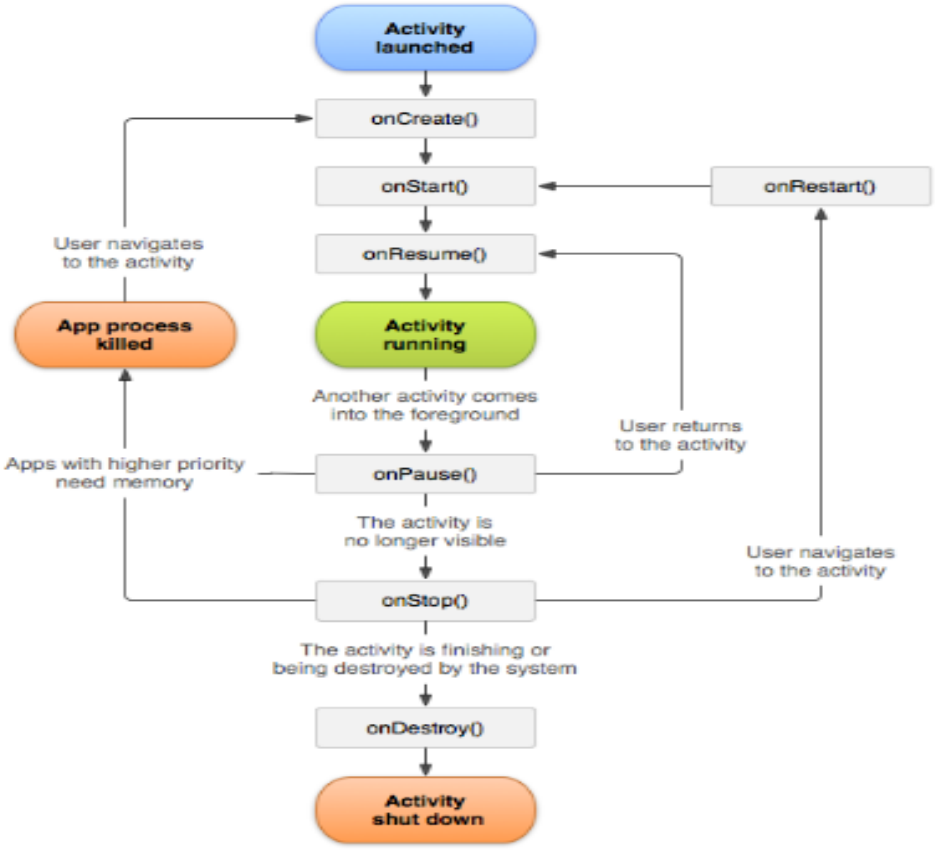
```
Log.i(TAG, "map initialised");
```

The output is as follows:

```
08-30 07:42:54.820 10304-10304/com.plantree.edward.plantree I/CreatePlantation: map initialized
```

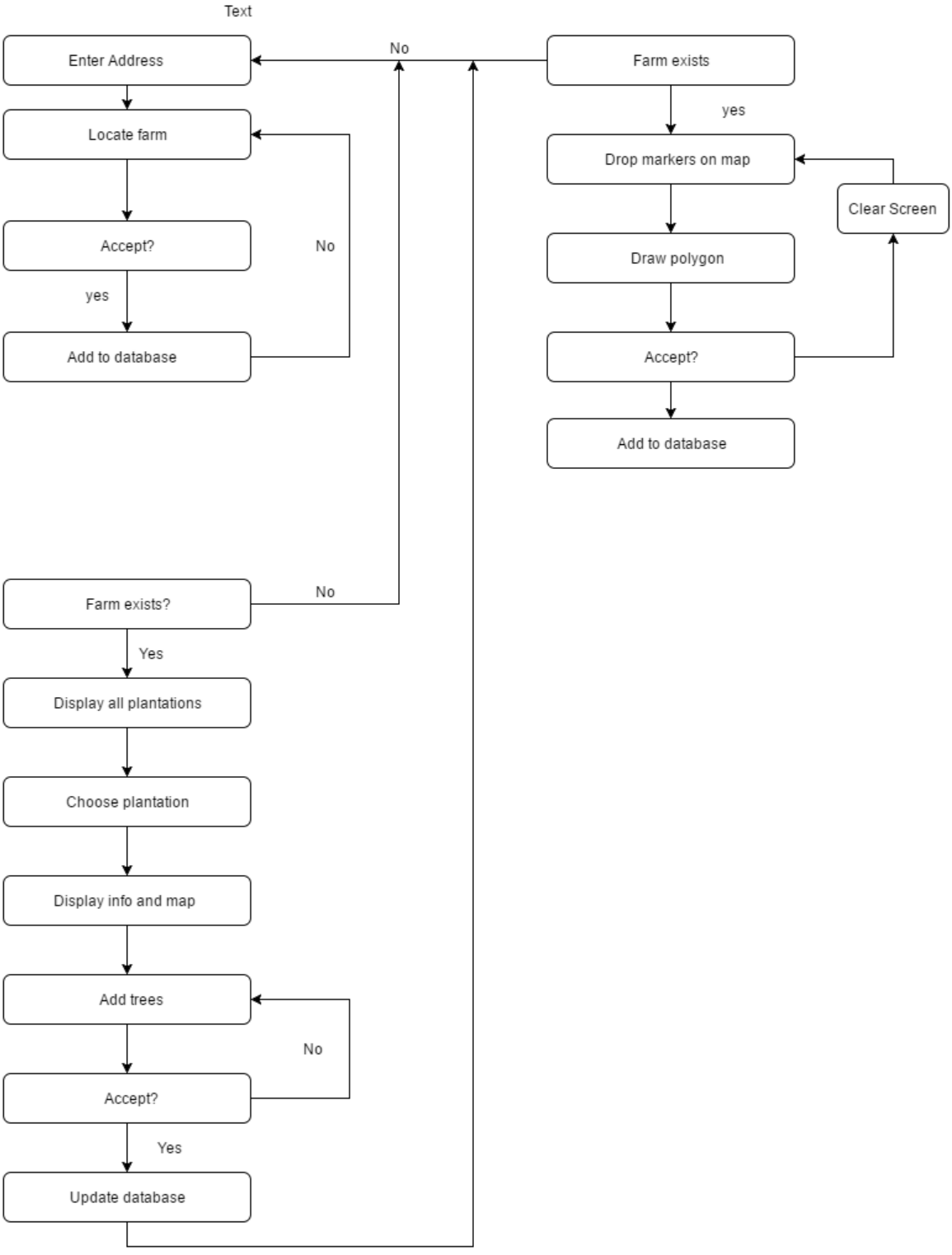
The logcat output is a very useful tool for debugging. Most errors in code can be identified easily by analysing its output, and variable values can be passed to it for analysis without effecting the outcome of the program logic. Detailed examples of how debugging was carried out will be provided in further chapters of this report.

Fig 9: Lifecycle of an activity



The following is a map of how the user will progress through the various activities of the app.

Fig 10: Application map



Moving between activities is, in general, done with the use of intents. An intent can be set to operate on the click of a button or UI element, or the completion of a specific task. They can also be used to pass data between activities for further use. Intents can be useful in directing where the user goes in the app, for instance it may not be desirable to go back to the most recently accessed activity in the case that that activity formed part of a transactional function in conjunction with other activities. An example of this would be returning to the CreatePlantation activity without first visiting the NamePlantation activity. Intents will be used to marshal this movement. As with other elements of the app, intents must be named appropriately so as to be recognisable.

4.4 Database Design

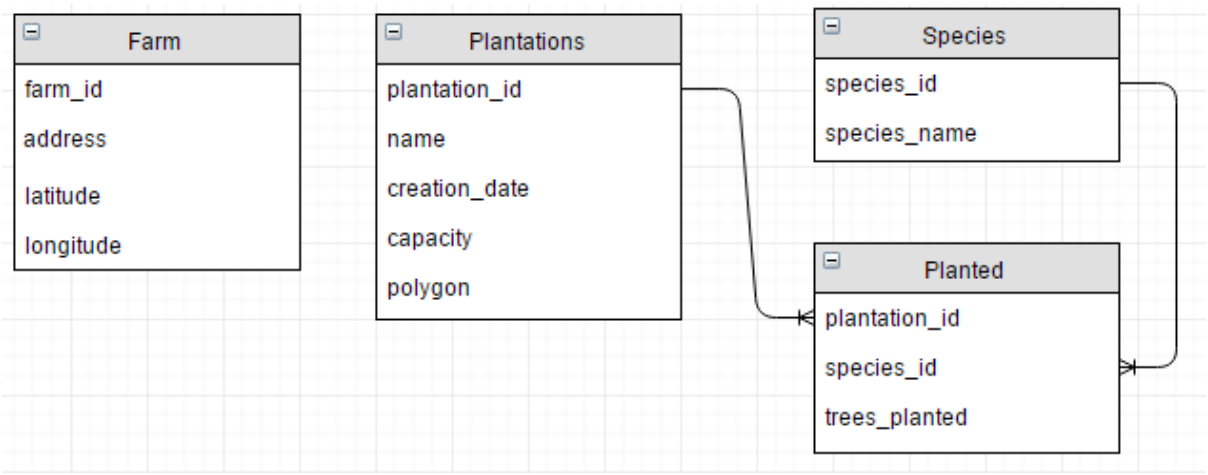
The database used to store the app data will consist of four tables.

1. The farm table:
   This will be used to store the details relating to the user's farm. The user will input their postal address and will define the location coordinates of the centre of the farm. At this stage there will only be one farm per user. The location coordinates will be used by other activities as a way of centring the map on the general area of the farm.

2. The plantations table:
   This will hold the data relating to the individual woodland plantations that the user creates. The attributes will be the id of each plantation, the name of each plantation, the date created, the capacity, and the polygon details (which will be encoded as a string value).

3. The species table:
   This will hold data relating to the varieties of trees that can be planted. The attributes for each variety will initially be the id and the name, although there will likely be future development in this area. These values will be prepopulated.
4. The planted/stock table;
   This will hold data relating to the types and numbers of trees planted on each plantation. It will reference the id of tuples / rows in the plantations table and the species table. This will function to break down the many to many relationship that might otherwise exist between the plantation table and the species table.
   The table structure is illustrated below. It is expected that this table will have a composite primary key in order to enforce a relationship between the plantations table and the species table

Fig 11: Plantree database ER diagram



Note: Due to the fact that there is at this stage only one farm per user, it is not necessary to have a relationship between the farm table and any others, as this table will only contain one row.

4.5 Design Summary

In conclusion the aim is to develop the core functionality of the Plantree app to be as robust and user friendly as possible. There should be no possibility of the user making errors that would cause the app to fail and the user interface should be clear and intuitive to follow. The design of the app will be object orientated and as interactive as possible in terms of the user interface elements, with clickable items such as list views, maps, and polygons. The user should at all times be aware of their location within the app and navigation back to the home screen should be easily achieved with a maximum of one touch from any activity. The user will be provided with feedback throughout their progress through the app and advised appropriately and clearly of any errors and progress.

5. Implementation

This chapter will deal specifically with the process of implementing the designed features and interface of the app. There will be a short discussion of the IDE used to build the app, best practice in terms of coding standards, and there will also be a description of the method of version control as previously discussed.

The specific methods used to achieve the desired functionality of the app will be discussed.

5.1 Coding standards

When developing an app or program of any type it is important that any code that is written should be easily understood by anyone else who reads it, this is an especially important aspect of agile methodology when collaboration on projects necessitates sharing of work. Other considerations include:

- Maintenance – 80% of the lifetime cost of software is maintenance, if code is easily understood and correctly structured this will reduce long term costs. (Hommel et al p.1)

- Troubleshooting and debugging pre-release will be much easier if code is easy to read and logically laid out.

Some of the standards to be adhered to are as follows:

- Naming classes, methods, variables etc.: – These should be named so as to be recognisable by their function and following case convention, i.e. the use of capitalisation, underscores, camel case. For example a class name such as CreatePlantation will be recognisable by the capitalisation of the first letter, whereas a method such as drawPlantation will be recognisable as a method by the first letter being lower case. Both the examples used give strong hints as to their function with their names.
- Commenting: Comments should be included where necessary to explain the purpose of blocks of code or the role of a class or method. Comment tags such as //TODO are useful in marking areas where further work is required.
- Indentation: Code within methods or subclasses should be indented to allow the reader to more easily identify the beginning and end of these sections. Indentations should be uniform where possible.
- Layout of classes: Layouts should follow the same pattern throughout all the classes, for example, grouping of imports, variable types etc. This contributes to defining a style for the developer.

5.2 Platform IDE

Android development primarily uses the Android Studio IDE. This is based on the IntelliJ IDE and is a versatile and user friendly system

There has been a thorough investigation of the Android environment, with research and practical testing on the various elements of an application running on it. A full understanding of the system was required before a project of this scope could be carried out. Small practice apps were built to investigate and test the various functions of Android, such as:

- Activity lifecycles and their interaction with layouts
- The multitude of user interface elements available
- The use of intents to activate other activities and pass data between activities,
- The use of API's
- The use of services to perform functions without user interfaces
- Programmatic manipulation of UI elements
- Android libraries
- Data storage and integrity

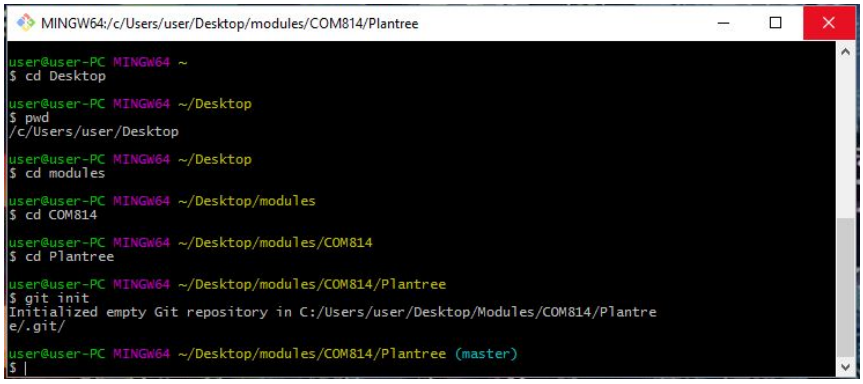
The Android studio IDE allows full visibility of the structure of the application in order that the novice developer can get a good understanding of the relationships between the various files held within. External files such as images and text can be easily imported for use in the project and many plugins are available also. Layouts can be examined in the layout design editor which simulates how they might appear on a device for quick and easy alterations, rather than having to wait for an emulator to run them. Debugging is easy due to the device monitor which displays real time log messages from the device or emulator that is connected for testing, and the user can set filters and create log messages to display as required. It is also allows version control software integration which will be discussed in the next section.

5.3 Version Control

As previously discussed, one of the most important areas of the project is the mitigation of risk in terms of data loss and corruption. It is necessary to employ some method of version control, to allow both rollbacks to earlier versions should the need arise and also for the retrieval of project data in the case of machine issues or loss. A few different systems were investigated and it was decided that GIT should be used as the main means of version control. This was due to the reputation of the product and its ease of use Git uses a bash script command line to target files to track. In this case I set the android project folder as the master. Any changes to this would be detected by GIT and each update would be a version that can be accessed at any time in the future if required. These versions are stored locally on the machine in use. In a single developer project this is mainly used as a risk mitigation strategy, but it also allows analysis of the project by the scrum leader, and in a multiple developer project it would allow the tea members to collaborate more effectively as versions can be cloned to allow division of labour, and as a collaborative tool it is very useful.

The first step in setting up version control is to identify the master repository. In this case it was the Plantree project file within the COM814 project file. This was achieved by a series of BASH script commands as illustrated in figure 8. The correct folder is located within the file hierarchy of the machine and the command '$ git init' is called in that location. The file is them marked as the master and any changes to it are tracked. Any time an update is made by creating a 'commit' a log is kept of what has changed, along with a message input by the developer to aid identification of versions.

Fig 12: Initialising GIT



In order that the project content is safeguarded from loss in the event that the computer hard drive fails or the machine is destroyed or stolen, it is necessary to setup a remote repository to push the content to. The Ulster University's GIT Lab server was utilised for this. This provides a secure and private repository. Once the account is setup and linked the content is pushed to it by using the command "$ git push .".

Android studio will also recognise when Git has been assigned to the project, and will automatically update the staging area with any newly created files in the project if the developer so wishes. In this case however, it was decided to retain control over what was added to Git based upon achieving the requirements of the app design. This would prevent the creation of versions with non-functional, incorrect or superfluous code that would need to be corrected should a revert be required.

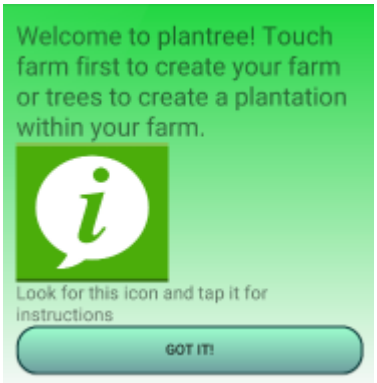
5.4 User Guide

It was decided that the user guide should take the form of pop-ups that would activate when the user presses the info button on the action bar of each activity of the application.

The first pop-up shows the user what step to take first and how to find further help on each screen.

The user must dismiss the popups by pressing the button marked 'GOT IT!'. This provides an un-intrusive yet comprehensive user guide throughout the app.

Fig13: Example of user guide popup.



Each pop-up consists of an XML layout file and is activated by the use of a LayoutInflater object in the onOptionItemSelected method of each Java class controlling the activities of the app. The first pop-up is activated by a prominently placed button on the main screen, this gives an overview of how to proceed and where to find help in subsequent sections of the app. In all other screens the help button is placed on the action bar at the top of the activity. There is also a home button located here which will take the user directly to the main menu from anywhere in the app. The action taken by pressing an action bar button are decided via a switch statement in the onOptionItemSelected method.

The strings for the instruction text are kept in a separate resource file from other strings to allow easy updating in the event of future updates and development of the app.

Fig 14: Example of onOptionsItemSelected method (onClick methods removed)

```
@Override
public boolean onOptionsItemSelected (MenuItem item){
    switch (item.getItemId()){
        case R.id.create_plantation_help:
            //code to run when help button is pressed

          return true;

        case R.id.butt_home:
            //code to run when home button is pressed

          return true;
    }
    return super.onOptionsItemSelected(item);
}
```

5.5 Home button

In consulting the focus group, it was deemed important that the user can at any stage return to the home screen of the app. To this end a home button was included on the action bar of each activity alongside the user guide button. This functionality was also accessed via the switch statement on the onOptionsItemSelected method. The action bar is used throughout the app as a tool for user orientation. Every page includes the home and user guide buttons and some pages include a text title which will give the user hints as to the purpose of that particular page.

5.6 Google Maps API

The App will need to make use of mapping for most of its functionality. Rather than creating this functionality from scratch it is expedient to make use of an Application Plug-In (API). This will allow

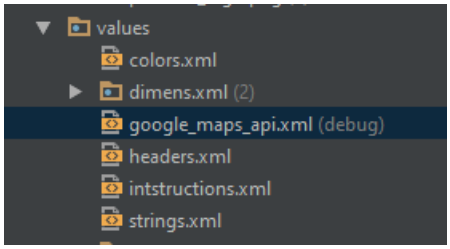
the app to make use of pre-existing solutions and dramatically cut the development lifecycle time. The Google maps API offers the best solution for this for Android devices and offers many useful features in conjunction with mapping. In terms of the definition of the farm itself, this can be done using a geocoder, which converts a postal address into a location object that can be displayed on a map.

For defining a plantation within the farm, Google maps support markers and polygons. Once a polygon is created, its area can be calculated programmatically to allow calculation of the number of trees that a particular plantation can support.

The API also supports geolocation which may be used in further developments of the app.

Using this API with Android is relatively straightforward, as template layouts are included to hold map fragments and where to insert the API key generated by the google API dashboard is clearly signposted. Whilst Google do have a pricing scale for use of the API, it is unlikely that this implementation will see enough use to trigger the lowest boundary, which is 25000 user hits per day. As the Google maps interface is well established and used by many existing apps, most users should be familiar with it and the touch ability it includes, such as zooming in and moving around the map. It is seen as the benchmark for mapping applications and as many additional functionalities that may be utilized.

Fig 15: Pre-existing XML file within the project for the Google Maps API information



5.7 Creating the Farm

The first task the user must perform is to define the farm. The purpose of this is two-fold. Primarily this is to assist the user in navigating to the correct area when later defining plantations with the farm. The farm will be defined as a single location in the centre of the farm. To achieve this the Geocoder library was used. This is part of the Google maps API and facilitates the conversion of a postal address into a location on a map. The address entered by the user is then stored in the database as the address of the farm.

Once the location of the postal address is found the map will centre over that area. The postal address of a farm may often be at the end of a lane so the user can manipulate the map, and by holding a finger on a central location, they can define the central point of the farm. This was achieved by implementing the onMapLongClickListener interface of the Google maps API. When a long click is detected by the listener, a marker is placed at that position on the map.

A custom marker was created for this purpose and displays the app logo at the chosen location. This location will be reused is subsequent map activities to centre the map on the farm. This method is sympathetic to user error as it allows the user to redefine the centre point by pressing on the map again. The location is only saved to the database when the user presses the button marked ‘done’. In this way the whole process can be viewed as a single transaction and is less likely to lead to database corruption than if farm details were entered in stages. As always the user is advised by a toast message that the operation has been completed successfully and that the farm has been defined.

Fig 16: onMapLongCLick method from the CreateFarm class.

```
@Override
public void onMapLongClick(LatLng latLng) {
    if (farmMarker!=null) {
        farmMarker.remove();
    }
    farmMarker = farmMap.addMarker(new MarkerOptions()
            .position(latLng).title("Farm yard")
            .icon(BitmapDescriptorFactory.fromResource(R.mipmap.plantree_logo)));
    farmYard = latLng;
    Toast.makeText(this,"Farm yard defined",Toast.LENGTH_LONG ).show();
```

A further benefit to this approach was that in the Republic of Ireland postcodes are not used, and addresses can be quite vague. The input address can centre the map a few kilometres away from the actual farm and using this interface the user can more accurately specify the farm location. This functionality was tested on ROI addresses and performed well.

The second reason to define the farm is for use in future development of the application, for example, if building a website to compliment the app, also for future data mining opportunities to gather information based on trees planted in certain areas.

One important function of the CreateFarm class is to store the details of the farm of the farm (its postal address, and its coordinates in the SQLite database. This information is then easily accessible by other activities in the app. The process of inserting data to the database will be described in detail in subsequent chapters. This transaction is one of the final actions of the CreateFarm class before its activity ceases, the user having ample time to double check and refactor their farm defining info before accepting it. This means that there is less chance of erroneous data being stored. Furthermore, the geocoder functionality in the findFarm method is surrounded by a try – catch, in order that the application will not crash when the user enters text that does not create an address. In this case the user is prompted to try again.

In the event that the address is not recognised the user can still manipulate the map manually to find their farm using swipe gestures on the map.

5.8 Creating, Measuring, and Storing Plantations

The creation of the plantations is done within the CreatePlantation class. The first implementation was to allow the user to manually define plantations by placing markers on the map to define the corners. The app should then join these points up, creating a polygon and calculate its area. This was achieved by implementing the following:

- OnMapLongClickListener – When the user touches a point on the screen this is recognised as a long click. The app then positions a marker at that point and also stores the marker in an array list of markers. When the user has finished placing markers they press a button which uses the markers to create a polygon drawn on the map. The benefit of using an array list for this is that it is mutable in length so as to not restrict the user to a maximum number of markers. In this way, a plantation can be virtually any shape.
- DrawPlantation – This method is called once the user has placed the markers to define the corners of the plantation. The minimum amount of markers that can be placed is three, to form a triangle. This is enforced by an if / else statement which will evaluate whether this is the case before allowing the method to run. If the user places less than three markers and attempts to draw the plantation, a toast is displayed to advise the minimum amount of markers.

This method passes the list of markers to a PolygonOptions object which will then draw the plantation on the map. A Boolean variable called plantationDefined is then assigned the value 'true', This will allow the user to progress to the next activity of the app as it is required in order for the plantation to be saved to the database.

Fig 17: Method to draw plantation

```
private void drawPlantation(){
    if(markers.size()<3){
        Toast.makeText(this,"You need at least 3 corner markers!",Toast.LENGTH_SHORT ).show();
    }else {
        PolygonOptions polygonOptions = new PolygonOptions()
                .fillColor(0x330000FF)
                .strokeWidth(3)
                .strokeColor(Color.MAGENTA);

        for (int i = 0; i < markers.size(); i++) {
            polygonOptions.add(markers.get(i).getPosition());
        }
        plantation = plantationMap.addPolygon(polygonOptions);
        plantationArea = SphericalUtil.computeArea(markerLatLngs);
        Log.i(TAG, "Plantation area is " + Double.toString(plantationArea));
        Toast.makeText(this, "Plantation area is " + Double.toString(plantationArea), Toast.LENGTH_LONG).show();
        plantationDefined=true;
        dateCreated = getDate();
        capacity = Double.toString(plantationArea*0.5);
        Log.i(TAG, "plantationDefined = "+ plantationDefined);
```

This method utilises two array lists of information to create and measure the plantation. One is called markers and stores marker objects, these are then used to draw the plantation by the polygon options object. The second array list is created at the same time and holds LatLng objects relating to the positions of the markers in the first array. These are used to compute the rea of the plantation

5.8.1 Polygon Area

The plantation area is found with the use of the SphericalUtil class of the android-maps-utils library. This is a pre-existing means of determining the areas of regular and irregular polygon objects. Once the area is found, it is then used to calculate the maximum number of trees a plantation can support. The basis for this calculation is that in general, trees should be planted at least two square metres apart, and as a result the calculation is a trivial matter of dividing the area of the proposed plantation by two.

SphericalUtil requires a list of LatLng objects in order to assess the area of a polygon. This necessitated the creation of an array list of LatLngs corresponding to the locations of the markers defining the plantations. This was achieved by extracting the latitude and longitude information from each marker as it was created and creating a LatLng object for each. This information was then stored in the array list called markerLatLngs.

During development an issue was encountered where if the plantation was restarted the new area was a combination of the first attempt and the second. This was due to the fact that the clearScreen() method had not been updated to clear the array list of marker locations as well as the array list of markers. This caused the area calculation to be dealing with a much larger array size. This was remedied by updating the method to clear both array lists on restart of the plantation.

Fig 18: Clearing the plantation

```
361         //erase contents of marker arraylist
362         markers.clear();
363         //erase contents of marker locatons arraylist
364         markerLatLngs.clear();
```

5.8.2 Encoding and decoding polygons and viewing the plantations

Once the plantation polygon has been created and accepted by the user, it must then be stored for future reference. Due to the fact that SQLite is to be used to store the app data (this is discussed in detail in subsequent chapters), this would prove to be one of the more challenging aspects of the implementation. SQLIte stores all info in the form of text files, as result, all data must be converted to strings or basic datatypes such as int in order that it may be inserted to the database. Objects such as polygons are quite complex, and could only be stored by extracting the latitude and longitude data from the points that make up the polygon for conversion to strings. When this information is subsequently extracted from the database (i.e. if the polygon is to be displayed again) the same process must be done in reverse, and the polygon would be redrawn from the string information. Managing varying amounts of points for different polygons would make this quite a laborious process where it not for the existence of the PolyUtil class in the android-maps-utils library. When this class is implemented it can decode a polygon or poly line object of any size or complexity into a simple string object which can then be inserted to an SQLite database. This function is carried out by the encodePlantation method within the CreatePlantation class. The previously created array list of LatLng objects that correspond to the marker positions placed by the user when defining the plantation is passed as a parameter to the encode method of the PolyUtil class of the aforementioned class of the android-maps-utils library.

This can also be reversed so a polygon can be stored, retrieved and re-displayed with ease. This functionality is utilised when the user reaches the display plantation info activity of the app. This takes the form of a list view showing the details of each of the plantations. When the user taps any of the items in the list a map fragment is displayed showing the position of the plantation. It is envisaged that due to the regular periodic updating of Google's map data the actual trees will be visible within the plantation by the time they achieve a moderate level of growth.

The call to the DecodePlantation and drawViewablePlantation methods was placed within the onMapReady rather than the onCreate method in order to avoid null pointer errors being generated by trying to place a polygon on the map before it was ready. A lower zoom level was set for this map to allow for a wider view of the farm area.

5.8.3 Displaying the plantations in a list view

In order for the user to be able to easily get an overview of the plantations on the farm it was necessary to incorporate a list view activity. The initial intention had been to use a placeholder activity to display this information, but this was abandoned as it would have restricted the user to a certain number of plantations with their farm and would have required more programming to implement. The creation of the list view was, in itself quite challenging. This required the creation of a data adapter class in order to insert the required information into each item in the list view and a data provider class to provide string objects for the data adapter. When each plantation was added to the list view it was necessary to create an array list to keep track of the polygon details of each of them so that information could be passed to the view plantation activity for decoding and display. This demonstrates the effective use of parallelism in arrays.

In the final implementation it was decided to minimise the amount of data displayed by the list view in order that when viewed on smaller devices the list does not appear overcrowded. Only the

plantation ID and the name is displayed ant the other details are passed to the next activity for display alongside a map of the plantation as illustrated in figure 12

Fig 19: Listing plantations

5.8.4 Passing plantation data to another activity

The main functionality of the list view activity descried above is to display the plantations and pass their data to a map activity when clicked.

When passing more than one data item to another activity it is necessary to use a bundle to do so. Figure 12 demonstrates the intent and bundle used to transfer data from the list view activity that retrieves the data from the database and the show plantation map activity that will display the full details. Each item in the bundle is given a key (in capitals) in order that it can be identified by the receiving activity and a value. In this case all the values are retrieved from the local array lists holding the data from each plantation, these are created at the same time as the list view adapter is populating the list view as otherwise it would not be possible to identify the data required.

Fig 20: An intent with a bundle of data

```
Intent pass_to_map = new Intent(getApplicationContext(), ShowPlantationMap.class);

//pass_to_map.putExtra("POLYGON",polygons.get(position));
//startActivity(pass_to_map);
Bundle extras = new Bundle();
extras.putInt("EXTRA_ID", identities.get(position));
extras.putString("EXTRA_NAME", names.get(position));
extras.putString("EXTRA_POLYGON",polygons.get(position));
extras.putString("EXTRA_DATE", dates.get(position));
extras.putInt("EXTRA_CAPACITY",capacities.get(position));


pass_to_map.putExtras(extras);
startActivity(pass_to_map);
```

The intent object specifies the class that the bundle will be sent to, in this case, ShowPlantationMap.class. When the bundle is received in the target class, (in the onCreate method) the tags specified in the sending intent are searched for as illustrated in figure 14:

Fig 21: Receiving an Intent and Bundle

```
Bundle recievedBundle = this.getIntent().getExtras();

if(recievedBundle!=null){
    id = recievedBundle.getInt("EXTRA_ID");
    name = recievedBundle.getString("EXTRA_NAME");
    plantationPolygon = recievedBundle.getString("EXTRA_POLYGON");
    dateCreated = " "+recievedBundle.getString("EXTRA_DATE");
    capacity = recievedBundle.getInt("EXTRA_CAPACITY");
```

Once the values are received they can be used by the map activity to display the plantation details. In the case of the plantationPolygon string, this is decoded using PolyUtil.decode into a list of LatLang objects. This is then translated to an array list to be used as an argument with a PolygonOptions object in order to redraw the selected plantation.

When this polygon is created it is also assigned as clickable. When the user taps on the plantation the final activity is triggered.

5.9 Adding trees to plantation

The function of final activity in the process of creating a plantation is to add trees to the previously selected plantation. Two variables are passed from the map activity using the intent-bundle method as described above. These are the plantation id and the plantation capacity. The AddTrees activity class consists of a spinner which holds the available options for tree species, and an input text which allows the user to select a value for the amount of trees of a particular species planted. There is error handling which prevents the user entering more than the capacity of the plantation, in the form of a trivial evaluation of the input value against the capacity and zero. If the number of trees is invalid, no action is taken, other than displaying a toast to the user.

Fig 22: Evaluating number of trees planted

```
if((treesAdded>0) && (treesAdded<= plantationCapacity)) {

    updatedCapacity= Integer.toString(plantationCapacity-treesAdded);
    newPtDb.putStockInfo(newPtDb,plantationId,speciesId,numTrees);
    newPtDb.updatePlantationCapacity(newPtDb,updatedCapacity,plantationIdInt);
    //add a line of code to reduce the capacity figure in the plantation by the amount of trees planted
    Intent go_home = new Intent(getApplicationContext(),MainActivity.class );
    startActivity(go_home);
}else{
   Toast.makeText(getApplicationContext(),"Invalid amount of trees! Enter a number from 1 to "
           + plantationCapacity,Toast.LENGTH_LONG).show();
}
```

The method will then create a new capacity value by subtracting the number of trees added from the initial capacity of the plantation. In this way the user cannot then overplant the area. The AddTrees class then uses an object of the DbHelper class to insert data to the planted table and update the capacity column of the row in the plantation table relating to the chosen plantation. Database actions will be discussed in the following section.

5.10 Database Implementation

For the initial phase of development it was decided that any data relating to the Plantree app should be stored locally to the device using the SQLite database that Android employs. This is a secure and straightforward way of persisting data as only the device upon which the app is loaded can access this data. It was necessary to create a Java class to carry out all the database operations in order that these be organised effectively. The class was named DbHelper and it extends the SQLiteOpenHelper class which has all the required functionality incorporated. Information relating to the individual tables such as table names and column identifiers or any pre-populated data was stored in separate files for easy updating and reviewing, these have easily identifiable names such as FarmTableInfo.java and PlantationTableInfo.java. When these are imported into the DbHelper class it can access all their values for use in creating the database.

The DbHelper class has a series of methods which use a combination of SQL and Java commands to carry out queries upon the SQLIte database on the device. Figure 23 illustrates a static string used to define the farm table columns and constraints.

Fig 23: DbHelper create farm query

```
public static String CREATE_FARM_QUERY = "CREATE TABLE " + FarmTableInfo.FARM_TABLE_NAME
        + "(" + FarmTableInfo.FARM_COL_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
            + FarmTableInfo.FARM_COL_ADDRESS + " TEXT NOT NULL, "
            + FarmTableInfo.FARM_COL_LATTITUDE + " TEXT NOT NULL, "
            + FarmTableInfo.FARM_COL_LONGITUDE + " TEXT NOT NULL);";
```

The onCreate method of the DbHelper then passes this string as an argument to the db.execSQL method of the superclass for execution.

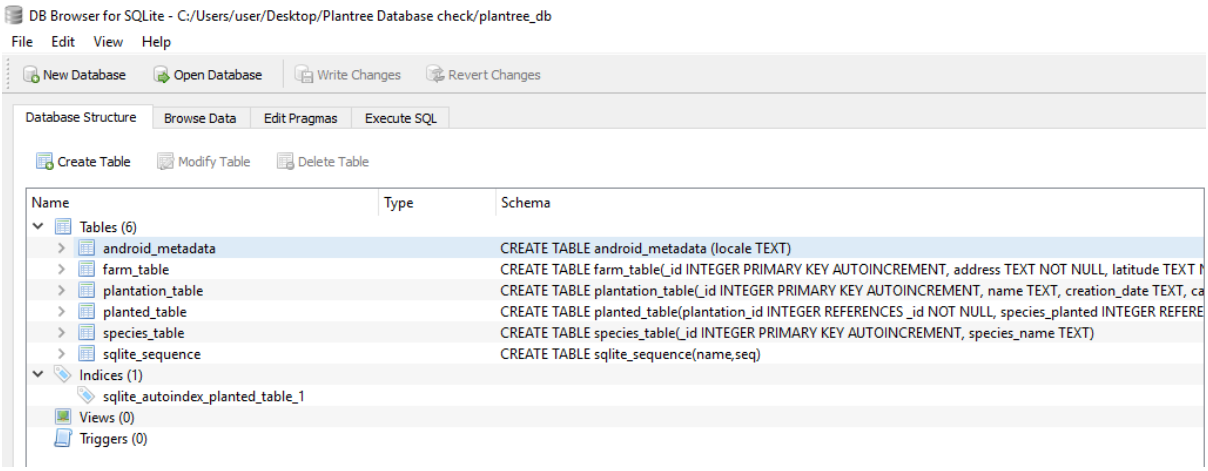
Fig 24: DbHelper onCreate using queries

```
db.execSQL(CREATE_FARM_QUERY);
Log.i(TAG, "farm table created");
```

In practice this process was quite difficult due to the mixture of SQL type syntax and java syntax, it was necessary to ensure that spacing was correct and that all required delimiters were present. SQL exception feedback is relatively vague in comparison to that generated when using java so knowledge of correct syntax is very important.

The database is created the first time an object of the class DbHelper is instantiated in the app, In this case it is in the onCreate method of the main activity, which is the first activity the user sees when opening the app. The Dbhelper will only create the tables if they do not already exist on the device, in this way the stored data is not overwritten each time the app starts up. For security reasons, it is inherently difficult to view database files on an android device, so an emulator must be created in android studio to run the app and extract the database tables for inspection.

Fig 25: Database tables viewed in DB Browser upon creation



Using the DB browser program it is possible to check if the methods of the DB helper class are operating correctly in terms of adding and updating the data in the tables and maintaining the relational constraints required.

5.10.1 Relational integrity in SQLite

As its name suggests SQLite is not a full implementation of SQL and as such some of the functionality that exists in normal SQL queries is not supported. The version currently used by Android, foreign keys are not supported so the DbHelper must manage all transactions to ensure database integrity.

It is possible to create relational links between tables by using the 'REFERENCES' argument in SQL queries to create tables. In the Plantree database the planted/stock table exists to prevent the occurrence of a 'many to many' relationship existing between the plantation table and the species table. It has one column which references the plantation id from the plantations table, and another which references the species id from the species. When the stock table is being created, a composite primary key is declared which is essentially a combination of these two fields, and enforces the relational integrity between these tables.

Fig 26: Creating table with composite primary key

```
public static String CREATE_STOCK_QUERY = "CREATE TABLE " + StockTableInfo.STOCK_TABLE_NAME
        + "(" + StockTableInfo.STOCK_COL_PLANTATIONID + " INTEGER REFERENCES "
              + PlantationTableInfo.PLANTATION_COL_ID + " NOT NULL, "
             + StockTableInfo.STOCK_COL_SPECIES + " INTEGER REFERENCES "
              + SpeciesTableInfo.SPECIES_COL_ID + " NOT NULL, "
             +StockTableInfo.STOCK_TREES_PLANTED + " INTEGER, "
             + " PRIMARY KEY ( " + StockTableInfo.STOCK_COL_PLANTATIONID + ", "
              + StockTableInfo.STOCK_COL_SPECIES + "));";
```

At this stage in development it was not deemed necessary to create any more relationships between tables, but a relationship between the farm table and the plantation table would be necessary should a web based implementation of the app be developed, as there would be more than one user and more than one farm to consider.

5.9.2 Inserting data and updating tables

The SQLiteOpenHelper class that the DbHelper class extends has pre-existing methods for the insertion and updating of tables. These methods take the table names, column names, rows and values to be added/inserted as arguments. An example of a call to one of these is shown in figure 27.

Fig 27: example of an update method

```
public void updatePlantationCapacity (DbHelper dbHelper, String updatedCapacity, int planationRow){
    SQLiteDatabase SQ = dbHelper.getWritableDatabase();
    ContentValues capacityCv = new ContentValues();
    capacityCv.put(PlantationTableInfo.PLANTATION_COL_CAPACITY,updatedCapacity);

    SQ.update(PlantationTableInfo.PLANTATION_TABLE_NAME,capacityCv, PlantationTableInfo.PLANTATION_COL_ID + " = " +planationRow,null)
}
```

The use of these methods significantly reduces the amount of SQL queries to be produced from scratch by the developer.

5.11 Future implementation

The previously described methods and classes make up the core functionality of the app, i.e. defining a farm, plantations within the farm and populating the plantations with trees. The next steps in development would include the following:

- Planting guides to assist users in the practical process of planting a tree.
- Interface with the device's calendar or use of the Google calendar API to provide alerts for scheduled maintenance tasks.
- Use of geolocation to define plantations more accurately.
- Interface with the device camera to allow documenting of each plantation in terms of growth or problems such as disease or pest damage.
- Implementation of a website with further, more detailed functionality and to facilitate backup / central storage of user data

5.12 Implementation summary

The process of developing the app to the current stage has seen some significant challenges and learnings for the developer. Refinements and changes were made at several stages and proposed functionality was not brought forward in some cases due to time constraints. It was decided that the core functionality of the app should be developed to the stage where it was as robust as possible in order to provide a strong basis for future development. Emphasis was placed on error handling from an early stage and thought was given to how a user might interact with the interface in a real world situation. The goal was to 'build in quality' at every stage. The following sections will deal with the testing of the app and the evaluation of how well this goal was realised.

6. Testing and Evaluation

6.1 introduction

In line with agile methodology, testing took place throughout the development process, with a heavy emphasis on the analysis of output by the app. To facilitate this there was heavy usage of the logcat system to analyse the output of each stage of the application. The developer at every stage tried to anticipate how the user could miss-operate the app and tried to circumvent this as much as possible. Specific tests were formulated where actual output was compared against expected output Errors were uncovered and fixed during this process and focus group participants were asked to trial the app and to provide feedback on it.

6.2 Debugging

Throughout the development process there was use made of the debugging capabilities of Android Studio. One very important tool is the logcat output of the android monitor. This displays a list of actions that the testing device is carrying out, and the developer can program more output for greater insight into the way the device is handling the app. One such example of where this was invaluable was in testing the passing of a bundle of data from the ListPlantationInfo activity class to the DisplayPlantationMap class. Since some of the values passed where to be used for other purposes than display to the user, such as building a map polygon, it was important that these could be checked both before and after passing via an intent. The following is the log message before passing intent:

```
Log.i(TAG, "passable values "+name+" "+date+" "+capacity+" "+polygon);
```

This demonstrated that the values were being extracted from the list view correctly and a similar log in the target activity confirmed they had been passed with no null values. The android monitor will also provide useful date in terms of where any errors or exceptions originate, displaying an error message and one or more locations in the app's code where functionality could not be completed.

6.3 Resolving failed tests

During the initial testing process some errors were discovered that would lead to user confusion or data corruption by allowing the user to input data inappropriately. The following is a detailed description of how some of these errors were resolved.

- The first of these was that it is intended that users create one farm at this stage. On opening the app it is possible to create a new farm each time. This is unnecessary so it was decided that if a query on the farm table database returns null the create farm button would be available, otherwise it would be greyed out to indicate that this step is no longer possible.
- A plantation with two corner markers could be created. This would allow the creation of a plantation with zero area. This was undesirable. This was remedied by specifying a minimum marker array size of 3 before the drawPlantation method would carry out its functions. If an array size of less than 3 is detected, the user is prompted to place more markers. The enforcement of this was implemented by placing a condition in the drawPlantation method which will only allow the method to complete if the number of markers is more than 3.
- when the user presses the restart button and redraws the plantation on the create plantation screen the area of the plantation was incorrect. This was due to the arrayList being used to store the location of the corner markers not being cleared, thus the area calculation was a summation of the erased plantation and the redrawn plantation. This was remedied by adding a call to the clear function of this arrayList in the clear screen method.
- The user was able to press the done button before a plantation had been defined. This would have entered a null row in the plantation table or created errors where null values are not allowed. Remedied by the declaration of a Boolean variable named plantationDefined. This is initialised to false and is assigned true as the final action of the drawPlantation method. It is reassigned false if the clearScreen method is called.
- When the user reaches the view plantations screen it was possible to return to the create plantations screen and create another plantation with the same name as the one created previously. This was remedied with the inclusion of the onBackPressed method. This would override the operation of the back button and take the user to the name plantation activity instead. A further press will take the user to the home screen.

6.4 Different devices

During the development process the app was tested on several different devices. A very low-budget (£20!) 7.5 inch screen EGL tablet, a Samsung A5 mobile phone, a Samsung Galaxy s3, a Sony Experia Z3, and a Sony Experia Z5 premium. The app performed well on all the devices with no functionality problems related to device variance but some issues were experienced with viewing the app in landscape mode, especially on smaller screens. It was decided at this stage to restrict the app to portrait layout to give a consistent appearance on all devices. A future implementation should include separate layouts for landscape mode, possible utilising fragments to display more information in landscape mode, a map showing all the user farm and all plantations for example, or the user's location if no farm is defined. There are many possibilities.

6.5 Testing

Throughout the development process the methodology of test driven development was used to progress the development. This allows the project to develop in the direction intended by the initial design, rather than the what the developer finds most expedient. Traditional testing methods can often lead to difficult problems being avoided and the moves away from the original design.

With test driven development the failure is anticipated and passing the set test becomes the task rather than creating the functionality.

The following table shows details outcomes of failed tests carried out on the functionality of the Plantree app throughout the development cycle and upon completion.

Test	Pass	Outcome	Resolution	Pass on retest?
Create farm when farm already defined	N	Nothing to stop creation of multiple farms per user	Include datatbase check when button pressed	Y
Search for farm address	N	Incorrect coordinates returned,	Duplication of latitude instead of longitude	Y
Search for farm address with null value	N	Fatal exception, nothing returned	Try-catch on method	Y
Search for farm address with nonsense value	N	Fatal exception , nothing returned	Try-catch on method	Y
Clear screen (CreatePlantation)	N	Cleared markers still affecting shape of new polygons	Store marker objects in an array list and empty when clearing screen	Y
Clear screen (CreatePlantation)	N	New plantation area incorrect after deletion of previous attemot	Empty array list of marker LatLngs used to compute area when clearing screen	Y
Draw plantation with zero markers	N	Null pointer exception	Specify minimum size of marker array with if/else in method.	Y

Draw plantation with one marker	N	Fatal exception	Specify minimum size of marker array with if/else in method.	Y
Draw plantation with two markers	N	Zero area polygon returned	Specify minimum size of marker array with if/else in method.	Y
View plantations (back button pressed)	N	Returns to create plantation screen and user can create duplicate name plantation	Override back button to return to home screen	Y
Show Plantation map	N	Plantation not displayed	Incorrect data types passed by intent, changed types	Y
Add trees to plantation	N	Plantation capacity exceeded with multiple additions	Update plantation capacity with every addition	Y
Create Plantation with no farm defined	N	Fatal error	Include db check when button pressed	Y
Press done plantation when no plantation drawn	N	Fatal error	Introduce Boolean variable plantationDefined, Which must be true before method will run	Y

Search for farm	N	Google API service not found	Known issue android OS issue, restart device resolves.	Y

All the issues causing the above failed tests were resolved with no loss of functionality using the solutions outlined. As they were run on an incremental basis in line with the agile process, once every functionality was troubleshot and proven at its iterative stage there was no need for extensive debugging in the final stages of development, as the apps behaviour at every stage was predictable and robust.

Although the test failure rate seems high, this can be seen as a positive because the developer now knows what is needed in order to progress, and progress can be clearly iterative. (*Introduction to test driven development (TDD)*, 2003). Indeed the main principle of test driven development is that tests must be formulated that the developer knows will fail at the outset, then the code is developed to allow the test to be passed. Most of the testing for this app, although not truly test driven, was anticipated to fail on the first attempt.

6.6 Focus group feedback

The app was provided to the available focus group participants, (3 of the original 10 were unavailable), for review when the core functionality was complete. They were provided with a simple questionnaire to gauge how well the app met their expectations. Overall the group members were impressed with the way the app looked and worked but some suggestions were made for possible improvements:

Group member	Suggestion
1	Some of the writing is quite small, can this be made larger?
2	Some sound feedback when things are pressed would be good.
3	No suggestions
4	A wee video showing how to plant a tree
5	Need to see all the plantations on a map somewhere, and how many trees on each.
6	I have another few fields a bit of a distance away, the viewing map should centre on the woodland, not on the farmyard
7	The shape on the map should not be coloured so you can see the trees growing over the years

All of the suggestion received were valid points that would improve the usability of the app, but time constraints prevented their implementation in this version. The suggestion about the polygon shading being removed had been considered during the process of development but had been overlooked at the time. This seems a good idea as Google tend to update their mapping imagery fairly frequently and a lot of the plantations on one of the test sites were clearly visible from above within 3 years of planting.

In general the focus group participants agreed that the app does not meet the full criteria set out in the original design in that only the core functionality is operative thus far, but on the basis of the quality of this they would be happy to use the finished app. Some of the group were not very experienced in the use of smartphones and were quite surprised at the ability of the app to create

and view the plantations. All of the participants said they would see this app as a good example of the use of technology in agriculture.

7. Conclusion and Recommendations

This chapter will discuss the final outcome of the project and evaluate whether or not the initial objectives set out for the project have been achieved.

7.1 Appraisal

The goal of the project was defined as follows:

"The importance of trees to our ecosystem is becoming more widely understood. There is a need for a straightforward application that would help novice planters plan, plant, and care for their trees."

The dissertation objectives can be summarised as:

- To provide a solution that will assist users in the creation of new woodland areas on their land, to provide the necessary information to encourage this activity and improve the confidence of the novice arborist.
- To research into what solutions already exist, as well as questionnaire led research into what the market requires, using opinions from those who work in the field of forestry and plantation as well as potential users who may be interested in planting trees or who have already done so.
- To develop design concepts for the user interface in terms of layout and functionality as well as graphical design considerations. There will be a discussion of the architecture of the application and how it will work on a software level, and there will be an outline of the testing and evaluation process.

These goals have been achieved to the satisfaction of the developer, a fully functioning prototype application has been produced in line with goals set by research into what the market required. A thorough evaluation of the existing solutions was made and a focus group questionnaire was created to gather the opinions of a cross section of those involved in tree planting. The design for the app was developed from the ground up and a full evaluation was made of the potential platforms tor the app before deciding upon the Android operating system for reasons of market appeal and the versatility of the Android Studio IDE.

The design objectives can be summarised as:

- The app must have a simple and easily navigable interface
- The app must have a consistent appearance throughout
- The app must allow user to plan and manage woodland plantations
- The app must be interruptible and not cause the device to run slowly or crash

A prototype app was developed and tested both by the developer and by the focus group participants. The testing process confirmed the functionality of the app, and that it was easily navigable with a consistent and attractive interface although it must be acknowledged that due to core functionality being developed that there is still some future development required to build an app that fully meets the requirements set out in the dissertation, as so far there is little functionality to allow the future management of the plantations created by the user.

Having said this, the base for future development provided by the current app structure is good. The ability to create, encode, store and decode plantations. Created by the user was a significant milestone in the development process and will allow much more functionality to be added in the future. The future development goals set out

The testing process also led to some further recommendations for improvements. One user though that some of the text on the app was quite small, so the requirement for accessibility of the app may not have been fully realised in this respect. The future implementation requirements are laid out in chapter 5.10.

7.2 Final Summary

The developer feels that overall the project and dissertation has been successful in achieving the set goals. The objective of producing an app to help in the planning and management of woodland plantations has been achieved. The development process was challenging although no functionality was compromised by difficulties that were encountered, with each problem being overcome through logic, testing and setting clear objectives for each level of functionality. The app works well on all devices tested upon and the testing process has resolved any issues with reliability.

References

Can native woodlands help reduce flooding?
Woodland Trust
No Date No author
Accessed 29/12/2015 1100

http://www.forestry.gov.uk/fr/INFD-7T9JL3

Lord Rooker – “Planting trees could help stop flooding”
Harrabin, Roger
BBC.co.uk
21/1/2014
Accessed 29/12/2015
http://www.bbc.co.uk/news/uk-25864631

The impact of rural land management changes on soil hydraulic properties and runoff processes: results from experimental plots in upland UK
M. R. Marshall, C. E. Ballard, Z. L. Frogbrook, I. Solloway, N. McIntyre, B. Reynolds and H. S. Wheater.
Article first published online: 19 APR 2013
Accessed 29/12/15 1130
http://onlinelibrary.wiley.com/doi/10.1002/hyp.9826/abstract

This flood was not only foretold – it was publicly subsidised
The Guardian
George Monbiot
Tuesday 29/12/2015
Accessed 30/12/2015 1305
http://www.theguardian.com/commentisfree/2015/dec/29/deluge-farmers-flood-grouse-moor-drain-land

Trees prevent run off from land
The Woodland Trust
No date no author
Accessed 30/12/2015 1400
http://www.woodlandtrust.org.uk/plant-trees/why-plant-trees/water-management/

Grants and what they cover
Woodlands.co.uk
Accessed 11/01/16 1150
No date no author
http://www.woodlands.co.uk/owning-a-wood/grants/

The State of the UK's Forests, Woods and Trees
Perspectives from the sector
Woodland Trust 2011
No author
Accessed 11/01/16
https://www.woodlandtrust.org.uk/mediafile/100229275/stake-of-uk-forest-report.pdf?cb=58d97f320cab43d78739766e71084f76

How can tree stumps improve agricultural productivity?

Caspar van Vark
Farming in Malawi, Africa
The integration of trees into farms in Africa fell away due to northern ideas that encouraged 'clean' fields. Photograph: Martin Godwin
Caspar van Vark
Thursday 23 May 2013
Accessed 12/01/16 12.00
http://www.theguardian.com/global-development-professionals-network/2013/may/21/agroforestry-farmer-managed-natural-regeneration-global-development

How Trees Can Retain Stormwater Runoff
Dr. James R. Fazio
2010
Accessed 12/01/16 1300
http://www.fs.fed.us/psw/programs/uesd/uep/products/11/800TreeCityUSABulletin_55.pdf
Tree city USA Bulletin no.55

http://www.clickonwales.org/2012/07/farmers-need-to-rediscover-the-virtues-of-trees/
Farmers need to rediscover the virtues of trees
John Osmond
9th July, 2012
Accessed 12/1/15 1500

http://stakeholders.ofcom.org.uk/binaries/research/cmr/cmr15/CMR_UK_2015.pdfCommunications Ofcom
The Communications Market Report
No author
6th August 2015
Accessed 24/08/16 1918

http://www.theguardian.com/environment/2012/dec/06/ash-dieback-infection-sites-double
Ash dieback infection sites have doubled within a month, figures show
Damian Carrington
Thursday 6 December 2012
Accessed 13/01/16 1400

Golden Eagles and an Uplands Crisis
Woodworth, Paddy.
The Irish Times
9/1/16

http://www.thejournal.ie/ireland-forest-area-577152-Aug2012/
Ireland now has the ‘second-smallest’ forest area in Europe
Wade, Jennifer
The Journal.ie
30/8/12

Accessed 15/01/16 15:05

http://www.belfasttelegraph.co.uk/news/environment/tree-disease-thats-spreading-across-northern-ireland-may-be-too-advanced-to-stop-29644929.html
Tree disease that's spreading across Northern Ireland may be too advanced to stop
Stewart, Linda
Belfast Telegraph
09/10/2013
Accessed 15/01/16

https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/208436/auk-2012-25jun13.pdf
Agriculture in the United Kingdom 2012
National Statistics
Department for Environment, Food and Rural Affairs
Department of Agriculture and Rural Development (Northern Ireland)
Welsh Assembly Government, The Department for Rural Affairs and Heritage
The Scottish Government, Rural and Environment Research and Analysis Directorate
Published 2012
Accessed 15/1/15

http://www.irishtimes.com/business/farming-apps-reap-rewards-for-kerrymen-1.1557780
Farming Apps reap rewards for Kerrymen
Worrall JJ
The Irish Times
Published 14/10/13
Accessed 05/08/16 09:40

https://www.theguardian.com/environment/2015/oct/20/hi-tech-agriculture-is-freeing-farmer-from-his-fields
Hi-tech agriculture is freeing the farmer from his fields
Vidal, John
The Guardian
Published 20/10/2015
Accessed 5/8/16 10:00

https://www.woodlandtrust.org.uk/mediafile/100083882/Benefits-of-trees-to-arable-farms-evidence-report.pdf
Benefits of trees
on arable farms
Woodland Trust Report
The woodland trust

Coding for Dummies
Abraham, Nikhil
New Jersey 2015

Android Activity lifecycle diagram

Available at:
https://developer.android.com/reference/android/app/Activity.html
accessed 20/8/2016

http://agilemethodology.org/
Agile methodology - Understanding agile methodology
No author
28/10/2008
Accessed 20/1/16 1300

https://www.atlassian.com/git/tutorials/why-git/git-for-developers
Why GIT for your organisation?
Author Unknown
Date published unknown
Accessed 07/08/16 21:00

Understanding non-functional requirements
Broadcast.orielly.com
Stellman, Andrew
17/2/2010
Accessed 20/1/2015 1340
http://broadcast.oreilly.com/2010/02/nonfunctional-requirements-how.html

A Wild Idea: Making Our Smartphones last longer
The New York Times
Manjoo, Farhad.
12/3/2014
Accessed 21/1/2016 1930
http://www.nytimes.com/2014/03/13/technology/personaltech/the-radical-concept-of-longevity-in-a-smartphone.html?_r=0

Mandel Theo,
The Elements of User Interface Design
John Wiley & Sons, 1997
Chapter 5
http://theomandel.com/wp-content/uploads/2012/07/Mandel-GoldenRules.pdf
Accessed 20/7/2016

Gnome info Icon 
Created by Sebastian Rubio
Commercial usage allowed
http://www.iconarchive.com/show/plateau-icons-by-sbstnblnd/Apps-gnome-info-icon.html

Java Code Conventions
Hommel Scott, King Peter, Naughton Patrick, DeMoney Mike, Kanvera Jonni, Walrath Kathy
No date

Available at
www.literateprogramming.com/javaconv.pdf
accessed 20/7/2016 1330

Introduction to test driven development (TDD) (2003) Available at:
http://agiledata.org/essays/tdd.html
(Accessed: 25th August 2016).

Appendices

Focus group feedback questionnaire results

Member 1

Q	Question	Yes	No	Don't know
1	Do you find the appearance of the app to be professional and attractive?	x		
2	Did you find the app easy to navigate?	x		
3	Did you find the user guide helpful?	x		
4	Did the app meet all your expectations?	x		
5	Do you agree that the app is a good example of smart technology in agriculture?	x		
6	Based on the current functionality, would you consider downloading and using the complete app?	x		

Comments/Suggestions
Some of the writing is quite small, can this be made larger? Nice colours and simple screens. The maps are great.

Member 2

Q	Question	Yes	No	Don't know
1	Do you find the appearance of the app to be professional and attractive?	x		
2	Did you find the app easy to navigate?	x		
3	Did you find the user guide helpful?	x		
4	Did the app meet all your expectations?	x		
5	Do you agree that the app is a good example of smart technology in agriculture?	x		
6	Based on the current functionality, would you consider downloading and using the complete app?	x		

Comments/Suggestions
Some sound feedback when things are pressed would be good.

Member 3

Q	Question	Yes	No	Don't know
1	Do you find the appearance of the app to be professional and attractive?	x		
2	Did you find the app easy to navigate?	x		
3	Did you find the user guide helpful?	x		
4	Did the app meet all your expectations?	x		
5	Do you agree that the app is a good example of smart technology in agriculture?	x		
6	Based on the current functionality, would you consider downloading and using the complete app?	x		

Comments/Suggestions
No suggestions The way you can see the plantations and the farm is really good

Member 4

Q	Question	Yes	No	Don't know
1	Do you find the appearance of the app to be professional and attractive?	x		
2	Did you find the app easy to navigate?	x		
3	Did you find the user guide helpful?	x		
4	Did the app meet all your expectations?		x	
5	Do you agree that the app is a good example of smart technology in agriculture?	x		
6	Based on the current functionality, would you consider downloading and using the complete app?	x		

Comments/Suggestions

No info on how to plant trees A wee video showing how to plant a tree

Member 5

Q	Question	Yes	No	Don't know
1	Do you find the appearance of the app to be professional and attractive?	x		
2	Did you find the app easy to navigate?	x		
3	Did you find the user guide helpful?	x		
4	Did the app meet all your expectations?	x		
5	Do you agree that the app is a good example of smart technology in agriculture?	x		
6	Based on the current functionality, would you consider downloading and using the complete app?	x		

Comments/Suggestions
Need to see all the plantations on a map somewhere, and how many trees on each, otherwise very good

Member 6

Q	Question	Yes	No	Don't know
1	Do you find the appearance of the app to be professional and attractive?	x		
2	Did you find the app easy to navigate?	x		
3	Did you find the user guide helpful?	x		
4	Did the app meet all your expectations?		x	
5	Do you agree that the app is a good example of smart technology in agriculture?	x		
6	Based on the current functionality, would you consider downloading and using the complete app?	x		

Comments/Suggestions
I have another few fields a bit of a distance away, the viewing map should centre on the woodland, not on the farmyard

Member 7

Q	Question	Yes	No	Don't know
1	Do you find the appearance of the app to be professional and attractive?	x		
2	Did you find the app easy to navigate?	x		
3	Did you find the user guide helpful?	x		
4	Did the app meet all your expectations?	X		
5	Do you agree that the app is a good example of smart technology in agriculture?	x		
6	Based on the current functionality, would you consider downloading and using the complete app?	x		

Comments/Suggestions
The shape on the map should not be coloured so you can see the trees growing over the years

Appendix 2

Git log

```
user@user-PC MINGW64 ~/Desktop/Modules/COM814/Plantree (master)
$ git log
commit cc810d8dfc8d4ae7d93832beb36013871fdfffca
Author: Edward Mullan <mullan.edward@gmail.com>
Date:   Thu Sep 1 12:37:57 2016 +0100

    final commit

commit 65d32188c4d00ae0e6db15909e1d54228e519e7f
Author: Edward Mullan <mullan.edward@gmail.com>
Date:   Tue Aug 30 22:11:12 2016 +0100

    update query working

commit ddfe483707d3e4ed0b6a12f62b232d3d56058130
Author: Edward Mullan <mullan.edward@gmail.com>
Date:   Tue Aug 30 15:24:05 2016 +0100

    full database funtionality

commit 77ab30f50e2d9e6e11f3b0ce4e3adc4ce59a315b
Author: Edward Mullan <mullan.edward@gmail.com>
Date:   Tue Aug 30 13:53:07 2016 +0100

    almost done
:...skipping...
commit cc810d8dfc8d4ae7d93832beb36013871fdfffca
Author: Edward Mullan <mullan.edward@gmail.com>
Date:   Thu Sep 1 12:37:57 2016 +0100

    final commit

commit 65d32188c4d00ae0e6db15909e1d54228e519e7f
Author: Edward Mullan <mullan.edward@gmail.com>
Date:   Tue Aug 30 22:11:12 2016 +0100

    update query working

commit ddfe483707d3e4ed0b6a12f62b232d3d56058130
```

```
Author: Edward Mullan <mullan.edward@gmail.com>
Date:   Tue Aug 30 15:24:05 2016 +0100

    full database funtionality

commit 77ab30f50e2d9e6e11f3b0ce4e3adc4ce59a315b
Author: Edward Mullan <mullan.edward@gmail.com>
Date:   Tue Aug 30 13:53:07 2016 +0100

    almost done

commit 56bd6bd3cd7fb7eed11d6a21c9b2535d09d07d48
Author: Edward Mullan <mullan.edward@gmail.com>
Date:   Wed Aug 24 11:13:34 2016 +0100

    Successful accessing of database info

commit e5228d7b5b362e1562352d7db8d8c9318a039cfd
Author: Edward Mullan <mullan.edward@gmail.com>
Date:   Mon Aug 22 18:59:10 2016 +0100

    relational integrity of db secure, core functions complete

commit 95e1a4a12f9e22f83c36940a3c3f22f2ee8a405f
Author: Edward Mullan <mullan.edward@gmail.com>
Date:   Sun Aug 14 00:51:35 2016 +0100

    Area calculation bug fixed

commit 0953e967efa19c729bec5e63d27f37d9ea935508
Author: Edward Mullan <mullan.edward@gmail.com>
Date:   Sat Aug 13 23:56:29 2016 +0100

    user guide added, plantation area calculation complete

commit e6fcdcceddad35cf78ec6d57261c720634e74561
Author: Edward Mullan <mullan.edward@gmail.com>
Date:   Sat Jul 30 22:03:30 2016 +0100

    moving to relevant activities once work completed

commit 1cea522e6aa770db944fea20e57b40c8720f8876
Author: Edward Mullan <mullan.edward@gmail.com>
Date:   Fri Jul 29 14:49:06 2016 +0100
:
```