# Visualising plants and metadata

Final Report for CS39440 Major Project

*Author:* Siôn Griffiths (sig2@aber.ac.uk)
*Supervisor:* Dr. Hannah Dee (hmd1@aber.ac.uk)

18th of April 2016
Version: 1.1 (Draft)

This report was submitted as partial fulfilment of a BEng degree in
Software Engineering (G600)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

# Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name ............................................................

Date ............................................................

# Consent to share this work

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name ............................................................

Date ............................................................

# Acknowledgements

I am grateful to...

I’d like to thank...

# Abstract

Visualising plants and metadata is a project delivering a web-based system which enables the convenient exploration of plant images and associated metadata captured as part of experiments carried out at the National Plant Phenomics Centre(NPPC).

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# Chapter 1

# Background & Objectives

This section should discuss your preparation for the project, including background reading, your analysis of the problem and the process or method you have followed to help structure your work. It is likely that you will reuse part of your outline project specification, but at this point in the project you should have more to talk about.

**Note**:

- All of the sections and text in this example are for illustration purposes. The main Chapters are a good starting point, but the content and actual sections that you include are likely to be different.
- Look at the document on the Structure of the Final Report for additional guidance.

## 1.1 Background

What was your background preparation for the project? What similar systems did you assess? What was your motivation and interest in this project?

## 1.2 Analysis

Taking into account the problem and what you learned from the background work, what was your analysis of the problem? How did your analysis help to decompose the problem into the main tasks that you would undertake? Were there alternative approaches? Why did you choose one approach compared to the alternatives?

There should be a clear statement of the objectives of the work, which you will evaluate at the end of the work.

In most cases, the agreed objectives or requirements will be the result of a compromise between what would ideally have been produced and what was felt to be possible in the time available. A discussion of the process of arriving at the final list is usually appropriate.

## 1.3 Process

Plan driven approaches traditionally associated with software development projects usually expect that all system requirements are understood and collected prior to any further work on design or implementation. A number of factors made such an approach unsuitable for this project, chiefly a lack of domain knowledge made up-front requirement gathering difficult and the requirements themselves were likely to be poorly defined and subject to change. With these considerations in mind it was decided that an agile approach would be best.

A SCRUM-inspired approach was adopted for the project methodology, featuring time-boxed iterations in the form of sprints with regular releases of the software. Work would be tracked in the form of user-stories, the planning and organisation of work would revolve around a defined functionality goal for each sprint and release.

# Chapter 2

# Design

## 2.1 Overall Architecture

MVC - for ease of testing, scalability, separation of concerns, maintainability through familiarity(people know mvc and what to expect), maturity of supporting technologies

3-tier based approach to data layer / service / presentation stuff that may not entirely fit with the mvc pattern

## 2.2 Framework and Programming Language

The sheer range of MVC frameworks available to developers is incredible and the decision of which to use is potentially difficult. It was not within scope to review all the available choices

## 2.3 Tools and third-party services

### 2.3.1 Intellij

### 2.3.2 Git and Github

### 2.3.3 Jira

Jira [4] in an issue tracking and project management tool provided by Atlassian, an Australian software company. It is an industry leading product used by many companies for tracking their projects and the issues within them. Its use on this project was in support of the agile approach to project development, allowing the specification of user stories, development tasks and their inclusions within configurable sprints or development iterations. Figure 2.1 shows the current sprint view in Jira, user stories are grouped into 'lanes' corresponding to their status, allowing a simple way to track the work completed and left to do involved in the current development iteration.

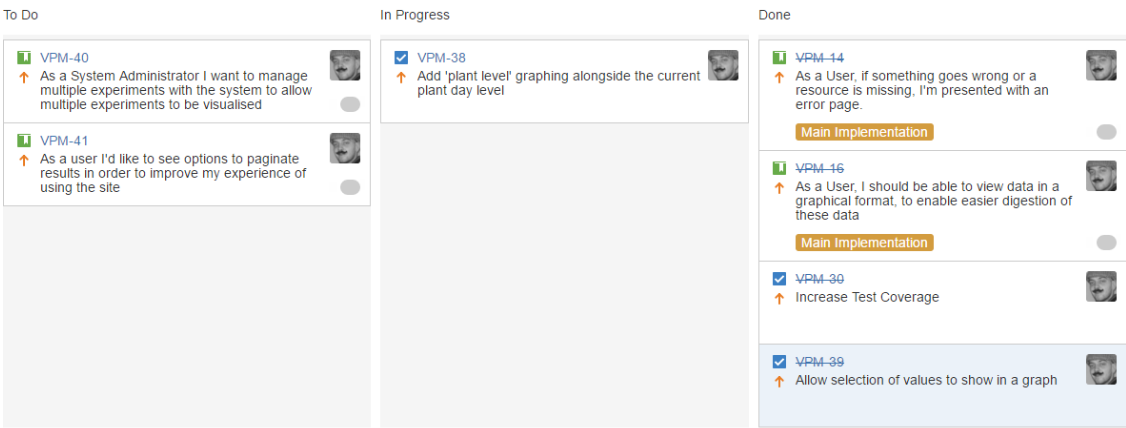


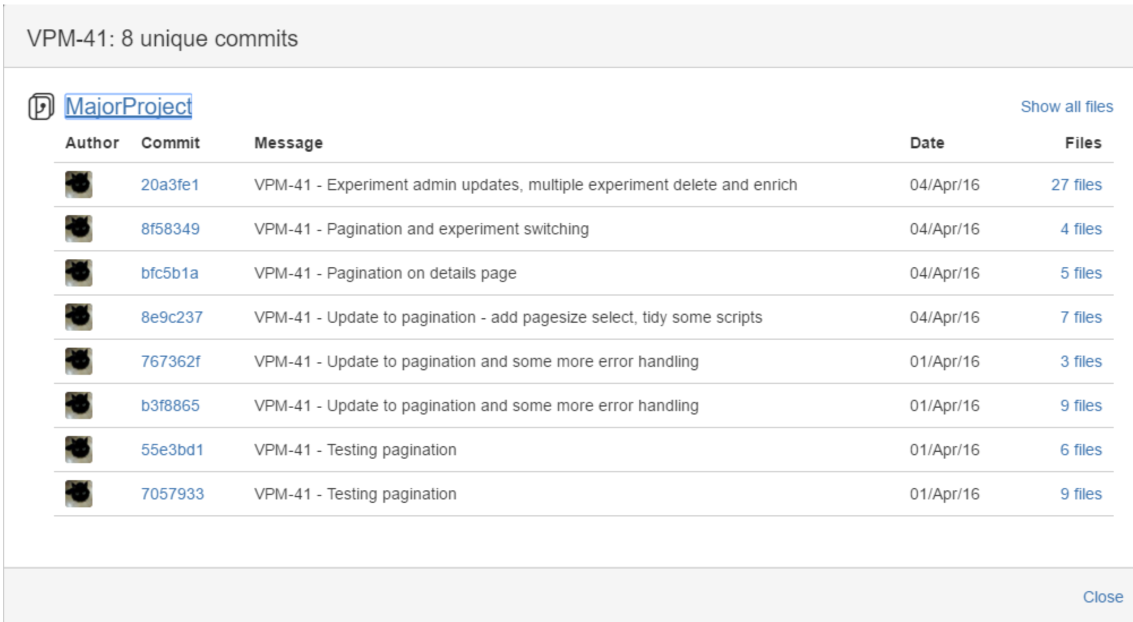

Figure 2.1: A screenshot showing part the current sprint screen in Jira

Bugs could also be tracked as issues within Jira and added to the current sprint if necessary, I found this to be a valuable way to deal with emergent issues during development as it allowed a simple way to assign priority to urgent issues and keep track of less usgent bugs in the project backlog to be worked on in a future sprint. Figure 2.2 shows a selection of bugs raised as part of development, Jira provides simple methods for filtering all issues against a project by type or status allowing quick access to screens such as this.

| T | Key | Summary | Assignee | Reporter | P | Status | Resolution | Created | Updated | Due |
|---|---|---|---|---|---|---|---|---|---|---|
| | VPM-37 | First image for each plantDay is skipped | sion griffiths [Administrator] | sion griffiths [Administrator] | | DONE | Done | 18/Mar/16 | 18/Mar/16 | |
| | VPM-34 | java.net.SocketException: Broken pipe in production system | sion griffiths [Administrator] | sion griffiths [Administrator] | | DONE | Done | 16/Mar/16 | 21/Mar/16 | |
| | VPM-29 | Option to create graph for duplicate attribute appears occaisonaly | sion griffiths [Administrator] | sion griffiths [Administrator] | | DONE | Done | 14/Mar/16 | 18/Mar/16 | |
| | VPM-26 | Ajax replace is replacing div used to target further replacements resulting in failure | sion griffiths [Administrator] | sion griffiths [Administrator] | | DONE | Done | 12/Mar/16 | 12/Mar/16 | |
| | VPM-24 | Adding a tag containing a space results in display and/or db commit issues | sion griffiths [Administrator] | sion griffiths [Administrator] | | DONE | Done | 11/Mar/16 | 12/Mar/16 | |
| | VPM-23 | Adding tag with same text to the same plant day causes exception | sion griffiths [Administrator] | sion griffiths [Administrator] | | DONE | Done | 10/Mar/16 | 10/Mar/16 | |

Figure 2.2: A screenshot showing version control commit tracking within Jira

Another helpful feature was the integration with the version control repository hosted on Github. Referencing the issue ID in Jira in a commit message linked the commits with the issue within Jira. This provided a handy way to track development against particular issues over time and allowed a quick way to navigate between the issues in Jira and the commits on Github.



VPM-41: 8 unique commits

MajorProject Show all files

| Author | Commit | Message | Date | Files |
|---|---|---|---|---|
| | 20a3fe1 | VPM-41 - Experiment admin updates, multiple experiment delete and enrich | 04/Apr/16 | 27 files |
| | 8f58349 | VPM-41 - Pagination and experiment switching | 04/Apr/16 | 4 files |
| | bfc5b1a | VPM-41 - Pagination on details page | 04/Apr/16 | 5 files |
| | 8e9c237 | VPM-41 - Update to pagination - add pagesize select, tidy some scripts | 04/Apr/16 | 7 files |
| | 767362f | VPM-41 - Update to pagination and some more error handling | 01/Apr/16 | 3 files |
| | b3f8865 | VPM-41 - Update to pagination and some more error handling | 01/Apr/16 | 9 files |
| | 55e3bd1 | VPM-41 - Testing pagination | 01/Apr/16 | 6 files |
| | 7057933 | VPM-41 - Testing pagination | 01/Apr/16 | 9 files |

Close

Figure 2.3: A screenshot showing version control commit tracking within Jira

There are a vast array of alternatives that could have been used for issue tracking within the project, many provide the full array of features that were used in Jira during the development of this project. However, Jira being the industry leader, provided an opportunity to gain further valuable experience of its use in a day to day, agile development project. Having previously been involved in the running of a Jira system during my time in industry provided me with familiarisation in configuring a project for my needs and confidence in being able to do so quickly. This was enough to chose Jira over the alternatives that were evaluated such as Waffle.io and the native issue tracking feature provided with Github.

### 2.3.4 Codeship

Codeship [6] is a web based Continuous Integration(CI) service. Working in conjunction with the version control repository, Codeship will detect up any commits made to the repository hosted on Github and execute build and test scripts defined as part of the initial setup of the CI service. Use of a CI system within the project provided assurance that each incremental change made to the system integrated correctly and that all tests continued to pass. A notification would be sent in the event of build or test failure.

The build script for the project can be seen in figure 2.4 showing how the project databases are setup and the environment is configured prior to executing the project build and test commands.

The scripts are invoked within small Docker [3] based environments which allow build dependencies to be modularised and configured quickly. The initial integration of the CI system into the project environment was extremely simple, linking the Github repository for the project was a couple of mouse clicks and the script below is the entirety of the extra configuration required to get the CI system fully up and running.

It was because of this speed and simplicity of configuration that Codeship was chosen over rival offerings such as TravisCI [2] which appeared to have a much more complex initial setup during evaluation.

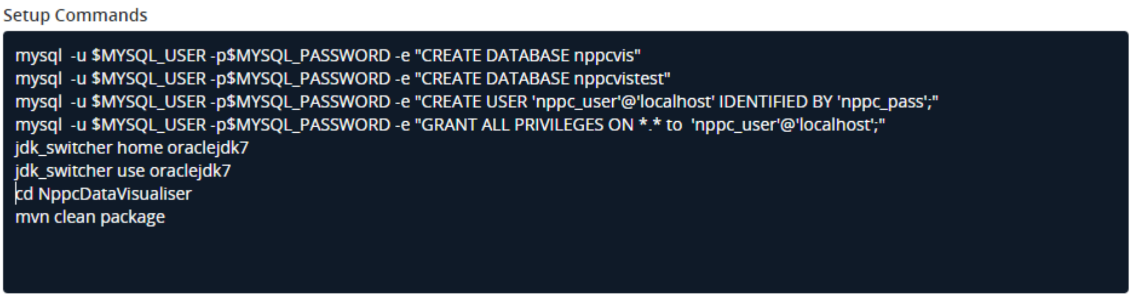



Figure 2.4: The project build script on Codeship


Figure 2.5: A sample of the build history in Codeship

### 2.3.5 Maven

## 2.4 Some detailed design

### 2.4.1 Even more detail

## 2.5 User Interface

## 2.6 Other relevant sections

# Chapter 3

# Implementation

The implementation should look at any issues you encountered as you tried to implement your design. During the work, you might have found that elements of your design were unnecessary or overly complex; perhaps third party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

You can conclude this section by reviewing the end of the implementation stage against the planned requirements.

## 3.1 Stuff

Model plant domain DB building Show plants in page, Data reading and routing via annotated csv Ajax submission of forms -¿ Graphing

# Chapter 4

# Testing

## 4.1 Overall Approach to Testing

The overall approach to testing was to have high test coverage of system features and functionality and to automate these tests wherever it was feasible to do so. Automated tests would run often as part of the normal development workflow and provide continuous assurance of functionality and system environments. Where automation was impractical, alternative approaches were taken to ensure that the system was fully tested in a robust manner.

## 4.2 Automated Testing

For the purposes of automated testing, a separate database was used. The database would be completely recreated for the start of each run of the test suite and dropped at the end. Prior to the tests running, the database would be seeded with test data that is similar to the real world data expected in the production database. By using this method the tests would more closely mirror the real world behaviours of the system and each run could be insulated from the data changes made in previous runs.

A Continuous Integration(CI) system was used in order to facilitate the convenient and regular running of all automated tests in the project. The CI system would build the project from source each time a commit was made into the version control repository. As part of this build process the full test suite would be run. Any issues encountered during this process, from compilation errors to test failures, would result in the build being rejected by the CI system. In the event of a rejected build, the CI system would notify via email of the build failure. This feature turned out to be invaluable since it highlighted a configuration issue that did not affect my local development environment but would have affected the server the project is hosted on. Because the tests were automated and I was notified of a failure, I saved what likely would have been a significant amount of debugging time at the next release of the project to the server. Time was also saved since the full test suite didn't need to be run locally at development time, single tests could be run and the full test and integration suite would be invoked on commit to the version control repository.

### 4.2.1 Unit Tests

When implementing most of the service layer classes for the system a TDD approach was employed in order to ensure high test coverage of the parts of the system which incorporate the business logic. Using TTD helped evolve the design of these service classes by ensuring that nothing was built in a way that was difficult or convoluted to test. Tests are implemented on a method by method basis for the most part, that is, each method in a service will have its own unit test to ensure functionality.

A simple example is shown in listing 4.1 detailing a test for the tags reset functionality in the PlantManager service class that is invoked as part of deleting the data associated with an experiment.

```
    @Test
    public void resetTagsForExperiment()  {
        Long id = 10L;
        Plant plant = plantManager.getPlantByID(id);
        Experiment experiment = plant.getExperiment();

        assertEquals("Expected number of tags to be 2", 2 ,
           plant.getTags().size());

        plantManager.resetTagsForExperiment(experiment);
        assertEquals("Expected number of tags to be 0", 0 ,
           plant.getTags().size());
    }
```

Listing 4.1: Unit test for the PlantManager service

Most of the classes not covered by unit testing are tested via integration testing. The overall coverage for automated tests in the system is 79 % of all lines written in Java.

### 4.2.2 Integration Testing

Integration testing for this project was achieved primarily through testing of the MVC controller classes. The goal behind these tests was to make requests to the various available routes within the system and verify that the correct results are returned. Being a web based system, all functionality is in some way linked to a request mapping or route in a controller class. Testing these routes provides a convenient method to ensure the distinct layers and components that make up the system are working as intended and the interactions between them are as expected.

Integration tests for this project take advantage of features provided by the Spring framework in order to simplify the configuration of the tests and the mocking of certain aspects of the system, such as the application context in which the tests are running. These mocked dependencies and use of the same static data and database each run ensure that results can be verified consistently.

The example test in listing 4.2 shows how a mockMvc object is used to simulate the web application context and perform requests against the application, in this case a HTTP GET to the path '/plants' which should return the plants page. The HTTP session object can be managed as part of the tests and injected into individual requests to ensure compatibility with real world usage. Following the HTTP request the results can be verified, in the case of the example test the HTTP status is checked to ensure that the server returned status code 200 (Ok). The content of the response is verified then finally, a check against the view() method is made to ensure that the correct page has been returned as a result of the request.

```
@Test
public void testShowPlants() throws Exception {
    String testBarCode = "bc1";
    this.mockMvc.perform(get("/plants").sessionAttrs(sessionattr))
                .andDo(print())
                .andExpect(status().isOk())
                .andExpect(content().string(containsString(testBarCode)))
                .andExpect(view().name("plants/show"));
}
```

Listing 4.2: Simple integration test example

A similar approach is adopted for all integration tests throughout the system. For each tested route, the request is simulated and results verified in much the same way as in the example test. In more complex tests or those testing functionality which require more robust verification there are extra steps taken such as asserting the existence of certain page model attributes or objects being passed to the front end views.

### 4.2.3 Stress and Performance Testing

Performance and stress testing was carried out through the use of Apache Jmeter [1], an open source Java application built to measure site and application performance under controlled loads. Jmeter enables the simulation of a number of concurrent users accessing a given site, these simulated agents follow a defined sequence of actions as specified in the test script. Unless otherwise stated the tests run with ten concurrent agents and the tests are repeated thirty times in order to smooth out any outliers in the data.

The tests were all carried out against the project hosted on the remote server provided by Ibers. The machine used to run the Jmeter scripts is a powerful desktop machine using a recent generation of Intel i7 processor featuring 4 cores and able to process 8 concurrent threads. It is necessary that the test machine be connected to the Aberystwyth University VPN in order to reach the target server although the impact of the extra overhead appears minimal and is considered for the sake of comparing results. Although the ten concurrent users may seem low, the throughput on average is over 100 requests per second when run from the test machine which is significantly more than ten real users would be able to generate.

For the purposes of this project Jmeter was used to assess whether pages in the site would load within defined time limits and whether implementation decisions have an adverse effect on performance. In general the goal was to have pages served within 300ms with a hard limit of 1000ms, or one second, although this does not include image load times. A target of 300ms is well under the 1 second limit for keeping a users flow of though as identified as part of a study conducted by Nielsen [11]. Running the tests regularly could also help highlight issues that may not be uncovered under other forms of testing such as intermittent problems that could result in request errors that would be difficult to reproduce otherwise.

General results as output by Jmeter are included in figure 4.1 for an experiment which has been initialised with data. The initialisation is an important distinction because the amount of experiment data significantly affects the initial page response time for the Graphs page, other pages are affected somewhat but to a much lesser degree. Figure 4.2 displays the results of running the same test without the data having been added to the experiment and it's clear to see the effect on the load time for the Graphs page. Having these results available during development informed some of the design choices within the Graph page such as having the graphs themselves and any plant objects loaded via Ajax following user interaction as opposed to being populated into the page on load.

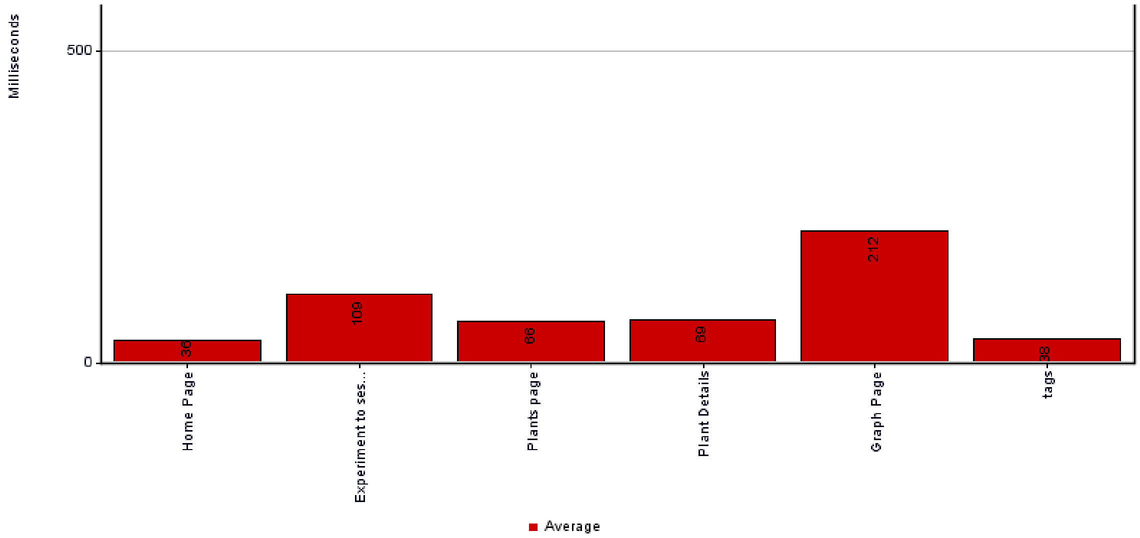



Figure 4.1: Visulisation of Jmeter test result of a fully initialised experiment

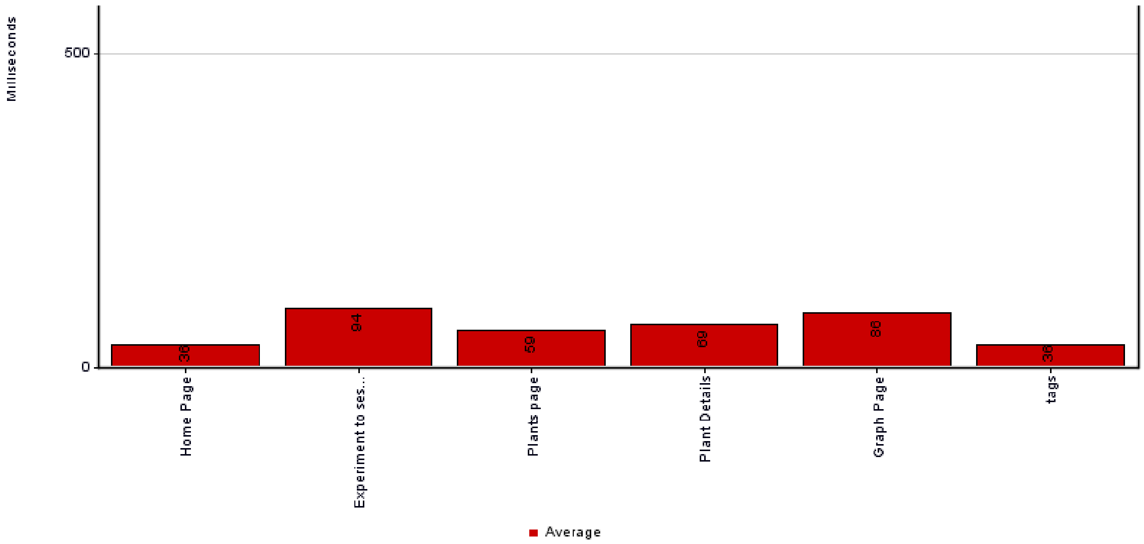



Figure 4.2: Visulisation of Jmeter test result of a partially initialised experiment

The effect of choosing pagination defaults for the plants and plant detail pages could also be measured although in the case of pagination the real limiting factor is the bandwidth and render time required. However, the effect on request time and loading on the server could be seen and monitored for any potential issues. In an experiment with many plants or a large amount of data the response time would increase significantly with page sizes of over 50 or so plants but no other adverse affects were noticed on the system even with a significant number of requests.

## 4.3 Manual Testing

For areas of the system where automated testing was impractical or insufficient to verify results, a manual approach was taken and test tables used to verify functionality is as expected.

### 4.3.1 Admin Page Test Table

Much of the functionality on the Admin page relies on an active network connection to the NPPC data repository and as such is unsuitable for automated testing. There was no feasible way to establish a connection between the continous integration server and the NPPC data repository therefore the manual verification of functionality is necessary.

| **Test** | **Input** | **Expected Output** | **Pass** |
|---|---|---|---|
| Attempt to access admin area without login | Go to /admin without login | Redirected to administrator login page | ✓ |
| Attempt to access admin area with correct login | Go to /admin with login | Admin is page is displayed | ✓ |
| Attempt admin login with incorrect credentials | Submit admin login form with incorrect credentials | Error displayed to user. | ✓ |
| Admin log out | Click logout button from admin page | Redirect to home page and authorisation cleared from session | ✓ |
| Initialise Experiment | Click initialise button for uninitialised experiment | Experiment begins initialising - plants are created | ✓ |
| Update experiment | Click Update button on initialised experiment | Experiment begins update, plants are updated or created | ✓ |
| Import data with valid csv | Click Init Data button on initialised experiment | Data is imported from csv | ✓ |
| Import data with invalid csv | Click Init Data button on initialised experiment | Invalid csv data is ignored | ✓ |
| Delete data | Click delete data on experiment | Data is deleted from the experiment | ✓ |
| Delete plants | Click delete plants button on experiment | Plant data and images are deleted | ✓ |

Table 4.1: Test Table for Admin page functionality

### 4.3.2 Graph Page Test Table

Although most of the functionality within the Graph page is verified via automated testing, certain aspects require visual verification and as such a manual approach is taken to verify functionality within the page.

| Test | Input | Expected Output | Pass |
|---|---|---|---|
| Test view graphs with no experiment | Go to /graphs with no selected experiment | No data' page is show with back button | ✓ |
| Test view graphs with experiment that has no data | Go to /graphs with experiment in session that has no data | No data' page is show with back button | ✓ |
| Test view graphs with experiment that has data | Go to /graphs with experiment in session that has data | Graph page is shown with graph creation options | ✓ |
| Test create graph | Click create graph button on /graphs page | A graph is displayed in the page with selected axis attributes | ✓ |
| Test box plot | Select 'Box' and create graph | Nodes in the graph are represented as box plots | ✓ |
| Test scatter plot | Select 'Scatter' and create graph | Nodes in the graph are represented as scatter plot | ✓ |
| Test swap axis | Click swap axis button | Selected axis attributes are swapped, x value becomes y value and vice versa | ✓ |
| Test plant results on graph node click | Click on or near a node in the graph | A clickable list of plants corresponding to the values of the clicked node appear in the page | ✓ |
| Test click result plant | Click on a plant link generated as result of clicking on a graph node | User is redirected to the detail page for the clicked plant link | ✓ |

Table 4.2: Test Table for Graph page functionality

## 4.4 User Testing

When development was near complete a small sample of volunteer test users were recruited to use the system and give feedback on usability and the system in general. An online form was provided with a number of questions and a section for general feedback the responses to which can be found in Appendix

Following the user testing, a number of changes were implemented according to the feedback given. Namely, adding pagination controls to the bottom of the plants and plant detail pages for more convenient page navigation and fixing an overlooked issue on the plant detail graph page. If a plant has no attributes recorded against individual plant days then the page should make the user aware that graph generation is not possible and provide a means to return to the previous page. Prior to user testing the page was confusing and mostly blank in the event that no graphable data was available.

# Chapter 5

# Evaluation

Examiners expect to find in your dissertation a section addressing such questions as:

- Were the requirements correctly identified?
- Were the design decisions correct?
- Could a more suitable set of tools have been chosen?
- How well did the software meet the needs of those who were expecting to use it?
- How well were any other project aims achieved?
- If you were starting again, what would you do differently?

Such material is regarded as an important part of the dissertation; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things and room for improvement with any project. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

Review the discussion on the Evaluation section from the lectures. A recording is available on Blackboard.

# Appendices

# Appendix A

# Third-Party Code and Libraries

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. The key requirement is that we understand what is your original work and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

As an example, you might include a definition such as:

Apache POI library The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the clients existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation [**?**]. The library is released using the Apache License [**?**]. This library was used without modification.

# Appendix B

# Ethics Submission

This appendix includes a copy of the ethics submission for the project. After you have completed your Ethics submission, you will receive a PDF with a summary of the comments. That document should be embedded in this report, either as images, an embedded PDF or as copied text. The content should also include the Ethics Application Number that you receive.

# Appendix C

# Code Examples

## 3.1 Random Number Generator

The Bayes Durham Shuffle ensures that the psuedo random numbers used in the simulation are further shuffled, ensuring minimal correlation between subsequent random outputs [**?**].

# Annotated Bibliography


[1] "Apache JMeter - Apache JMeter." [Online]. Available: http://jmeter.apache.org/

An open-source Java based performance and load testing tool originally designed for web applications.

[2] "Travis CI User Documentation." [Online]. Available: https://docs.travis-ci.com/

Travic continnuous integration documentation

[3] "What is Docker?" May 2015. [Online]. Available: https://www.docker.com/what-docker

Docker continuous integration system overview

[4] Atlassian, "JIRA Software - Issue & Project Tracking for Software Teams." [Online]. Available: https://www.atlassian.com/software/jira

[5] R. Boyle, F. Corke, and C. Howarth, "Image-based estimation of oat panicle development using local texture patterns," *Functional Plant Biology*, vol. 42, no. 5, p. 433, 2015. [Online]. Available: http://www.publish.csiro.au/?paper=FP14056

Paper detailing a technique used to detect oat panicles via computer vision techniques. Development of panicles can be directly corrolated with certain growth stage (around GS55) in oats

[6] "Codeship," Codeship Inc. [Online]. Available: https://codeship.com/

A continuous integration tool which can hook into other online resources such as GitHub

[7] "Build software better, together," GitHub, Inc. [Online]. Available: https://github.com

An online Git repository hosting service

[8] D. Kendal, C. E. Hauser, G. E. Garrard, S. Jellinek, K. M. Giljohann, and J. L. Moore, "Quantifying Plant Colour and Colour Difference as Perceived by Humans Using Digital Images," *PLoS ONE*, vol. 8, no. 8, p. e72296, Aug. 2013. [Online]. Available: http://dx.doi.org/10.1371/journal.pone.0072296

Paper detailing how a humans perception of colour in images of plants can affect judgements made about these images.

[9] M. N. Merzlyak, A. A. Gitelson, O. B. Chivkunova, and V. Y. Rakitin, “Non-destructive optical detection of pigment changes during leaf senescence and fruit ripening,” *Physiologia Plantarum*, vol. 106, no. 1, pp. 135–141, May 1999. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1034/j.1399-3054.1999.106119.x/abstract

Paper detailing senescence detection in plant images by analysis of colour

[10] “National Plant Phenomics Centre,” National Plant Phenomics Centre. [Online]. Available: http://www.plant-phenomics.ac.uk/en

National Plant Phenomics Centre

[11] J. Nielsen, “Response times: the three important limits,” 1994.

Article discussing tollerable wait times for web page loads

[12] J. R. Quinlan, “Induction of Decision Trees,” *Mach Learn*, vol. 1, no. 1, pp. 81–106, Mar. 1986. [Online]. Available: http://link.springer.com/article/10.1023/A%3A1022643204877

Paper detailing the ID3 decision tree algorithm

[13] J. M. Tanner, R. H. Whitehouse, W. A. Marshall, M. J. R. Healty, and H. Goldstein, “Assessment of Skeleton Maturity and Maturity and Prediction of Adult Height (TW2 Method),” 1975. [Online]. Available: http://core.tdar.org/document/125299

Paper detailing the atlas approach used in this instance to predict adult height in human’s from skeletal features in children

[14] G. W. Williams, “Comparing the Joint Agreement of Several Raters with Another Rater,” *Biometrics*, vol. 32, no. 3, pp. 619–627, 1976. [Online]. Available: http://www.jstor.org/stable/2529750

A paper describing the comparison of expert opinion with that of other experts or a group of experts

[15] J. C. Zadoks, T. T. Chang, C. F. Konzak, and others, “A decimal code for the growth stages of cereals,” *Weed res*, vol. 14, no. 6, pp. 415–421, 1974. [Online]. Available: http://old.ibpdev.net/sites/default/files/zadoks_scale_1974.pdf

Paper detailing the decimal growth stages of cereal plants