EXTENDING AWESOME

Final Report for CS39440 Major Project

Author:
Benjamin Brooks
beb12@aber.ac.uk

Supervisor:
Dr. Hannah Dee
hmd1@aber.ac.uk

Friday 8th May, 2015

This report was submitted as partial fulfilment
of a BSc degree in Computer Science
(inc Integrated Industrial and Professional Training) **[G401]**

Prifysgol Aberystwyth University

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 plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the sections on unfair practice in the Students' Examinations Hand- book and the relevant sections of the current Student Handbook of the Department of Computer Science.
- I understand and agree to abide by the University's regulations governing these issues.

Signature _____ 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.

Signature _____ Date _____

Ethics Form Application Number

The Ethics Form Application Number for this project is: **1019**.

Student Number

Many thanks to my supervisor Hannah Dee for the guidance and support throughout this project; Sandy for providing a server, and support when it went wrong; and the rest of the staff at Aberystwyth University for the countless hours they put into making it an outstanding place to study.

Thanks to Gareth Williams for providing better Welsh translations than Google Translate ever could, and thanks Keiron O'Shea for the AWESOME prototype upon which this project is based.

Thank you to my fellow classmates, flatmates, friends, and family for the continued motivation, ideas, and encouragement; not only over the past few months, but also the past four years through my university course.

But most of all thank you to Bethany Foskett, for being there with sugary snacks, caffeinated beverages, and occasionally being my rubber duck [1].

Abstract

The Aberystwyth Web Evaluation Of Module Experiences (AWESOME) is a tool used to evaluate the performance and quality of teaching of modules at Aberystwyth University. The aims of this project are to ensure the prototype version of AWESOME is security audited, refactored to use better programming practices, fully functioning, and is deployed to capture module evaluation data.

Contents

List of Tables

List of Figures

List of Acronyms

ASTRA	Aberystwyth Student Records and Admissions
AU	Aberystwyth University
AWESOME	The Aberystwyth Web Evaluation Of Module Experiences
CI	Continous Integration
CSV	Comma-Separated Values
i18n	Internationalisation
MEQ	Module Evaluation Questionnaires
MVC	Model-view-controller
OOP	Object-oriented programming
PDO	PHP Data Objects
RDBMS	Relational Database Management System
SME	Student Module Evaluation
SQL	Structured Query Language
SSCC	Staff Student Consultative Committees

1. Background & Objectives

1.1. Introduction

The Aberystwyth Web Evaluation Of Module Experiences (AWESOME) is a web-based tool that enables departments to generate personalised questionnaires that get sent out to students to gather feedback about modules, lecturers, or even events such as BCS Show & Tell[1] and other talks.

1.2. Background

Departments at Aberystwyth currently have no formalised process of gathering student feedback unlike many other universities. The University of Sussex requires courses to be evaluated through their Module Evaluation Questionnaires (MEQ) [2] which contains seven core quantitative questions and up to ten additional questions at the school-level, module-level or a mixture of both. The University of Westminster have Student Module Evaluation (SME) [3], which is an online questionnaire containing ten questions per module sent via e-mail about modules. There are many other similar examples of module evaluation systems across UK universities, with their main features being personalised, anonymised, and incentivising completion.

At Aberystwyth, some departments such as Geography & Earth Science, and English choose to hand out paper-based questionnaires in lectures for each module a student does. Having a student take multiple questionnaires to provide essential feedback can lead to many issues due to survey fatigue. This can significantly reduce response rates and also have an impact on the answers students provide. The Computer Science department has tried many methods of collecting module feedback with varying successes which will be elaborated upon in subsection 1.2.1.

AWESOME was originally proposed and developed under the Learning and Teaching Enhancement Fund by Dr. Hannah Dee, and work on the prototype was undertaken by Keiron O'Shea. The project was selected as my dissertation project to extend and implement.

[1] BCS Mid Wales Show & Tell: http://midwales.bcs.org/show-and-tell-events

Several meetings by the Learning and Teaching Enhancement Committee and Pro-Vice-Chancellor Professor J. Grattan took place to discuss the future of AWESOME and the need for a module evaluation system used university-wide and talks are still ongoing.

1.2.1. Module Evaluation Method Analysis

There are several important factors to consider when collecting module evaluation feedback. First and foremost, responses must be anonymised, secondly being able to provide incentives for completion drastically increases response rates. Being able to have students complete the survey in their own time, and also send targeted reminders to complete the survey are also important. If reminders are constantly being sent to students who have already completed the survey, they are likely to get frustrated or annoyed and are less likely to notice another survey e-mail. Consolidated surveys help with fatigue, as students only have to answer one survey. There have been several studies on how survey fatigue affects response rates and poor answer quality [4]. This consolidation only works if the surveys are personalised, as seen from Google Forms response rates in Table 1.1, students are lazy, and they will not skip over modules if asked to.

Table 1.1 shows a feature comparison of current methods of module evaluation at Aberystwyth University and corresponding response rates.

Paper based module feedback during lecture time can have effective response rates because lecture-time is set aside to complete questionnaires. However students usually have to complete one questionnaire per module which can lead to fatigue very quickly.

Qwizdom[2] is a hardware-based voting system, with 'clickers' handed to students in a lecture who can then cast their votes through a powerpoint style questionnaire. Response rates for Qwizdom module evaluations are high for the same reasons as paper based forms. Students are stuck in a lecture for an hour with nothing else to do. One problem Qwizdom does give, is that answers can only be quantitative and not qualitative. Students can't easily input textual comments through Qwizdom so a lot of valuable information is not gathered from students.

Google Forms has been the standard way of running module evaluation questionnaires for the past few years in CompSci. Google Forms provides anonymous answering in the student's own time and can also provide a way of gathering valuable textual comments. One disadvantage of using Google Forms is that there is no way of knowing which modules a student

[2]Qwizdom Homepage: http://qwizdom.com/higher-education/home

is enrolled for, so students have to skip over modules that aren't applicable. This makes the survey confusing and error prone at times and response rates suffer as a result.

AWESOME has been created from the ground-up to address all of these problems. Responses are anonymised, while retaining the ability to see who has, and has not completed the questionnaire yet. This allows for targeted reminders and incentives for completing the survey. Additionally, AWESOME imports data directly from Aberystwyth Student Records and Admissions (ASTRA) which allows all student, staff, and module data to be easily used without lots of manual data entry. This also allows for personalised surveys, asking questions only relevant to modules a particular student is enrolled for. By collecting both Quantitive and Qualitative data, AWESOME can run advanced analytics can be run on the data gathered. The questionnaires sent out are also fully responsive, working on phones to tablets, and to desktop computers by utilising Bootstrap and can be completed at any time by the student.

Method	Tailored Questions	Anonymous	Qualitative	Quantitative	Incentives for completion	Completion on own time	Targeted reminders	Responsive	Consolidated	Response Rate
Paper	✗	†	✓	‡	✓	✗	-	-	✗	75%[5]
Qwizdom	✗	✓	✗	✓	✓	✗	-	✓	✗	50%[5]
Google Forms	✗	✓	✓	✓	✗	✓	✗	✓	✓	20%
AWESOME	✓	✓	✓	✓	✓	✓	✓	✓	✓	TBC

† Anonymity may be compromised when completing paper-based form.

Table 1.1.: Module Evaluation methods at Aberystwyth University
‡ Manual processing is required in order to analyse the data.

[2]Bootstrap Homepage: http://getbootstrap.com

1.3. Objectives

The overall objective of the project is to implement AWESOME on the Aberystwyth University network and collect module evaluation feedback for the Computer Science department in both short mid-term, pre-Staff Student Consultative Committees (SSCC) questionnaires and comprehensive end-of-semester questionnaires.

This can be broken down into four main aims:

- **Security Audit the AWESOME prototype.** This was a large issue, as there were known security flaws with the prototype and it needed to be looked at immediately before any other work took place.
- **Bring the prototype up to modern development standards.** The program was known to be written in a procedural style, and the security audit brought up the poor i18n implementation too.
- **Finish any incomplete functionality.** Many areas of the prototype were half-implemented, but not fully completed. These had to be done before it was useable by students and staff.
- **Run AWESOME on a departmental server.** The main objective was to get a survey sent out to students and collect real-world data. This is the final step in that process.

2. Requirements

2.1. Features

The following feature list is the one proposed for the AWESOME prototype. The project follows this feature set as a baseline set of requirements.

- **Automatic questionnaire generation per-student** - Generate unique questionnaires per-student, depending on their modules and lecturers.
- **The ability to generate quick mid-term questionnaires** - A one question per module survey with a one to five scale from 'This module is going well' to 'This module has problems'.
- **No need to type in registration details** - Import of module registration data via ASTRA Comma-Separated Values (CSV) export.
- **Targeted follow-up reminder emails** - Only send reminder emails to respondents who have yet to complete their questionnaire.
- **Anonymous responses** - Ability to know which particular student has, or has not completed the questionnaire, but not who has said what.
- **Visually appealing analytics** - Reports available to staff on a by-module, by-department, and by-scheme basis, with graphs and textual responses laid out nicely.

2.2. Development Practices

AWESOME should be developed using Object-oriented programming (OOP) practices if it is to be maintainable by other developers after the project is over. Internationalisation should be easily extensible and translation strings should be easy to add. Additionally, unit testing is vital if the application is going to be extended further and refactored at a later date for any reason.

2.3. User Roles

The user roles in AWESOME are quite simple, there is only an admin and a respondent and each one can only do a few tasks.

- **Admin** - The person who creates the surveys, adds questions, and sends them. They can also view results and see who has not yet completed the questionnaire.
- **Respondent** - The recipient of a questionnaire email who fills it in and submits answers. They can also submit feedback about AWESOME if debug mode is enabled.

2.3.1. Use Cases

Admin User Role

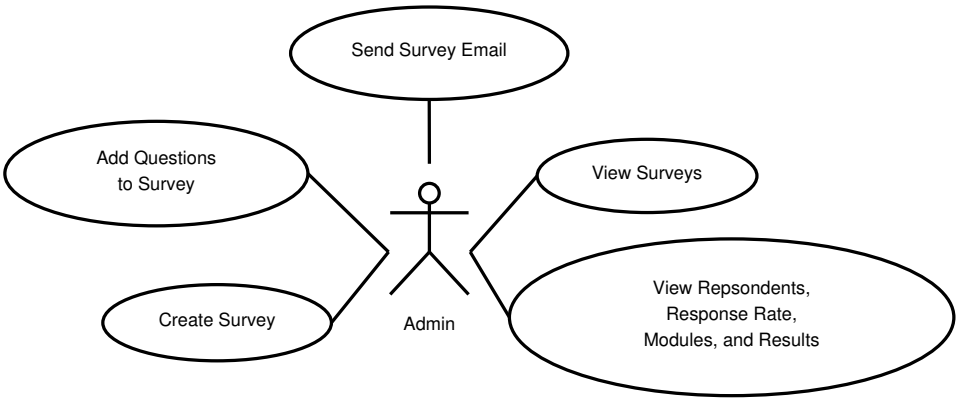


Figure 2.1.: Admin use-case diagram.

Figure 2.1 shows that admins can view surveys, create surveys, add questions to surveys, view more detail about survey such as respondents, response rate, modules, and results. They can also send an email to respondents to remind them to fill in the questionnaire.

Respondent User Role

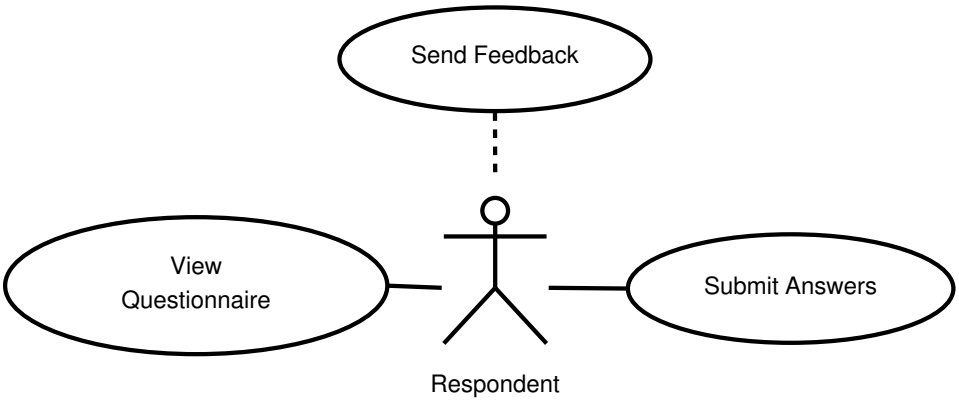


Figure 2.2.: Respondent use-case diagram.

section 2.3.1 shows that a respondent only has one responsibility in the system, and that is to respond to a survey through a link sent via email. They can only respond once, and submitted answers cannot be seen as it would break the irreversible anonymity of the system.

3. Design

After the security audit, it was apparent that the majority of the codebase needed to be refactored in order to make it object-oriented. The decision to rewrite the program was not a light one, by rewriting the program, the scope of extensions to AWESOME became much more limited via time constraints.

With the rewrite came the opportunity to utilise both a framework and a design pattern. Frameworks were ruled out under the stance that it had to be deployed on a server without shell access. However later research discovered that this is achievable in Laravel[1], which would have been the framework of choice for several reasons.

Laravel would have provided a complete, and mature Model-view-controller (MVC) framework, as well as an i18n framework and a solid basis for AWESOME. Ultimately, poor research into Laravel on a shell-less server resulted in an attempt to produce a bespoke MVC framework, which, unsurprisingly is harder than it first seems.

3.1. Programming Language

The programming language choice was fairly fixed, as it had to be easily runnable on an AU server. This limited the choice to PHP, but which minor version of PHP was only found out later in the project which is discussed further in **??**.

If this requirement wasn't an issue, Ruby on Rails would have been a great candidate for this project, as it already provides an MVC framework, gems for i18n, and many other features.

[1] Laravel Homepage: http://laravel.com

3.2. Model-view-controller Framework

Since using a framework was ruled out, a custom-made MVC framework was written to support this project. The design of which is laid out in this section and is very heavily taken from the 'Write your own PHP MVC Framework' series of tutorials[6].

3.2.1. Routing

URL Routing is a way to access specific controllers and views from a URL. This works via an Apache rewrite rule, which appends a $_GET variable which contains the current requested URL.

The URL routing is handled as such: *awesome.url/controller/view[/query]*

3.2.2. Directory Structure

Figure 3.1 shows the directory structure of the MVC framework. '*src*' is the source code to the program. Inside it contains everything needed to run AWESOME. Inside '*tests*' are the unit tests to be run automatically by TravisCI and PHPUnit.

'*src/app*' has the MVC classes which are used when the routing engine rewrites URLs. '*src/config*' contains the config file for the program. In this file, database credentials are set, as well as SMTP mail server settings, some i18n settings, and the debug flag which displays errors and shows a feedback form/notice (See Figure 3.8).

'*src/db*' is a directory for Structured Query Language (SQL) dumps for the database schema. '*src/i18n*' contains translation files for the i18n framework, but more detail is in section 4.4

'*src/lib*' is the 'glue' that holds the MVC framework together. It contains the autoloader class, the bootstrap script and the third-party, open source PHPMailer[7] classes used for sending mail via SMTP.

'*src/logs*' is where any PHP errors get logged when debug mode is off. This is in order to hide any potential security issues with displaying errors.

'*src/public*' is the main entry point to the program and where the Apache vhost will be set. '*src/public/assets*' contains the Javascript, SASS, CSS, and images used in the HTML. Users with a valid token will be taken to their corresponding questionnaire. '*src/public/admin*' is protected by HTTP authentication to prevent anybody from creating surveys and sending them out, or reading the results of previous surveys.

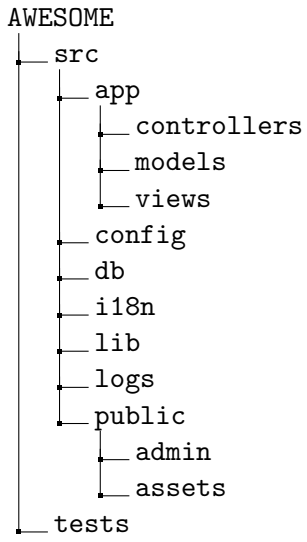


Figure 3.1.: Directory structure of the MVC Framework

3.3. Internationalisation Framework

The i18n framework is a standalone OOP module which utilises JSON formatted files with strings for translation.

The JSON files are structured like the one below. The first two entries are i18n metadata, for ISO639-1 code[2], and language. Each string has an ID, and a translation string to be returned. Every language uses the same ID, and what gets returned depends on the language set via a global variable.

```
{
    "@ISO639-1" : "en",
    "@lang" : "English",
    "invalid-url" : "The URL provided is incorrect.",
    "send-responses" : "Send Responses"
}
```

The i18n class has a global function, __($string_id) which will then return the appropriate translated string depending on the $lang global variable.

[2]ISO639-1: http://en.wikipedia.org/wiki/ISO_639-1

3.4. Database Schema

The database schema (see Figure 3.2) used is very similar to the prototype version. Documentation needed to be written for the existing schema, so the opportunity was taken to make some changes in the structure of the database before this happened. The old database schema did not use any foreign key constraints, nor was the use of primary keys ideal.

Foreign keys are very useful to have in a system like this, as orphaned rows can be cleaned up nicely with a cascade delete. This prevents any data remaining in the table incase it is missed by an SQL query that hasn't been updated in the code.

I also made some changes to the way things were named, mostly to prevent confusion. In the new schema, a survey is a set of questionnaires, each of which is tailored to a specific student. In the old schema, everything was called a questionnaire, which led to confusion between a group of questionnaires and a single questionnaire.

Additional changes need to be made in order for AWESOME to support multiple departments across the university, but this is out of the scope of the dissertation, although this feature is on the roadmap if AWESOME is to be used university-wide.

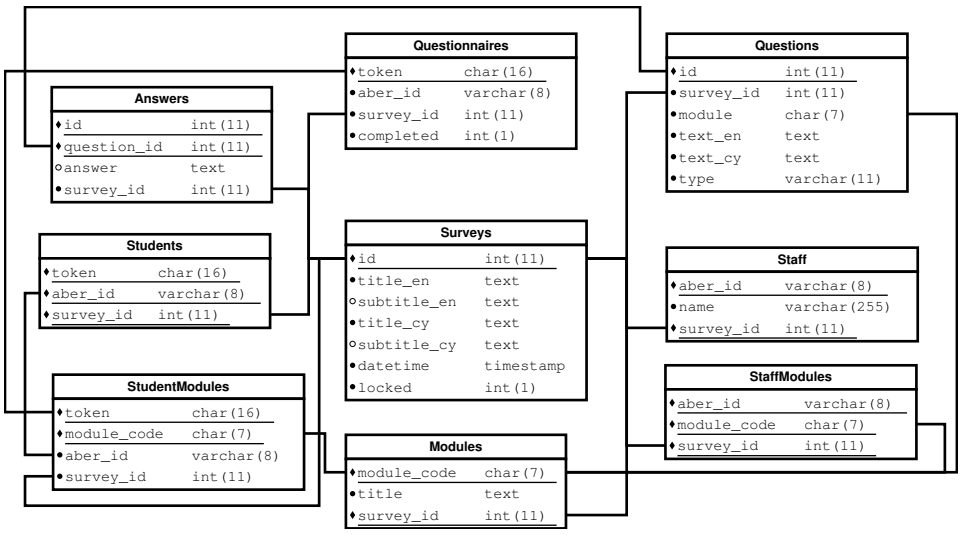


Figure 3.2.: AWESOME database schema design.

3.5. User Interface

The user interface was fairly good on the questionnaire-side in the prototype, so a lot of elements were re-used in that, with a few minor changes.

Figure 3.3 to Figure 3.7 shows a comparison between the prototype version and the submitted version of AWESOME.

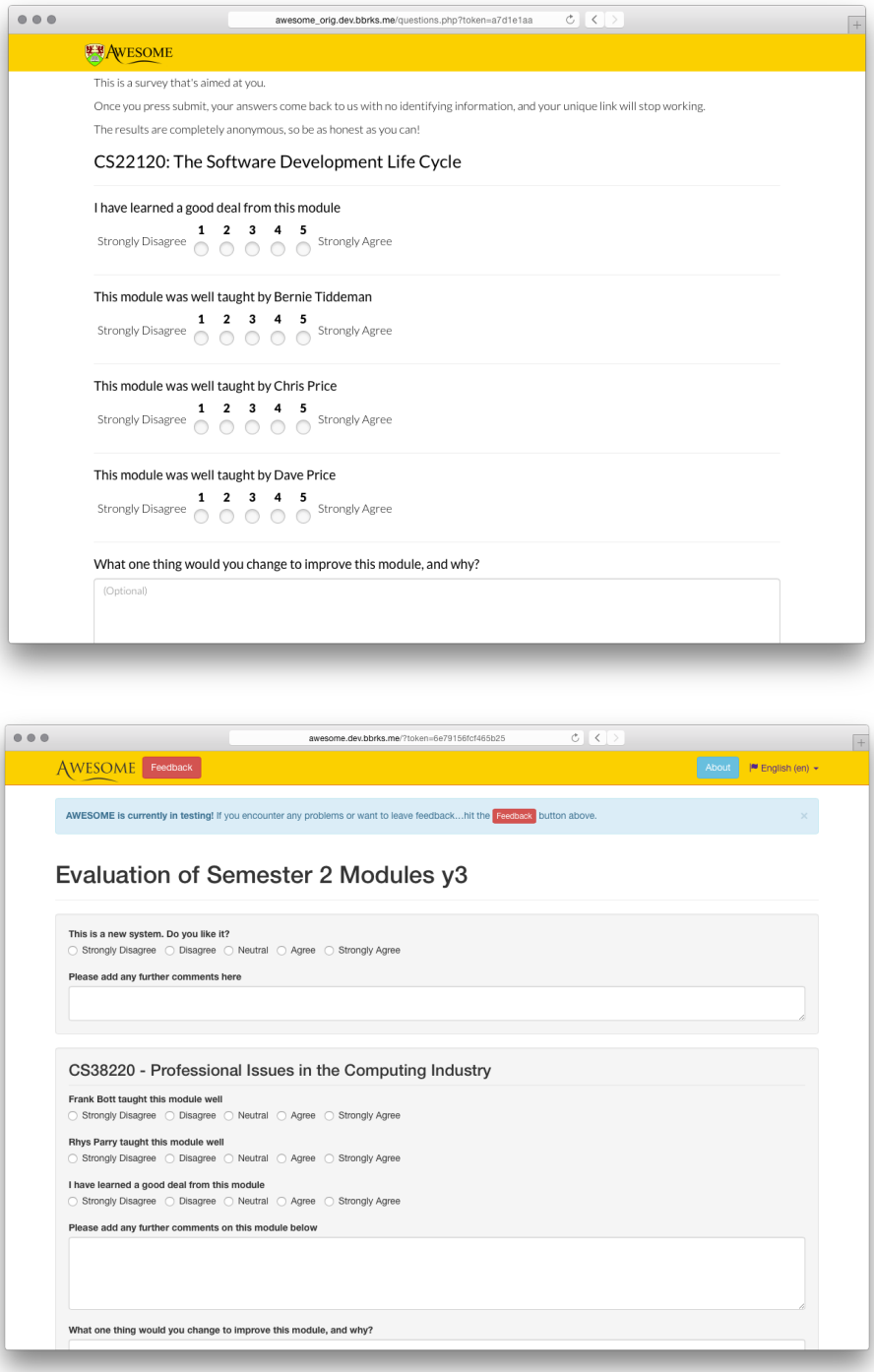


Figure 3.3.: A comparison of questionnaire pages between the prototype version and the submitted version of AWESOME

The admin dashboard of the prototype wasn't very user friendly. Creating surveys, adding questions, modules, students, and staff was fairly convoluted. Results weren't clear to look at, and took up a lot of space per-question by pie charts, which were unnecessary given the data provided.

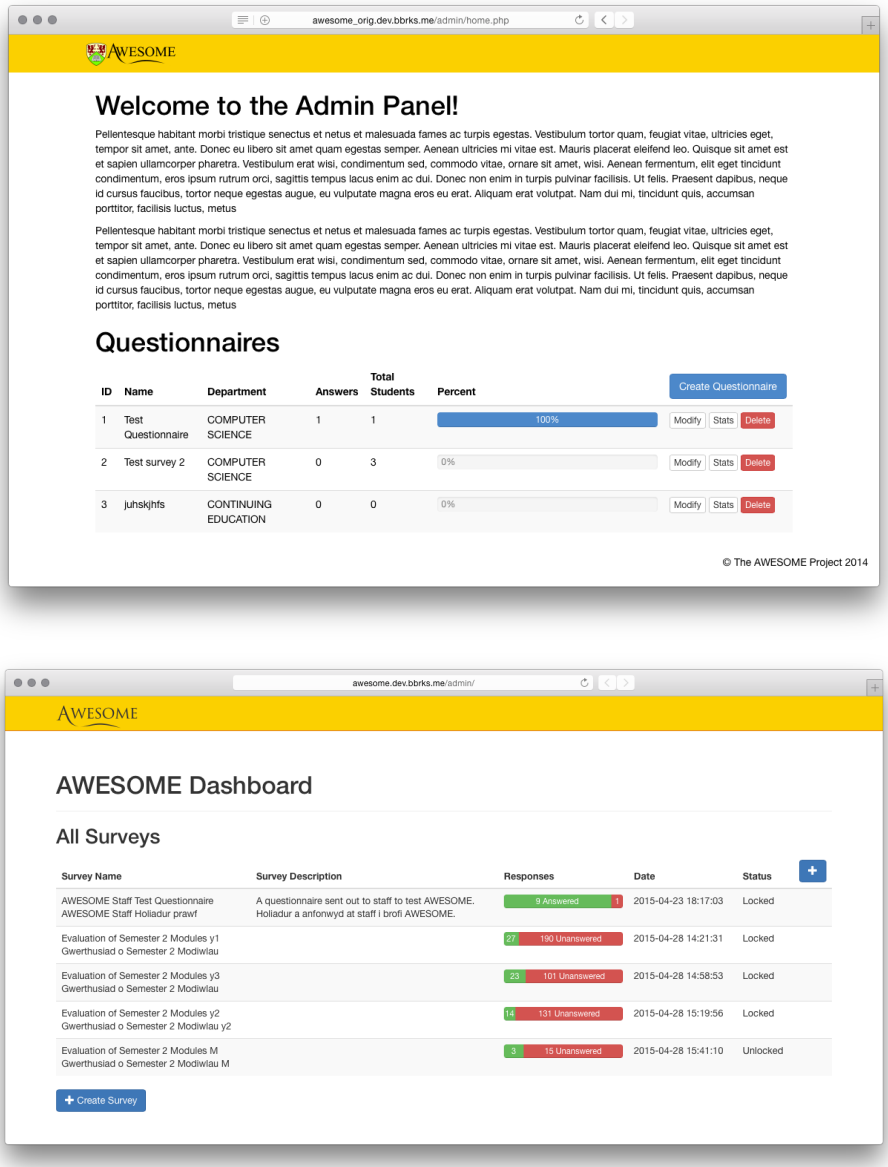


Figure 3.4.: A comparison of surveys view between the prototype version and the submitted version of AWESOME

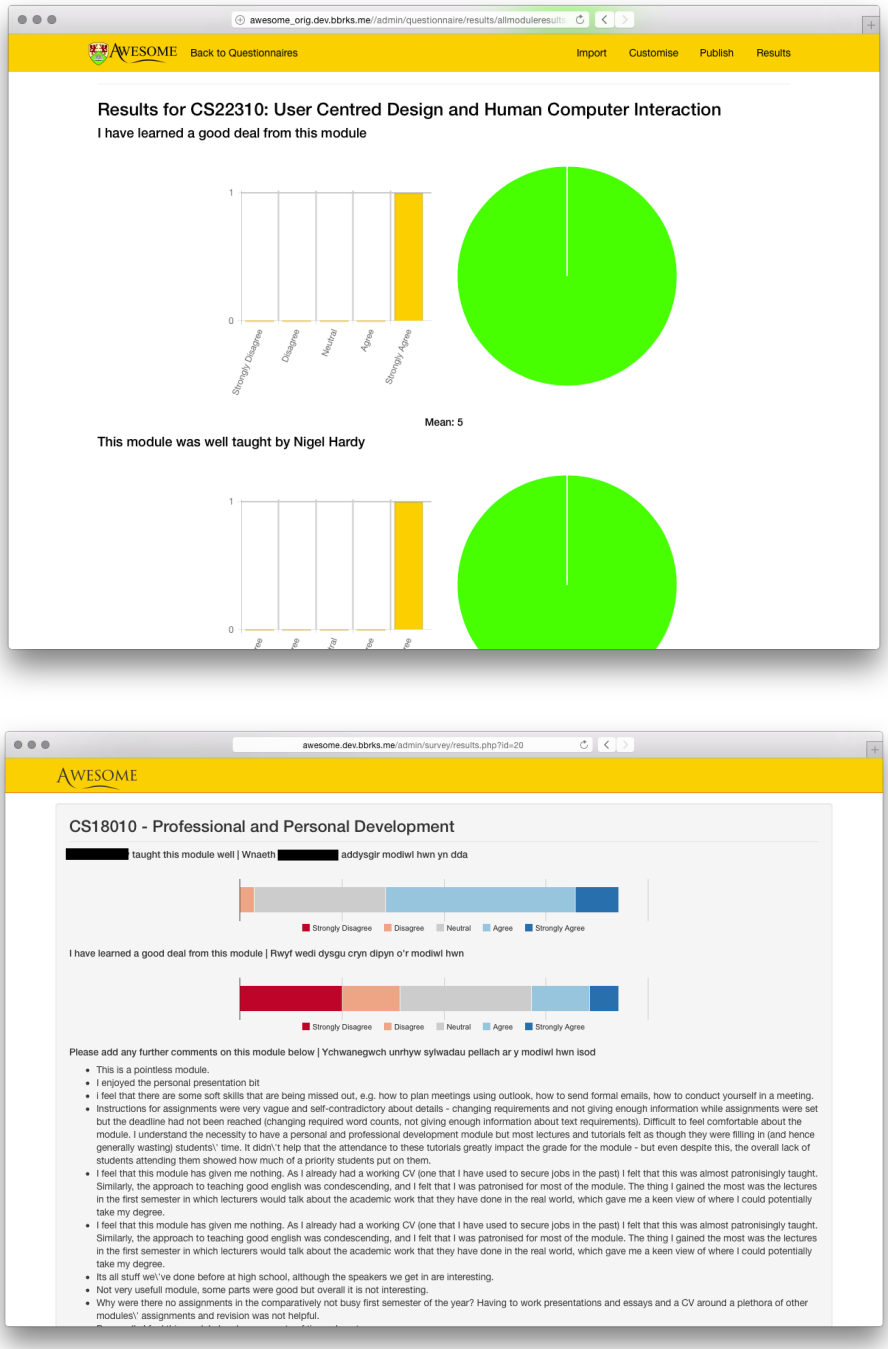
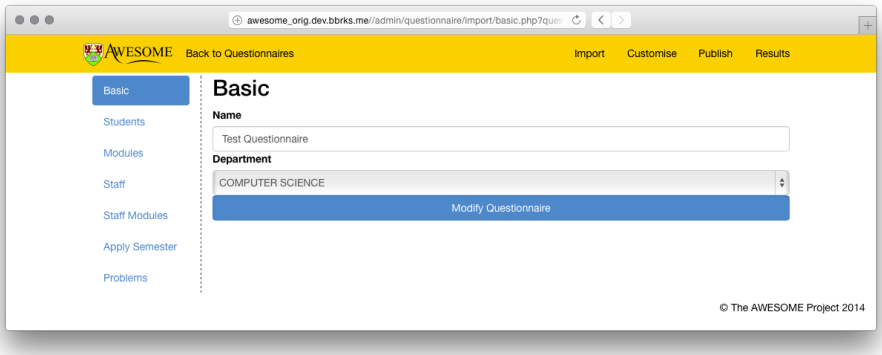


Figure 3.5.: A comparison of results between the prototype version and the submitted version of AWESOME



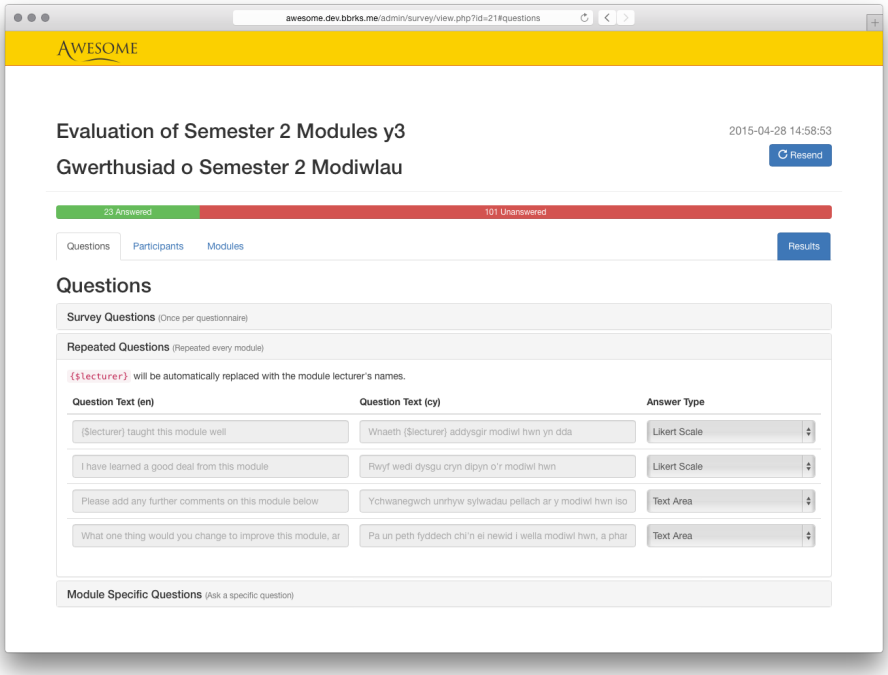


Figure 3.6.: A comparison of survey view between the prototype version and the submitted version of AWESOME

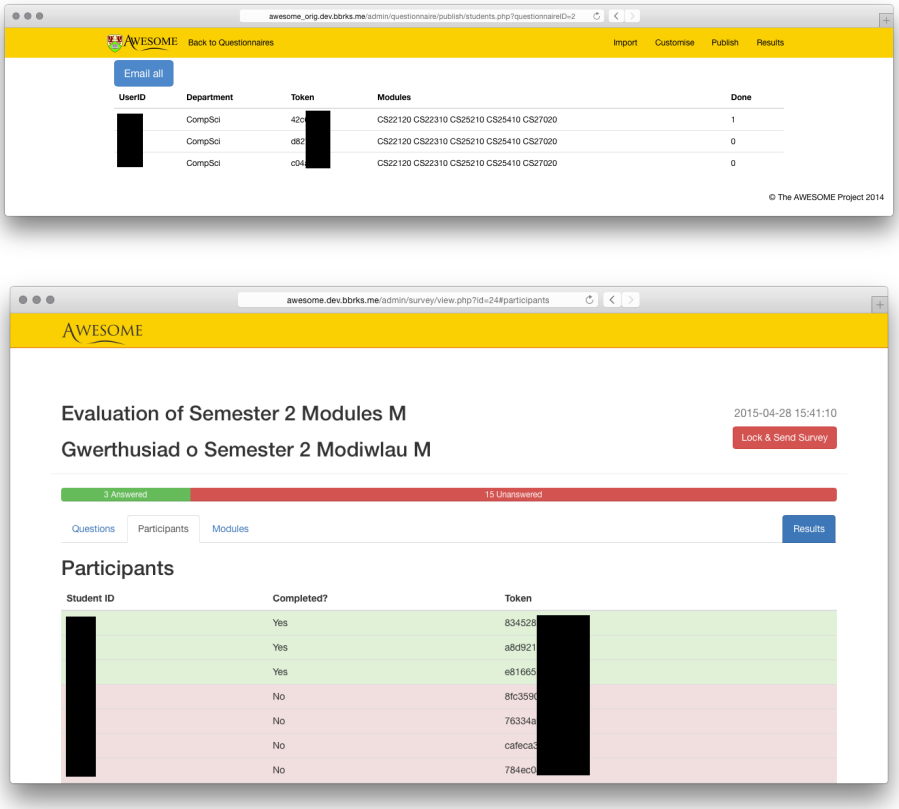


Figure 3.7.: A comparison of respondents between the prototype version and the submitted version of AWESOME

Additional features include a feedback form (Figure 3.8) when AWESOME is in debug mode. This allows users to send an email to developers, containing any text they wish, as well as automatically including user-agent and other metadata to help identify the problem.

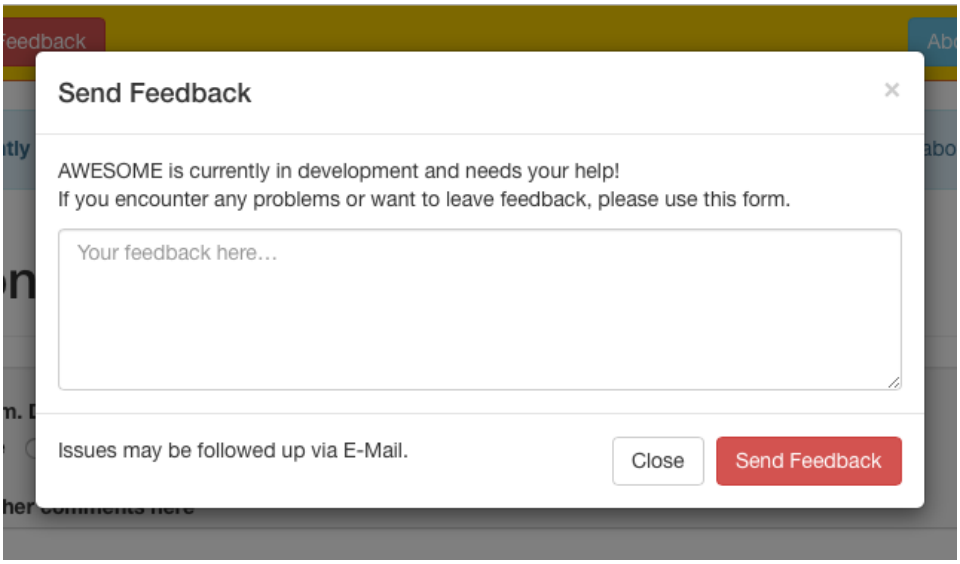


Figure 3.8.: A screenshot of the feedback form in AWESOME

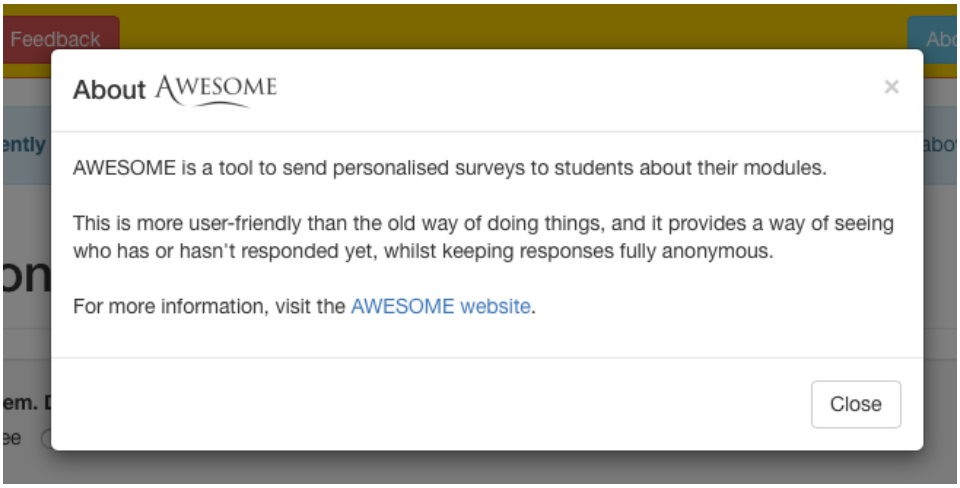


Figure 3.9.: A screenshot of the about dialog in AWESOME

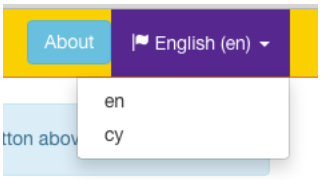


Figure 3.10.: A screenshot of the i18n selector in AWESOME

4. Implementation

4.1. Prerequisites

4.2. Security Audit

4.3. Model-view-controller Framework

4.4. Internationalisation Framework

4.5. Deploying to an AU server

5. Testing

5.1. Automated Testing

For this project, a Continous Integration (CI) platform was chosen to be used in order to facilitate automated unit tests. By using CI ensured that every time code was committed with unit tests, the suite was ran and results were instantly available. Any test failures resulted in a notification email being sent to identify when and where a problem had occurred.

After seeing fellow classmates use TravisCI[1] on previous projects, and reviewing the features it provides, I decided to utilise it in this project. Travis offered testing across multiple PHP versions, which proved helpful when trying to provision a server on the university network.

This was indispensable mid-way through the project, when functionality in the i18n framework needed to be changed and tests started to fail. I wouldn't have noticed any errors when manually testing, but the unit tests brought up edge cases which were then dealt with.

5.2. User Testing

User testing was carried out by creating a survey and entering volunteer's AU usernames in the student CSV data along with sample modules. The first real test through the AU servers was sent to staff. This uncovered many issues with the setup of AWESOME on the AU server, which took some time to fix. After these issues were resolved, an end-of-semester questionnaire was sent out to the vast majority of students in the Computer Science department. From first years, to masters students.

User testing revealed some useful information via the feedback form, as well as through a question in the survey which was asked about how easy AWESOME was to use.

[1] TravisCI Homepage: http://travis-ci.com

5.3. Acceptance Testing

Acceptance testing was carried out on the submitted version of AWESOME and results in test tables can be found in Appendix B with 16/18 (89%) of tests passing. More detail of the two failing tests can be found in the appendix entry.

6. Evaluation

6.1. Process

6.2. Development Environment

6.3. Project Stages

6.4. Blog

6.5. Degree

6.6. Upper Management

6.7. Time Management

6.8. Future Scope

Bibliography

[1] Anonymous, "Rubber ducking." Portland Pattern Repository, Oct. 2014. (http://c2.com/cgi/wiki?RubberDucking) Accessed: 2015-04-30.

[2] Anonymous, "Module evaluation questionnaires : Student evaluation : Teaching and learning development unit." University of Sussex. (http://www.sussex.ac.uk/tldu/ideas/eval/ceq) Accessed: 2015-05-05.

[3] Anonymous, "Student module evaluation." University of Westminster, London. (http://www.westminster.ac.uk/study/current-students/your-studies/student-surveys/student-module-evaluation) Accessed: 2015-05-05.

[4] S. R. Porter, M. E. Whitcomb, and W. H. Weitzer, "Multiple surveys of students and survey fatigue," *New Directions for Institutional Research*, vol. 2004, pp. 63–73, Jan. 2004.

[5] N. McEwan, "Use of quizdom as a means of assessing student comprehension of lecture material," 2009. (http://cadair.aber.ac.uk/dspace/bitstream/handle/2160/7419/2009%20-%20McEWAN,%20N.%20-%20TC2%20-%20Use%20of%20Quizdom%20as%20a%20Means%20of%20Assessing%20Student%20Comprehension%20of%20Lecture%20Material.pdf?sequence=1) Accessed: 2015-05-05.

[6] A. Garg, "Write your own php mvc framework," Mar. 2009. (http://anantgarg.com/2009/03/13/write-your-own-php-mvc-framework-part-1/) Accessed: 2015-02-15.

[7] "Phpmailer github repository." (https://github.com/PHPMailer/PHPMailer) Accessed: 2015-04-12.

A. Outline Project Specification

Aberystwyth Web Evaluation Surveys Of Module Experiences *(AWESOME)*

Report Name	Outline Project Specification
Author (User Id)	Benjamin Brooks (beb12)
Supervisor (User Id)	Hannah Dee (hmd1)
Module	CS39440
Degree Scheme	G401 (Computer Science)
Date	February 12, 2015
Revision	1.1
Status	Release

Outline Project Specification - 1.1 (Release) Benjamin Brooks (beb12)

1 Project description

The Aberystwyth Web Evaluation Surveys Of Module Experiences (AWESOME), is a prototype that enables departments to gather feedback by students about modules, lecturers, and other departmental issues. It is intended to replace and improve upon the current method of collecting feedback via Google Forms. This can be achieved by providing a personalised survey for each student to make questions personalised whilst also keeping results anonymous and confidential, and being able to chase up students for not completing their questionnaire.

AWESOME is a PHP web application developed by Keiron O'Shea during the summer of 2014 under the supervision of Dr. Hannah Dee. This project's goal is to bring the current prototype up to a well written, functioning, implementable, and extensible standard. Security of the system is critical, and so implementing a continuous integration system with unit tests and vulnerability scanning is vital to get up and running early on in the project. The system must be multilingual and accessible to adhere to the university's policies.

Advanced analytics and reports is also a feature which is highly requested by university management. For example, the system needs to be able to extract textual comments from the worst performing modules containing the word 'Feedback'. This can help upper management identify problematic areas in the university and look into the issue further.

The prototype has been demoed at a Learning and Teaching Enhancement Committee meeting to gather feedback about the current status of the project. The consensus from the committee is that the prototype is very impressive and would solve a need that is longed for by the university management. This confirmation by the committee further proves a need for this software to follow best software development practices to ensure that it can be used and extended in the future.

2 Proposed tasks

The prototype is currently written in procedural PHP, using the Twig [2] framework as a templating engine. In order to make the program more extensible and easier to maintain, it would make sense to refactor the current codebase to follow an object oriented (OOP) model-view-controller (MVC) [4] [5] architectural pattern.

The first task that needs to happen is a full security audit of the current version of the prototype. From there, the project can be refactored using best software practices, such as OOP, unit testing and MVC to separate the view and logic.

After the refactoring is done and the software is up to the same functional specification, additional functionality can then be introduced. The following task lists are what can be expected of each aspect of the project.

2.1 Refactoring tasks

Change procedural design – Design a new software architecture using MVC with OOP.

Unit Testing – Use Travis CI [1] to do automated unit testing and vulnerability scanning.

Secure admin dashboard – Use LDAP HTTP authentication via *.htaccess*.

PHP Data Objects (PDO) – Change the current *mysqli* and *tidy_sql* implementation to use PDO for greater security, flexibility and features when interacting with databases.

Accessibility (a11y) audit – Ensure all student-facing pages are accessible for disabled users.

2.2 Additional Functionality

Internationalisation (i18n) – Implement extensible i18n system to fully support Welsh, and additional languages.

Relational Database – Modify the database schema to be object oriented and relational.

Advanced Analytics – Create a system which can narrow down responses to criteria (e.g. Find modules with a low satisfaction score which mention 'feedback' in comments)

Traffic Light Dashboard – Have a dashboard showing traffic lights for all modules and departments to detect current issues.

3 Project deliverables

As this project has a fairly tight and fixed deadline on the temperature questionnaire testing, a lot of the refactoring work will be done as soon and as quickly as possible in order to get the project to a state that could be used.

After this initial sprint, the project can then be extended with the additional features described above. A rough outline of timeframes for deliverables is below.

Outline Project Specification – 2015-02-06 – This document.

OOP MVC Class Diagram – 2015-02-09 – UML Class diagram to describe MVC design.

OOP MVC Release – 2015-02-20 – Functional OOP MVC version of the prototype.

i18n and a11y – 2015-02-24 – Add internationalisation support and audit accessibility.

Temperature Test – Week 4-5 – Internal/closed functional testing.

Temperature Questionnaire – Week 6 – Questionnaire sent out to two departments.

Mid-Project Demonstration – 2015-03-09 – Start date of Mid-Project Demonstrations.

Analytics/Reports feature – 2015-04-17 – Have the analytics feature finished.

Final Report – 2015-05-07 – Final report hand-in.

Final Demonstrations – 2015-05-11 – Final project demonstrations.

Annotated Bibliography

[1] "Travis CI: Building a PHP project," http://docs.travis-ci.com/user/languages/php/, Feb. 2015, accessed Feb 2015.

Travis CI is a hosted continous integration system that connects to GitHub to help with automated unit testing and vulnerability scanning. I will be using this to provide automated unit testing and vulnerability scanning.

[2] "Twig - The flexible, fast, and secure template engine for PHP," http://twig.sensiolabs.org, Feb. 2015, accessed Feb 2015.

Twig is the templating engine used in the prototype by the previous author.

[3] K. T. Brinko, "The Practice of Giving Feedback to Improve Teaching: What Is Effective?" *The Journal of Higher Education*, vol. 64, no. 5, 1993. [Online]. Available: http://www.jstor.org/stable/2959994

An article describing effective practices in gathering feedback in an academic environment. This is general background reading to get a greater understanding of the project's aims and goals.

[4] C. Hopkins, "PHP Master — The MVC Pattern and PHP," http://www.sitepoint.com/the-mvc-pattern-and-php-1, Feb. 2015.

Another set of articles on OOP MVC in PHP which have been read in preparation for the refactor.

[5] J. Stump, "Understanding MVC in PHP," http://archive.oreilly.com/pub/a/php/archive/mvc-intro.html, Feb. 2015.

A series of articles on how to write MVC architecture in PHP which have been used as background reading ready for the OOP MVC design.

[6] H. K. Wachtel, "Student Evaluation of College Teaching Effectiveness: a brief review," *Assessment & Evaluation in Higher Education*, vol. 23, no. 2, pp. 191–212, Jan. 1998. [Online]. Available: http://dx.doi.org/10.1080/0260293980230207

An article describing effective practices in gathering feedback in an academic environment. This is general background reading to get a greater understanding of the project's aims and goals.

B. Test Tables

These tests were carried out as part of acceptance testing using the submitted version of AWESOME. Results are listed below.

- Test D8 fails as there is currently no safeguard in place for incorrect CSV format.
- Test D12 fails as questionnaire respondent tokens aren't created until the moment of sending.

Both of these issues a quick fixes, but are unable to be implemented in time for the dissertation hand-in. They will be fixed in a future version.

ID	Requirement	Input	Expected Output	Actual Output	Pass
D1	Admin Dashboard sits behind a login	Go to `awesome.url/admin`	Login form pops up	Login form pops up	✓
D2	Admin Dashboard login accepts correct username and password	Enter correct credentials into login form	Redirected to admin dashboard	Redirected to admin dashboard	✓
D3	Admin Dashboard login refuses incorrect username	Enter incorrect username and correct password	Login fails	Login Fails	✓
D4	Admin Dashboard login refuses incorrect password	Enter correct username and incorrect password	Login fails	Login Fails	✓
D5	Create Survey button starts the creation of survey wizard	Click 'Create Survey' button	Taken to screen asking for titles, description, and CSV data	Taken to screen asking for titles, description, and CSV data	✓
D6	Survey must have a title	Enter no title and press 'create' button	Message pops up disallowing action	Message pops up disallowing action	✓
D7	Survey must have CSV data	Enter no CSV data and press 'create' button	Message pops up disallowing action	Message pops up disallowing action	✓
D8	Must check CSV data for formatting issues	Enter incorrect CSV data and press 'create' button	Message pops up informing of CSV formatting error	Taken to survey page without any errors	✗
D9	Unlocked survey page allows entry of question text and type	Type a question title and select an answer type for a question	Able to enter text in question text and select an answer type	Able to enter text in question text and select an answer type	✓
D10	Able to add a new question	Click add question button	A new question text and answer type row appears	A new question text and answer type row appears	✓
D11	Able to delete an existing question	Click the delete question button next to a question	The question is removed and deleted on save	The question is removed and deleted on save	✓
D12	Be able view participants in a survey before sending	Click 'Participants' tab in a survey page	Respondents should be listed	Respondent list is empty	✗
D13	Be able to send a survey	Click the 'Send' button in a survey page	A message displaying how many people the survey was sent to	A message displaying how many people the survey was sent to	✓

Table B.1.: Acceptance testing table of AWESOME's Admin Dashboard

ID	Requirement	Input	Expected Output	Actual Output	Pass
Q1	Students receive an email with a link to personalised questionnaire	Check email inbox	See an email with unique link	See an email with unique link	✓
Q2	Questionnaire can be viewed correctly	Click link in email	See questionnaire displayed	See questionnaire displayed	✓
Q3	Answers can be selected or filled in	Click on a Likert rating or type in a text box	See answer selected or text typed in	See answer selected or text typed in	✓
Q4	Answers can be submitted	Select or type some answers and press the send button	Page redirected with a message informing of completion	Page redirected with a message informing of completion	✓
Q5	Questionnaire can't be completed twice	Re-visit the unique link and try to complete the survey again	Receive an error message informing that the questionnaire has already been completed	Receive an error message informing that the questionnaire has already been completed	✓

Table B.2.: Acceptance testing of AWESOME's Questionnaire