Classroom Quiz System

Final Report for CS39440 Major Project

Author: Alexander Taylor (amt22@aber.ac.uk)
Supervisor: Mr. Chris Loftus (cwl@aber.ac.uk)

13th April 2017
Version: 1.0 (Draft)

This report was submitted as partial fulfilment of a BSc degree in
TODO: Computer Science With A Year In Industry (G401)

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

Declaration of originality

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

By including my name 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

This project converns creating an application that can be used within lectures by lecturers to allow students to answer questions and for the lecturer to display live results of these questions. This project is based on a preexisting application, called Qwizdom (TODO: cite) but this project aims to improve upon Qwizdom in several ways.

The system is a web based application built in Laravel, a PHP based framework. The system provides two main functions, firstly the running of questions on their own, and the second is to stream lecture slides with questions embedded within the slides to students. This system improves upon Qwizdom in several ways, it runs via a website rather than via a PowerPoint extension, meaning more lecturers can use it, and it provides more functionality such as more possible answers and students cannot submit their own answers as they can with Qwizdom.

Development followed an Extreme Programming approach, utilising a number of practices to help development.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Background & Objectives

1.1 Background

1.1.1 Qwizdom

Currently lecturers at Aberystwyth University have the option to use a service called Qwizdom (TODO cite) which allows lecturers to embed quiz questions into their slideshow presentations. During a lecture, students can join a session through an online portal and have the slides and questions streamed through to their devices. Quiz questions can then be answered by the students and their results can be displayed live by the lecturer, who can also save these results for later analysis.

A replacement system was wanted to fix some of the problems that Qwizdom has. Primarily that the University only has one session key, which means only one session can be used by all lecturers at any one time, leading to some clashes. With a new system, there would be no limit to the number of sessions that can be run. Other problems with Qwizdom include:

- Maximum of 6 answers (TODO: was it six? ask Chris)
- Aging interface on the student end
- Students can submit their own answers by changing the HTML on the page
- Creating quizzes was tied into Microsoft Powerpoint which not all lecturers use

1.2 Analysis

1.2.1 Two sections

1.2.2 Approach

1.2.2.1 Frameworks and hosting

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 determined to be possible in the time available. A discussion of the process of arriving at the final list is usually appropriate.

As mentioned in the lectures, think about possible security issues for the project topic. Whilst these might not be relevant for all projects, do consider if there are relevant for your project. Where there are relevant security issues, discuss how they will this affect the work that you are doing. Carry forward this discussion into relevant areas for design, implementation and testing.

1.3 Process

The methodlolgy followed within this project was an Extreme Programming based approach. One of the core advantages of using an XP based approach is the ability to adapt to change. The second part of this project, implementing a method to stream slides to students, was much more open to change than the first part where the scope is far more understood. If another process such as Feature Driven Development was used, it would work for the first part of the project as all the requirements and features are known but most likely would be a struggle to use in the second part. XP gives the flexibility needed to complete the second part of the project whilst also allowing the first part to be done with ease.

1.3.1 Practices

A number of practices were selected for the project that work in a single developer project:

- Test Driven Development - Writing tests before coding any of the application logic helps to enhance both the design and also mean tests should actually be written rather than left until the end and ”hacked” in.
- Coding Standards - Keeping to strict coding standards helps ensure the code is good quality and easy to extend in the future as this project may well be.
- Small Releases - Small releases help enforce releases bit of working code regularly and results in a better overall project if the sub parts are all working.
- On Site Customer - This practice is somewhat applicable, the developer can act as a customer, and the project supervisor can also somewhat act as a customer, due to them being the originator of the idea and also a lecturer, one of the main users of this application.
- Merciless Refactoring - Refactoring is already encouraged by using TDD, but it can also be used at other points in the project to ensure the code is structured sensibly.
- Planning Game - This can be adapated to be done by one person at the start of the project, the list of stories was written and then iterations and releases planned out.

- Continuous Integration - Some online tools can be used to provide an CI workflow, to run tests continually with the constant small releases.

1.3.2 Stories

1.4 Plan

Chapter 2

Design

You should concentrate on the more important aspects of the design. It is essential that an overview is presented before going into detail. As well as describing the design adopted it must also explain what other designs were considered and why they were rejected.

The design should describe what you expected to do, and might also explain areas that you had to revise after some investigation.

Typically, for an object-oriented design, the discussion will focus on the choice of objects and classes and the allocation of methods to classes. The use made of reusable components should be described and their source referenced. Particularly important decisions concerning data structures usually affect the architecture of a system and so should be described here.

How much material you include on detailed design and implementation will depend very much on the nature of the project. It should not be padded out. Think about the significant aspects of your system. For example, describe the design of the user interface if it is a critical aspect of your system, or provide detail about methods and data structures that are not trivial. Do not spend time on long lists of trivial items and repetitive descriptions. If in doubt about what is appropriate, speak to your supervisor.

You should also identify any support tools that you used. You should discuss your choice of implementation tools - programming language, compilers, database management system, program development environment, etc.

Some example sub-sections may be as follows, but the specific sections are for you to define.

2.1 Overall Architecture

2.2 Database Design

2.3 System Interaction Diagram

This could possible be part of the overall architecture section

Chapter 3

Iterations

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 Iteration 0 16/02 - 22/02

3.2 Iteration 1 23/02 - 01/03

3.3 Iteration 2 02/03 - 08/03

3.4 Iteration 3 09/03 - 15/03

3.5 Iteration 4 16/03 - 22/03

3.6 Iteration 5 23/03 - 29/04

3.7 Iteration 6 30/03 - 05/04

3.8 Iteration 7 06/04 - 12/04

3.9 Iteration 8 13/04 - 19/04

3.10 Iteration 9 20/04 - 26/04

3.11 Iteration 10 27/04 - 03/05

Chapter 4

Testing

Detailed descriptions of every test case are definitely not what is required here. What is important is to show that you adopted a sensible strategy that was, in principle, capable of testing the system adequately even if you did not have the time to test the system fully.

Provide information in the body of your report and the appendix to explain the testing that has been performed. How does this testing address the requirements and design for the project?

How comprehensive is the testing within the constraints of the project? Are you testing the normal working behaviour? Are you testing the exceptional behaviour, e.g. error conditions? Are you testing security issues if they are relevant for your project?

Have you tested your system on “real users”? For example, if your system is supposed to solve a problem for a business, then it would be appropriate to present your approach to involve the users in the testing process and to record the results that you obtained. Depending on the level of detail, it is likely that you would put any detailed results in an appendix.

The following sections indicate some areas you might include. Other sections may be more appropriate to your project.

4.1 Overall Approach to Testing

4.2 Automated Testing

4.2.1 Application Testing

4.2.2 Stress Testing

4.2.3 Security Testing

4.2.4 User Testing

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?

Other questions can be addressed as appropriate for a project.

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.

In the latter stages of the module, we will discuss the evaluation. That will probably be around week 9, although that differs each year.

5.1 Methodology

5.2 Story Comparison

5.3 Tools and Technologies

5.4 Extra Tools

Diary and Trello

5.5 User Testing Evaluation

Appendices

The appendices are for additional content that is useful to support the discussion in the report. It is material that is not necessarily needed in the body of the report, but its inclusion in the appendices makes it easy to access.

For example, if you have developed a Design Specification document as part of a plan-driven approach for the project, then it would be appropriate to include that document as an appendix. In the body of your report you would highlight the most interesting aspects of the design, referring your reader to the full specification for further detail.

If you have taken an agile approach to developing the project, then you may be less likely to have developed a full requirements specification. Perhaps you use stories to keep track of the functionality and the 'future conversations'. It might not be relevant to include all of those in the body of your report. Instead, you might include those in an appendix.

There is a balance to be struck between what is relevant to include in the body of your report and whether additional supporting evidence is appropriate in the appendices. Speak to your supervisor or the module coordinator if you have questions about this.

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. If third party code or libraries are used, your work will build on that to produce notable new work. 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 client's 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 [1]. The library is released using the Apache License [2]. 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

For some projects, it might be relevant to include some code extracts in an appendix. You are not expected to put all of your code here - the correct place for all of your code is in the technical submission that is made in addition to the Final Report. However, if there are some notable aspects of the code that you discuss, including that in an appendix might be useful to make it easier for your readers to access.

As a general guide, if you are discussing short extracts of code then you are advised to include such code in the body of the report. If there is a longer extract that is relevant, then you might include it as shown in the following section.

Only include code in the appendix if that code is discussed and referred to in the body of the report.

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 [3].

```
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0 - EPS)
```

```
double ran2(long *idum)
{
  /*---------------------------------------------------*/
  /* Minimum Standard Random Number Generator          */
  /* Taken from Numerical recipies in C                */
  /* Based on Park and Miller with Bays Durham Shuffle */
  /* Coupled Schrage methods for extra periodicity     */
  /* Always call with negative number to initialise    */
  /*---------------------------------------------------*/

  int j;
  long k;
  static long idum2=123456789;
  static long iy=0;
  static long iv[NTAB];
  double temp;

  if (*idum <=0)
  {
    if (-(*idum) < 1)
    {
      *idum = 1;
    }else
    {
      *idum = -(*idum);
    }
    idum2=(*idum);
    for (j=NTAB+7;j>=0;j--)
    {
      k = (*idum)/IQ1;
      *idum = IA1 *(*idum-k*IQ1) - IR1*k;
      if (*idum < 0)
      {
        *idum += IM1;
      }
      if (j < NTAB)
      {
        iv[j] = *idum;
      }
    }
    iy = iv[0];
  }
  k = (*idum)/IQ1;
  *idum = IA1*(*idum-k*IQ1) - IR1*k;
  if (*idum < 0)
  {
    *idum += IM1;
  }
```

```
    k = (idum2)/IQ2;
    idum2 = IA2*(idum2-k*IQ2) - IR2*k;
    if (idum2 < 0)
    {
      idum2 += IM2;
    }
    j = iy/NDIV;
    iy=iv[j] - idum2;
    iv[j] = *idum;
    if (iy < 1)
    {
      iy += IMM1;
    }
    if ((temp=AM*iy) > RNMX)
    {
      return RNMX;
    }else
    {
      return temp;
    }
  }
```

Annotated Bibliography

[1] Apache Software Foundation, “Apache POI - the Java API for Microsoft Documents,” http://poi.apache.org, 2014.

This is my annotation. I should add in a description here.

[2] ——, “Apache License, Version 2.0,” http://www.apache.org/licenses/LICENSE-2.0, 2004.

This is my annotation. I should add in a description here.

[3] W. Press *et al.*, *Numerical recipes in C.* Cambridge University Press Cambridge, 1992, pp. 349–361.

This is my annotation. I can add in comments that are in **bold** and *italics and then other content.*

[4] M. Neal, J. Feyereisl, R. Rascunà, and X. Wang, “Don’t touch me, I’m fine: Robot autonomy using an artificial innate immune system,” in *Proceedings of the 5th International Conference on Artificial Immune Systems*. Springer, 2006, pp. 349–361.

This paper...

[5] H. M. Dee and D. C. Hogg, “Navigational strategies in behaviour modelling,” *Artificial Intelligence*, vol. 173(2), pp. 329–342, 2009.

This is my annotation. I should add in a description here.

[6] Various, “Fail blog,” http://www.failblog.org/, Aug. 2011, accessed August 2011.

This is my annotation. I should add in a description here.

[7] S. Duckworth, “A picture of a kitten at Hellifield Peel,” http://www.geograph.org.uk/photo/640959, 2007, copyright Sylvia Duckworth and licensed for reuse under a Creative Commons Attribution-Share Alike 2.0 Generic Licence. Accessed August 2011.

This is my annotation. I should add in a description here.