

An automated tool for teaching

Developing an e-learning tool

Desmond Patrick Joseph Traynor

Final Year Project
Computer Science & Software Engineering

Department of Computer Science
National University of Ireland, Maynooth
Co. Kildare
Ireland
February 26, 2003

A thesis submitted in partial fulfilment of the requirements for the Computer
Science and Software Engineering.

Supervisor: Thomas J. Naughton

Abstract

e-learning has been an area of massive growth commercially in recent years, however its growth within the academic world has been comparatively slow. Whilst there is still much research being performed within the area, maintaining the independent motivation of students still remains a problem. This project aims to tackle this problem using a combination of adaptive questioning and a staggered learning system. The desired results of developing such a system are to maintain student motivation and thus increase the ability to teach with computers. This project centers around the creation of a testing tool to be developed for use in an academic environment.

Declaration

I hereby certify that this material, which I now submit for assessment on the program of study leading to the award of B.Sc. Computer Science and Software Engineering, is entirely my own work and has not been taken from the work of others - save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed:_____ Date:_____

Acknowledgements

I would like to thank the following people, all of whom have helped me greatly in one way or another in the course of this project. In no particular order, Rory & Phil for keeping it light hearted, Vish & Dave for helping me greatly. Tom, my supervisor, for making sure I couldn't leave it to the last week. Most importantly my family for putting with me, and caroline, cause she's a **roide!!!**. **Thanks to everyone. Detroit!**

Contents

1	Introduction	3
1.1	Developing a tool for e-learning	3
1.2	Overview of the report	5
2	Requirements and Requirements Analysis	6
2.1	Initial requirements	6
2.1.1	System features	6
2.1.2	System attributes	8
2.2	Additional requirements	8
2.3	Analysis of requirements	9
2.3.1	Feasibility of requirements	9
2.3.2	Restrictions on development	10
2.3.3	Conclusions	10
3	Design	11
3.1	The login system	11
3.2	The question and answer system	13
3.3	Storage of data	14
3.3.1	The log file	14
3.3.2	The database structure	17
3.4	System operation	17
3.5	System justification	18
3.6	Alternative designs	18
3.7	Analysis of design	20
4	Implementation	22
4.1	The tools chosen	22
4.2	Using the tools	24
4.2.1	Installing Apache	24
4.2.2	Using MySQL	24
4.2.3	Using the Perl language	26

4.3	Implementing the user side	27
4.3.1	Implementing the administration system	29
5	Software Testing	32
6	Testing	33
6.1	System Overview	33
6.2	Test Plan	33
6.3	Test Cases	34
6.3.1	The login system	34
6.3.2	The question and answer system	35
6.3.3	Storing a user's results	37
6.3.4	The Administrator system	38
6.4	Conclusions	39
7	Conclusion	41
7.1	Analysis of project development	41
7.1.1	Requirements	41
7.1.2	System design	41
7.1.3	Implementation	42
7.2	Future work	42
7.2.1	Extensions to current project	42
7.2.2	Further work related to current project	43
7.3	Summary	43
A	The User Manual	47
B	New Material and Skills Acquired	54
C	Project Plan and Tasks assignment	57
D	Sample data for inputting questions.	60

Chapter 1

Introduction

This chapter aims to give a background to the project as well as stating the project goal. The first section details the difficulties with developing an e-learning and states the project goal. The second section gives an overview of this report, explaining the contents of each chapter.

1.1 Developing a tool for e-learning

e-learning has revolutionised the commercial world. The business world has acknowledged the power that e-learning offers, and has adjusted itself accordingly. e-learning claims to reduce training costs enormously, and claims it can speed training up hugely. As a result of this the U.S. e-learning market is worth approximately 10 billion in revenues [1]. Academia, however, has shown to be less fond of e-learning, partially because of skepticism and also because it requires changing both how students learn and also how we teach.

In this project I aim to develop a tool that will avoid the classic pitfalls of typical academic e-learning tools. In academia e-learning is often wrongly employed as a substitute to teaching rather than as an accomplice. As a result e-learning has frequently been dismissed as an option because it is believed there is no substitute to human teaching. Although it may be powerful in the commercial world its advantages are mostly financial and these have had less impact on the academic world which for the most part remains unconvinced. However, in recent years a number of e-learning packages for schools and universities have been developed see Ref [15],[16],[17],[18],[19]. The motivation for these developments is to enable the power of the internet to be used for educational purposes. However the lack of a clear market leader in the area shows that this goal proves very difficult.

It is important before beginning to design an e-learning tool, to first learn

from the market that currently exists. To avoid the most common pitfalls it is important that they are understood and alternative methods discovered.

The majority of e-learning tools consist of a number of lecture slides followed by a series of multiple choice questions. The student is presented with a grade at the end, and in some of the advanced tools told where his/her weaknesses lie. There is rarely a facility to ask questions of the system so the students who struggle can become disheartened very quickly and lose motivation. Also, the students who are progressing quickly may find the course too easy and they too can lose motivation. The main shortcoming of e-learning is maintaining students motivation [24]. In a classroom a student cannot walk away, but they can easily leave a computer. Furthermore, students in a classroom are not distracted by other content accessible over the internet, or the computer itself.

However there is still much research and development being performed within the area. As the technology improves the advantages of e-learning become more obvious. The tools are constantly available to the students, the results are returned immediately, and because it is web based, the students do not need to be gathered together in one central location. Students can progress at their own pace at a time that is best suited to them.

There are also several features that have yet to be implemented in current market contenders, however, and these are the points I hope to address with my tool.

- Good educational software should react to the students' problems effectively, if a clear difficulty is detected in one area it should both inform and aid the student.
- If a student is progressing well, it should have the capacity to make the questions more challenging. Some students lose interest if they are not challenged by a course.

The full development of the above two points would lead to *personalised testing*, where the good students are kept interested by challenging questions, and the weaker students are not disheartened by lack of understanding. These points would address the main problems with e-learning. If motivation can be kept high by both helping students overcome difficulties and creating more challenging questions for the better students then e-learning will overcome its primary problem which is how to maintain independent motivation.

I am attempting to design an e-learning tool that will meet the above criteria for a small subset of applications. The tool I develop will not attempt to rival the current market contenders, but to offer new insight into the development of e-learning. The tool will have limited functionality in

comparison, but will aim to address issues that have previously been ignored. Such a tool could hopefully be deployed in academic institutions with relatively more success than the current ones, considering the functionality offered. Although it will be limited in functionality, the tool is to incorporate a generalised question-selection mechanism that can easily be adapted to any corpus of multiple-choice questions.

1.2 Overview of the report

The project requirements are detailed in chapter 2, also discussed is the analysis of the requirements and the conclusions drawn.

The design chapter explains the design of the systems, and also outlines other designs. The chapter finishes with a justification of the design. The design chapter is followed by the implementation chapter which discusses how the design was followed in the development of the system. The implementation chapter also explains which tools were used in the creation of this project. The testing chapter details how the product was tested for its requirements, and explains how the system behaved under each case.

The final chapter shows how requirements were satisfied, and how the project was completed successfully. The chapter also includes possible extensions to the tool, as well as potential projects inspired by developing a tool for e-learning. In the appendices are, the user manual created for the tool (A), a list of the materials and skills gained during development of this project (B), the project plan (C), and finally some sample data for question input (D).

Chapter 2

Requirements and Requirements Analysis

In this chapter I hope to detail all of the requirements both initially provided by my supervisor and those specified at later dates. The first section will outline the requirements as stated in the project proposal. The second section will detail all additional requirements. The third section will analyse the requirements and detail what constraints they enforce on the tool.

The goal of this project is to produce a web based tool that can examine students knowledge using multiple choice questions. The tool must consist primarily of a subject independent teaching architecture and a separate database that stores the questions.

2.1 Initial requirements

These are the initial requirements specified for the tool. This explains the required software features that the tool must have. These were agreed with my supervisor on 30th October 2002.

2.1.1 System features

The system features are divided into two groups, requirements 1-7 detail the minimum functionality this should have. There are numerous supplemental features (8-14) that are desirable; these should only be included once the initial requirements are fulfilled.

1. The tool must generate multiple choice questions for students to answer.

2. The questions must have a level of difficulty associated with them, and each answer a student gives should be recorded.
3. Each successive question should be decided based on the level of difficulty and outcome from the previous one.
4. Each session/test should start from a level based on the students overall performance.
5. The tool should be web based.
6. The tool itself and the questions should remain entirely separate. The lecturer should be able to change the question database, with no impact on the tool itself.
7. There must be an interface for the lecturer, from which the questions can be altered, and students' progress monitored.
8. The tool should offer links to online tutorials that are relevant to the current or previous questions.
9. Every effort should be made to ensure that all students take a randomly chosen path through the questions.
10. There should be some measurement of timing for each student. This could be used to restrict the allowed time per test, or merely to inform the student how long his/her exam took.
11. The tool should be able to modify the difficulty of each question to some extent, for example it could limit the time allowed per question. This would make a question more challenging for the student.
12. There should be the possibility for students of not answering a question, this will make a negative marking system possible, if desired.
13. The addition of these features should not compromise the reusability of the system. If another lecturer wished to use the system, he/she should have the option of disabling these features.
14. The tool should be installed on a Departmental server so that it can be employed in a learning environment and its performance measured.

2.1.2 System attributes

The system attributes are the requirements that the system must meet for successful deployment in an academic environment.

- **Reliability** The tool must not exhibit any errors which may result in the loss or corruption of test information.
- **Security** The database will hold sensitive information, therefore security is a high priority for the tool.
- **Maintainability** The administration interface must provide all the functionality required for the maintenance of the tool.
- **Portability** The tool must work on all computers in the environment where it is to be used.
- **Performance** The tool should work at a tolerable speed for users, given current technology, therefore the potential lag following a user action should be minimal.

2.2 Additional requirements

The following requirements were added during initial research and design, when it became more obvious how the features would be implemented. As these were added, it became necessary to alter both the specifications and the possible system designs.

1. The range of possible levels of difficulty associated with questions must be unbounded in terms of increasing difficulty, and between any 2 given levels it must be possible to introduce a new level.
2. The number of possible answers associated with each question must not be bounded. Also, any given question may have more than one correct answer.
3. There must be a history file that records every answer ever given on the site, plus the user and time associated with it. This is so that the tool can justify all the marks it awards. This feature is required given the spirit of freedom of information that the university has embraced.
4. A specific security issue is that it must not be possible to gain entry to another users account.

5. Students must not be able to re-answer questions using the *back* button, or select another question using the *refresh* button, on their browser.
6. The system must have some browser independent method of handling symbols (e.g. ϵ , δ etc) that are frequently used in teaching science and technology courses.

2.3 Analysis of requirements

This section discusses the feasibility of incorporating all of the above requirements. It will also discuss the restrictions imposed on the implementation of the tool, due to requirements.

2.3.1 Feasibility of requirements

- It is not possible to give each student a randomly chosen path through the questions (as requirement 9 requests) if each question is decided by the outcome of the previous question. It would be more correct to say that each student will be given a randomly chosen question from a pre-determined set of questions.
- The requirements request that the number of answers per question be unbounded (additional requirement 2). This is not possible because the amount of data stored will always be bounded by the capacity of the machine holding it. Therefore, the requirement will be interpreted to mean bounded only by the underlying data type, operating system, and hardware.
- The requirements request an unbounded level of difficulty (additional requirement 1), which again is not possible, as bounds will be put on this number once it is assigned to a specific data type. Therefore, the requirement will be interpreted to mean bounded only by the underlying data type chosen (assuming a reasonable choice of data type).
- It is not possible to establish a fully secure system that will prevent accounts being attacked and accessed illegally (as requested in additional requirement 4). Every effort will be made to ensure that it would be computationally unfeasible to attempt such an attack. However it is not possible to also authenticate the users, i.e. all that can be checked is that the user has a matching name and password, we cannot ensure they **are** the person whose account they are using. Consequently, any

such high security system will have a trade off against the performance of the tool in terms of speed and overall performance.

2.3.2 Restrictions on development

- The selection of languages that can be used to implement this tool is heavily restricted by the requirement that the tool must be web based. This leaves the option of running it through a Java applet , or writing it in a scripting language.
- The requirement for an unbounded number (in principle) of questions and answers means that a proper database structure is required with tables representing questions and answers, as it would be unfeasible to attempt this with a text-file storage system.

2.3.3 Conclusions

The requirements analysis has concluded what technologies can be used in the implementation of this project. A database driver that can handle large data input and output quickly is required. Also required is a programming language that can

1. be executed from a webpage and provide all the functionality necessary for dynamic web page creation, and
2. be able to communicate with a database to extract and insert arbitrary amounts of data.

Also required for the development of this project is some sample questions for both developing and testing the system. I will be using past exam papers for creating sample questions, as this will give realistic test data.

With the above requirements and restrictions in mind, the next phase is to design a system or systems capable of meeting all of the above requirements.

Chapter 3

Design

The requirements for this tool heavily restricted the possibilities for system design as shown in section 2.3.2. Having completed requirements analysis, I designed a system that I believed capable of meeting all the requirements. The system is based around one centralised server (see Fig. 3.1) that holds all the information for the tool. Users log in to the server through a secure channel and communicate with the system via a web-based interface.

Figure 3.1: A centralised server

3.1 The login system

It was a requirement (additional requirement 4) for the system to have security so that one user can not access another user's account. This required a high security login system so that user passwords would not be accessible. I chose to use a popular login system which uses a one way hashing function and randomly generated numbers. This is a standard method for secure logins and is used by many email and e-commerce sites. Further information on secure login systems can be found in [20]. This method is illustrated in Fig. 3.2

The security on this system involves the use of a cryptographic hash function, as only the hash value of a student's password is held in the database.

Figure 3.2: The login system

When a student's password is created, a *hash* of the password is stored in the database. A hash function H is a transformation that takes an input string and returns a fixed-size string, which is called the hash value. A hash function should also have the properties that to calculate the hash of an input string should be computationally inexpensive, but given a hash value, it should be computationally infeasible to find the input string which generated it.

1. Login script

When a student is prompted to login, a random number is generated for that particular student, and it along with the users *IP* number is entered into the login database. An IP number is a unique address for the device that the student is connecting to the internet with. The login database holds IP numbers and randomly generated numbers. The randomly generated number a student was given can be found, by looking up the IP number in the database.

2. Hashing algorithm

The hashing algorithm takes a student id (a unique identifier for that student) as well as a password and a randomly generated number for input. When a student enters their student id and password, the password is immediately hashed before being passed to any other script. This is achieved using a client side hashing algorithm. The randomly generated number is then appended to the hash value and this new string is then hashed again.

3. Verification Script

The verification script firstly retrieves the randomly generated number from the login database (using the IP of the student as a primary key). Secondly it retrieves the hash-value for the student's password, held in the student database. It then appends the randomly generated number to the hash value and hashes this newly created string is then hashed. If this hash-value is equal to the output from the hashing algorithm, then the student has entered the correct password as is verified.

The hash functions are used to ensure that the password is never transmitted as plaintext and therefore any breach of security would still require the intruder to find the string that generated this hash value. The randomly generated numbers are used, to ensure that all hash values are valid for one login only, therefore fully protecting the passwords.

Figure 3.3: Asking a question

Figure 3.4: Receiving an answer

3.2 The question and answer system

The requirements specified that each question a student is asked must meet some criteria. Although the questions are selected randomly, they are from a pre-determined set of questions. In order for the tool to select an appropriate question it must impose some restrictions on the question itself. The method for asking questions is illustrated in Fig. 3.3.

The script to select questions must find a random question that the user has not been asked already and is appropriate for his/her level. All students have a level associated with them, this level corresponds to their progress with the tool thus far. To select an appropriate question the script must first consult the users log file, and then the database. Firstly the script consults the log file and obtains variables defining which questions the user has been asked, and what is current level. Using this information the script constructs a SQL statement to extract a random question that they have not already asked, and is appropriate for the current user level. For a detailed description of the data stored in the logged and the database see section 4.3 .

When a user answers a given question, the answer script checks if the question that the answer corresponds to is a valid one. It does this by viewing the logged, and ensuring it is not in the list of questions already asked. It then retrieves the feasibility of the answer from the database and informs the user if the answer is correct. The method for answering question is illustrated in Fig. 3.4. For a detailed description of the database tables see section 4.3.

After a fixed number of question and answers, the server will terminate communication with the user. A flowchart describing a full session with the user can be seen in Fig. 3.5.

Figure 3.5: The flowchart for the tool. The number of questions n is adjustable by the administrator

3.3 Storage of data

All data required for the tool is held on the server in 2 separate methods. A log file is maintained for each user for the duration of their exam. The structure of the logged is detailed in section 4.3.1. All student information, as well as questions, answers and marks are held in the database. The database structure can be found in 4.3.2.

3.3.1 The log file

This is a file that is maintained on the server and stores all required information for user during their session with the tool. The log file is a || separated file storing the following information.

1. **time**
The time that the current test began.
2. **questions**
The questions that the user has already answered.
3. **number correct**
This is the number of questions the user has answered correctly.
4. **current level**
This is the current level of progress the user is at.
5. **last question**
This is the most recent question the user has been asked, this must be maintained to recognise when the refresh button has been pressed.
6. **number correct at current level**
This records the number of questions the user has answered correctly at their current level.
7. **questions remaining** This is a boolean value that records if the system has new questions remaining for the user. If there are no remaining questions on or below the current users level, it is permissible to re ask an old question. This variable is maintained to recognise the difference between running out of questions and cheating.

3.3.2 The database structure

The database structure was to ensure that all possible requirements could be satisfied. Five tables were created, and are explained below.

1. **questions** This table holds all the information about each question that may be asked. A question entry has associated with it a question id, type, question text, link to a hint, link to an answer and a level of difficulty.
2. **answers** This table holds all the information about each possible answer. Each answer field holds an answer id, answer text, the corresponding question id, the feasibility of the answer (1.0 if correct), and its position (e.g. none of the above must be last answer).
3. **marks** This table stores all the final marks in each exam taken. The marks table holds a student's email, the date of his/her exam, the number of questions examined, and his/her final mark.
4. **students** This table holds all details about each student that is using the tool. The table has columns holding firstname, surname, email, year, course, cao number and hash value of the password.
5. **sessions** This table is used for the login system, to keep track of the random numbers that are used. It holds the IP of the computer that is logging in, the random number being used during the login, and the time they first visited the login page. (All sessions older than three minutes are deleted, for security.)

3.4 System operation

The following is a brief overview of the desired behaviour of the system when implemented correctly. The system is designed to operate as follows:

1. The lecturer will have a system with a full database of questions/answers, and a full database of students.
2. The lecturer decides what level the students will begin at, what score must be achieved to advance a level, and what score must be achieved to regress a level. The lecturer also decides a time limit for a test.
3. The lecturer turns the system active using the config.cgi script.
4. From this point students may log in to the system.

5. A student's test is over when they either a) run out of time, or b) answer the predecided number of questions
6. All students are logged out when their test is finished. Students are then prevented from logging in until the next time the system is activated.

3.5 System justification

This system design is capable of meeting all the initial requirements possible at design level, this section explains how each of the requirements can now be satisfied using the chosen system.

- The requirements requested a web-based system that asks multiple choice questions. The designed system is web-based, and holds a finite number of answers for each question.
- The requirements requested a level of difficulty associated with each question, and that each successive question be based on the output of the previous one. The database stores a level of difficulty for each question, and the log file holds the current level, it is possible to select an appropriate question for the user (based on their current level), and also to increment their current level when necessary.
- The requirements requested that each answer a student submits is to be recorded, each answer is both recorded in the database and on the logged. The marks database also maintains information on the students overall performance.
- Finally the initial requirements state that the lecturer must be able to alter the questions and view student progress. It is possible to write web-based scripts that are capable of viewing and altering fields in the database.

From the above it is clear that the chosen design is clearly capable of satisfying the initial requirements for the tool.

3.6 Alternative designs

In this section I will discuss alternative designs and explain how they were less appropriate for this tool.

The choice to use a centralised server was made having looked at some alternatives and analysing one in particular. An alternative design I thought

Figure 3.6: A client loaded model

of was to make the client machines perform most of the computation, and merely send results to and from a central database. Each client would run a program on his/her machine that communicates with the server to extract questions, and inform the database of their outcomes. This client-loaded model is illustrated in Fig. 3.6.

There were some problems with designing a system of this type.

- It could cause problems in security of the tool, as the program will hold the correct answers and they could therefore be attacked more easily than if the correct answers were stored in a centralised secure location.
- It would not result in a significant speed up in performance, as each client side program must still communicate with the database at least once per question. (It is not possible to change this, as the requirements say each question must be given based on the outcome of the previous one and it is not feasible to store the complete corpus of questions on each client.)
- It requires writing a database server and a client side program whereas the centralised server requires only the server side programs, and the utilisation of a third party browser at each client.

The login system is standard web security design. An alternative to this would be to purchase an SSL certificate (Secure Socket Layer) and have the entire website under high security. This was not an option I pursued as I did not wish for the tool to be tied to any third party products, especially when feasible alternatives existed.

The question and answer model was designed to ensure that the question and the answer are never transmitted at the same time. This is important for maintaining secrecy of the questions. The user receives the question in a webpage and chooses an answer. The server then receives the *question id* and the *answer id*. These are 2 unique identifying numbers that only have meaning in the database. The server will then decide if the answer is correct by consulting the database.

3.7 Analysis of design

During implementation of this design, several further design issues were observed. These were mainly requirements issues that had yet to be defined, or small problems observed with the design.

- The requirements do not specify how the system should behave, if the tool were to have no new questions of appropriate level remaining for a student. The chosen behaviour was to re-ask a question, as the student should not be punished as a result of the system having no remaining questions.
- The requirements do specify what subset of the questions the tool should pick from. The decision made by my supervisor was to pick twenty questions at or below the students current level and randomly choose one.
- The *Refresh* button featured in most internet browsers was causing unexpected errors. If a user presses refresh the server would re-select the user a random question. This meant users could keep clicking refresh until they had a question that suited them. It is impossible to detect when a user has clicked refresh on the server side, so the solution I chose was to keep a variable in the clients log file indicating the current question under examination, and set to zero if no question was being asked. The server must check if a question is currently under examination, before assigning a new one. If a question is currently being asked, it reprints the current question again.
- In my design of the question and answer model I made it impossible for the user to answer the previous question again ,by using the back button. I achieved this by keeping a record of the questions as they are asked, and only accepting answers that correspond to the current question. I failed to see the possibility of multiple clicks of the back button, to answer a question that was asked more than 1 question previous (i.e. re-answering question 1, whilst on question 4). I had to solve this by also keeping a record of all questions already answered. This record was already being maintained, as it was necessary for selecting a question, it simply required some minor changes in error checking to ensure that this was no longer possible.
-

In conclusion the design chosen is capable of meeting all requirements stated. The tool will be web-based, and a database of students, questions and answers will be maintained. Each software-based requirement can be met through development, and the design chosen is suitable for their implementation. The next stage is to choose the tools required and begin development.

Chapter 4

Implementation

Implementing the chosen design required a decision about which tools and languages I will use for the creation of the system. There were four main issues to be resolved:

1. **Server** I had to decide which type of webserver I would use for hosting the system I would create. The ideal webserver would be fast, reliable, and capable of dealing with scripting languages.
2. **Database** I had to decide which database server I should use for holding the information relating to the questions, answers, marks, logins, and students. The ideal database server would be reliable, secure, and fast.
3. **Scripting Language** I had to choose a scripting language to implement the system with. The ideal scripting language should be compatible with the server chosen, and capable of interacting with the database server chosen.
4. **Freely Available Technology** All of the technology used should be freely accessible and modifiable so that the application can be maintained easily and its lifespan maximised throughout operating system upgrades, and so on.

4.1 The tools chosen

1. **The Apache Webserver**

The freely available Apache web server [23, 6], is arguably the most popular Web server in use on the Internet today. It has been ported to

most operation systems, and is highly stable. As a result of its popularity the bugs in each release are found very quickly and immediately resolved. I chose to use the Apache Webserver to host the website for the tool.

2. The MySQL database server

MySQL [21, 4] is a relational database software package that has become highly popular in recent years. Its popularity is due to many of the following features.

- (a) MySQL is highly supported by many modern scripting languages including Perl, PHP and ASP. These languages have easy to use packages built in for direct interaction with a MySQL database. This makes it a very popular choice for websites.
- (b) MySQL is very portable, it is available for most modern operating systems including all versions of Windows and GNU/Linux.
- (c) MySQL is a very fast database package. It claims to outperform the majority of the current market contenders on most operating systems [22].
- (d) MySQL is open source software meaning that the source code is freely available and not proprietary. An advantage to this is that developers from any institution, organisation, or company can help with its maintenance.

For the above reasons I chose to use MySQL as my database server.

3. The Perl scripting language

Perl [2, 3, 4] is a scripting language that was originally developed for text manipulation. P.E.R.L stands for Practical Extraction and Report Language. However its ability to manipulate text so easily has made it a very popular choice throughout the years, and as a result many Perl packages providing extra functionality have been developed.

Perl is now used primarily as a scripting language for web sites as it has a very useful CGI package that enables it to dynamically create web pages, and pass variables between them. Perl has a huge collection of modules, all of which are open source and freely available (as is Perl itself). One example of such a module is the MySQL database driver. Using this Perl can communicate with a MySQL database with great ease.

Although other modern scripting languages such as PHP or ASP are becoming very popular, Perl is still one of the most widely used languages for websites. It is for the above reasons, and my own previous experience with it, that I chose Perl as the language to implement my tool.

For a complete listing of all tools used, plus their version numbers and the location of their sources, see Appendix A: The User Manual.

4.2 Using the tools

This section will give an outline of the steps required to install each of the tools, and to enable them to communicate with one another.

4.2.1 Installing Apache

Having downloaded the appropriate files from the website and decompressing them, the Apache webserver installs itself to a directory of your choice. This directory will contain many folders, three of which are important for development purposes.

1. **Apache\htdocs** This folder contains all standard html documents that you want visible to the outside world. The file `index.html` is the first page a user will see when they visit the website.
2. **Apache\cgi-bin** This folder contains all scripted programs that you want executed by the server when a user selects them. All Perl scripts must be placed in this folder for execution.
3. **Apache\logs** This folder contains the log files of all visits to the website (including the time and IP number). It also contains the logs of all errors caused by scripts (including the time, the file, and the line number).

Having installed apache, I wrote a basic index page explaining the details of my project, and from this page I began to reference all future work. The original index page can be seen in Fig. 4.1.

Figure 4.1: The original index page

4.2.2 Using MySQL

I downloaded the most recent MySQL version from the website, and installed it. As MySQL runs a database, it must be installed as a service on the host machine, so that the databases are permanently accessible. Once installed, I set about designing the database, I created five tables for the operation of the tool as specified in section 4.3.2.

When creating the five tables, appropriate types for each field had to be decided.

1. **questions** The level and question_id of a question are held in numerical types. The remaining fields are set to a data type suitable for text storage.
2. **answers** In the answers table the fields vary in type. The numerical fields are answer_id, question_id, and feasibility. The answer itself is held in a text field and the position is an enumeration of strings (for example 'top', 'bottom').
3. **marks** This table stores the student_id as a text entry, the questions asked and score are numerical and a date field is used to record the date. The date field maps identically to a corresponding function in Perl.
4. **students** All details about students are held in text fields, as although in some case a number may seem intuitive (as in student number) unless the numeric qualities are being used, it is best to use a text field as they are less bounded and more versatile.
5. **sessions** The sessions table holds the IP of the user as a text entry. The time and randomly generated number are both held in numerical fields.

More explanation of these tables is available in the user manual.

The first installation of MySQL was command line based, although there is a free graphical user interface available from the website. Having learned all the SQL (Standard Query Language) that I believed would be required, I then installed the MySQL Control Center which made database manipulation far simpler.

4.2.3 Using the Perl language

Although I had previous experience with Perl, it was necessary for me to revise how CGI scripting works, and how best to use it for my project.

Figure 4.2: The first successful output

The Common Gateway Interface (CGI) is a standard for interfacing external applications with information servers, such as a Web server. A plain HTML document that a web browser retrieves is considered static, which means it exists in a constant state: a text file that does not change. A CGI program, on the other hand, is executed in real-time, so that it can output dynamic information: a dynamically generated HTML document.

A simple example of a CGI script would be a webpage, where the user inputs their name, and are then brought to a page containing their name. This webpage is not held anywhere on the server, it is created for the user, and is no longer viewable once a user leaves.

For the creation of the website I have incorporated many Perl modules. The most important ones for development were:

- *CGI.pm* This is a module created specifically for CGI programs. It contains many useful functions for dynamic webpage creation.
- *DBI.pm* This is a Database Interface module. It contains the core functionality for creating database queries and sending them to database servers.
- *DBI-MySQL.pm* This is a separate extension to the DBI module. It contains all the specific functions for connecting to a MySQL server.

4.3 Implementing the user side

When I had fully familiarised myself with Perl and its capabilities, I wrote a script to select a question from the database, and display it on the screen. The output from this script is visible in Fig. 4.2.

With this initial implementation, I began to follow the incremental development model as outlined in my project plan (see Appendix B). I added extra functionality incrementally, moving from selecting a randomly generated question to displaying the answers in a randomly generated order. At each point I was satisfying a particular requirement, however as the remaining requirements grew fewer, the difficulty of satisfying them grew higher. This is partially due to the increasing difficulty of the additional requirements and partially due to the increasing complexity of certain scripts. It

Figure 4.3: The login screen

Figure 4.4: A sample question screen

was important that each additional feature must not impact on the performance of the tool, and must be optional. Therefore when a new feature is introduced it must be possible to disable it.

When I believed I had satisfied all of the requirements with the tool I began to redesign the simple user interface, including features such as a clock (for timed tests) and more information regarding the state of the exam. Some screen shots from a standard path through the system from login (Fig. 4.3) to question (Fig. 4.4) through to the end of exam (Fig. 4.5) When I was happy with the look of the user side of the tool, and satisfied that it met all the requirements, I began to develop the administration side of the tool as this was one of the few remaining requirements.

Figure 4.5: The end of a sample exam

4.3.1 Implementing the administration system

The requirements stated that the administration system must be a system independent module for managing the question database, and also must have an interface for monitoring the progress of the students. A key issue in designing the lecturer/administrator interface was the security of any such system. The data that the administrator could access is highly sensitive and must therefore be kept highly secure. I used a combination of passwords and IP checking (data can only be accessed from the machine hosting the database) to keep the information relatively secure.

The requirements for the administration interface were short and mainly required retrieving information from the database, and displaying it in a formatted manner. The requirements were satisfied using the following scripts:

- *viewuser.cgi* This script takes a username as input, and displays all information held about the user as its output.
- *viewdate.cgi* This script takes a date as input, and displays all test results from that date as its output.
- *viewall.cgi* This script takes a class and a number of tests as input, and displays the students in a class and their average score over the chosen number of tests.
- *adduser.cgi* This script takes a full set of user information as input, and stores the user in the students database.
- *editconfig.cgi* This script is for editing variables such as the number of questions per exam etc.
- *reload.cgi* This script takes a question file (in HTML format) as input, extracts the questions and stores them in the database.

The first 3 scripts all have a very similar structure, they take in some criteria and query the database using it. The fourth script *editconfig.cgi* takes in a full set of parameters which are the global variables of the tool and sets them accordingly.

The option for the administrator to update/reset the question bank is offered by the last script. The script *reload.cgi* takes a html file as input (see

Figure 4.6: What the script sees

Figure 4.7: What the lecturer sees

Fig. 4.7). The lecturer maintains one or more question files on the server and can load the questions from these files, using the web-based script. A html file was chosen as it enables the lecturer to view the questions before submitting them (see Fig. 4.6). The script ignores the html preamble and extracts the questions, answers and other available information. This information is then sent to the questions and answers database, and can then be accessed by the other scripts. A further example of sample question input is available in Appendix C.

When the implementation of the lecturer side was finished, I agreed with my supervisor that all software requirements had been satisfied, this was agreed at our regular weekly meeting of 15th January. The next stage in development was to begin to test the product.

Chapter 5

Software Testing

Chapter 6

Testing

This chapter explains the testing that has been performed on the system, after its implementation. Each script was tested during the development phase. To achieve this I wrote many test cases, based on simple input and output before writing a script. The test cases for each script covered all the functionality that I wished to the script to have. A script was only considered to be finished when it had passed all of the test cases. This method of testing during development, in isolation of other layers is known as *Unit Testing* [13, 14]. Unit testing, however, is not sufficient to ensure a system performs satisfactorily. The application and integration (the communication between scripts) of the units still must be tested. Proper paths through system components such as the login system or the question and answer system had to be rigorously tested to find errors that would otherwise be undetectable when testing individual pieces of code.

6.1 System Overview

The system itself can be divided into four sections for testing (see Fig 6.1). Testing these individual components in order will effectively test a full path through the system.

Figure 6.1: The four components to be tested

Each of these components had to be tested exhaustively to ensure that the tool will perform correctly and would meet its requirements when deployed in an academic environment.

6.2 Test Plan

In testing this system the goal was to find all feasible paths through each system component and establish that each path results in correct output from the system. This involved firstly finding these paths, and secondly generalising so that a feasible number of test cases are remaining. This involves a combination of both boundary testing and testing failure conditions. It is not feasible to test every possible username and password combination for a login, so the tests are generalised into groups, for example, successful login, unsuccessful login, timed out login etc. The testing cases were restricted to the requirements of the system, I also ensured that the addition of any extra features does not impact on the required features.

The test method used for each component is defined by the method of interaction with that component. The test cases for the login system cover text input whereas the question and answer system were tested by typical browser interaction (i.e. mouse input).

6.3 Test Cases

6.3.1 The login system

1. A valid login

Purpose

The purpose of this test is to verify that when a user enters a valid user id and password the system will log them in correctly.

Input

The input is a valid username, in this case an email address *destraynor@yahoo.ie* and a corresponding password (i.e. an alphanumeric string).

Expected Output

The expected output for this case is that the user is redirected to a new page informing them of their successful login, and is presented with the welcome screen. The tool completed 20 iterations of this task successfully.

2. An invalid login

Purpose

The purpose of this test is to verify that the system will not login users who present a bad username and/or password.

Input

There are three separate inputs for this test case. Firstly a valid username, with an invalid password, followed by an invalid username with

a valid password, and finally an invalid username and an invalid password.

Expected Output

The expected out for this case is that on all three occasions the user is asked to login again, reminding them that passwords and user names are case sensitive. The tool completed 20 iterations of this task successfully.

3. **A time out during login**

Purpose

The login procedure is restricted to three minutes in time as a security procedure. If a user spends more than three minutes logging in, the tool will ask them to re-enter their username and password.

Input

The input in this case is a valid username and its corresponding password. However upon visiting the page, there is a delay greater than three minutes before entering the details.

Expected Output

The expected output for this case is that although the input is valid the random number that is used for each login session has expired, therefore the login is invalid. The users is ask to re-enter their details. The tool completed 5 iterations of this task successfully.

6.3.2 The question and answer system

1. **Answering a question correctly**

Purpose

The purpose of this test is to verify that when a user selects the correct answer to a question, the tool acknowledges it and awards the user a mark.

Input

The input to this test is a username (i.e. the student taking the test), a question id, and an answer id.

Expected Output

The expected output for this case is that upon submitting the answer the user is brought to a page informing them the answer was correct. It should also update their file to represent their new score. The tool completed over 100 iterations of this task successfully.

2. **Answering a question incorrectly**

Purpose

The purpose of this test is to verify that when a user selects the incorrect answer to a question, the tool will inform the user that they are wrong.

Input

The input to this test is a username (i.e. the student taking the test), a question id, and an answer id.

Expected Output

The expected output for this case, is that upon submitting the answer the user is informed the answer chosen was wrong and given a hint which may help them in future. The users score should remain unchanged. The tool completed over 100 iterations of this task successfully.

3. Using the Back Button

Purpose

The purpose of this test is to ensure that students cannot re-answer questions using the back button in the browser. The back button returns a user to their previous page(s), and could therefore enable them to re-answer old questions.

Input

This case tests a session where the user has answered the previous question incorrectly and has returned to the same question to re-answer it. The inputs are therefore a username, a question id and an answer id.

Expected Output

The expected output for this case is that the tool will inform the user that the back button has been pressed and explain that as a result the outcome of the current question will be disregarded. The tool completed 10 iterations of this task (including many different combinations of back button usage) successfully.

4. Using the Refresh Button

Purpose

The tool should present the user with a random question for each unique visit to the questions script. However it must be able to distinguish between a refresh from the browser and a new visit. This test checks whether the tool can do so.

Input

The input is a browser session where the user has been presented with a question, and has pressed refresh hoping to be presented with a new question.

Expected Output

The expected output for this case is that the tool will realise the user has already received a question, and will restate the same question for the user. The actual output was that the user receives the same question, however the answers are in a different order. This has been concluded to be satisfactory output, therefore the tool has completed all iterations of this task successfully.

5. **Examining the level increment**

Purpose

The purpose of this tool is to check, if the tools will advance a user a level when they score a certain number correct on a given level.

Input

The input in this case is a user id in a session where they have scored five correct answers on a given level and a “score for increment of 5” (i.e. if a user scores five correctly they advance a level).

Expected Output

The expected output here is that the user is informed that they answered correctly, and the next question they are presented with is from the higher level. The tool completed 15 monitored iterations of this task successfully.

6.3.3 **Storing a user’s results**

1. **Testing a sample store**

Purpose

The purpose of this test is to verify that the tool stores its results correctly and securely when a student finishes their test.

Input

The input to this test is a user answering their final question correctly where they have answered 3 questions wrong overall.

Expected Output

The expected output is that the user is brought to the final page and informed of their overall score. This score is then stored in the marks database. The tool performed successfully over 20 iterations, although a minor delay (<1 second) was observed when storing results, if two users finish at identical times. This occurred once during testing.

2. **Re-entering an exam session**

Purpose

The purpose of this test is to ensure that once a user completes an exam, they may not begin another or return to their previous one.

Input

There two inputs required to verify this test case. Firstly a user who has finished an exam attempting to log in, and secondly a user who has finished an exam using the back button to re-enter his previous exam.

Expected Output

The expected output here is that first user will not be permitted to login again, and the second user will be informed that his/her session has ended. The tool is capable of doing this because before it attempts a login, it ensures that the user has not already taken an exam and also if a user presses “back” to return to an exam their log file has already been removed so they are unable to receive questions. Over 40 iterations of this task, the tool failed to exit gracefully on 3 separate occurrences. Whilst the tool meets its requirements here, it does so ungracefully at times. If a user is pressing the back button they may get an error instead of a page informing them what has happened.

6.3.4 The Administrator system

1. Adding a user

Purpose

The purpose of this test is to ensure that the administrator is capable of adding a user to the database, and to verify that the user is added correctly. As this is the administration interface, it is reasonable to presume that mischievous input will not occur, therefore the only checking performed is to ensure that both passwords are identical.

Input

The inputs for this case are all the required parameters for a student, i.e. email, firstname, surname, year, course, cao number, class, miscellaneous information and two copies of the same password.

Expected Output

The expected output is, given that both input passwords are equal, the user will be brought to a page showing the user’s details as entered and informing them the user has been inserted into the database.

2. Reloading the question file

Purpose

The purpose of this test is to ensure the that tool can take a valid html file of questions (See appendix C for example of a valid html input file) and insert them correctly into the question bank.

Input

The input to this test is a html file containing 10 questions and all the

information required for each.

Expected Output

The expected output for a syntactically correct question file, is that all questions are added to the questions table in the database, and all answers are added to the answers database. The tool completed this test successfully with 4 separate question files, each inputted many times.

3. Terminating the tool

Purpose

This test is to ensure that when the tool is terminated from the administration interface, that all users still taking tests will also be logged off, and informed that their time limit has expired.

Input

The input is a session, where a user has a test in progress, and the tool has been terminated.

Expected Output

The expected output is that the user's progress in the test has been recorded, but the is unable to finish the test, instead the user is taken to a page explaining why their test is over, and informing them of their score. Their final score, number of questions attempted, username, and the date should be stored in the database. The tool completed 5 iterations of this task successfully.

6.4 Conclusions

During the final testing many alterations were made to the code particularly in the question and answer system, and in storing results. The initial code did behave as required, but I introduced one common error page that the user is brought to when they have performed illegal operations. Adjustments were made in the code to increase performance also, as some slight lagging was noticed.

In conclusion the above testing verifies that all the software requirements placed upon the tool have been completed, and the tool can therefore be considered complete. Little maintenance is expected from the user, as the administration interface provides the majority of the functionality required to reconfigure the system.

Chapter 7

Conclusion

This chapter discusses the development of the project, and explains how several decisions led to a successful implementation an automated tool for e-learning. The first section analyses the development of the project from requirements through to implementation, and the second section discusses future work.

7.1 Analysis of project development

7.1.1 Requirements

The first and possibly most vital stage in the development was requirements analysis. At this point most of the potential pitfalls were avoided through acknowledging conflicts and infeasible requirements. Once the requirements were deemed fully correct and feasible in conjunction with my supervisor, I began to create a finite list of simple tasks that would meet these requirements. The first stage in development was to design a system capable of meeting the requirements.

7.1.2 System design

The particular system design chosen, made the project development straightforward. By designing a centralised system where users communicate with the tool through web pages, it was possible to incrementally develop the tool. Scripts could be written and tested one at a time, and as a result development was constant and uninterrupted during implementation.

7.1.3 Implementation

The implementation strategy was incremental development. One piece of functionality was added at a time, and as each one was added, it was subjected to rigorous testing. The *use strict* [5] module in Perl, prevented me from writing any code that strayed from the requirements or behaved erratically. This slow methodical coding strategy made the testing of all code far simpler, as the majority of bugs and defects were removed during development.

7.2 Future work

This section discusses possible extensions to this current system and also puts forward e-learning ideas inspired by the development of this project.

7.2.1 Extensions to current project

1. An extension I would like to perform on this project would be to test its limits, in terms of concurrent users. During development I was careful not to include any possible deadlocks (caused by two users using the same resource), and I desire to find at what point the tool will cease to perform effectively. Such a test may require up to 200 users or more. However a test suite could simulate this by having several client-side scripts constantly requesting access to the same scripts on the website. The results of this could be observed and while this is not equivalent to mass testing in a real environment, it would give a strong indication of how the tool would cope.
2. At present the tool stores for each answer a feasibility value, which represents how *wrong* any answer is. These figures are at present input by the lecturer as part of the html question file. However the tool could derive these values by studying the frequency of answers chosen. If an answer is often picked, but is wrong it could be said that the answer has a high feasibility because it tricks many students. If an answer is never picked and is wrong, it could be said that the answer has a low feasibility, because students never choose it. Such an extension to the project would hugely increase its adaptive properties, as it makes decision based on previous experience. It is, in effect, using the intelligence of the students to create its own rule based system for reasoning about the feasibility of an answer.

3. Multiple-choice test results are particularly suitable for machine analysis, however any exam that is based on exact input/exact output can easily be turned into a medium suitable for machine analysis. For a programming test, if the test specified that the answer would be the line of output from the user's program, then the user could either (a) upload the program, and the machine could run it and examine its output or (b) type their program into a webpage, where it would be compiled and its execution could be examined. Such a tool could be very useful as it would automate all practical laboratory examinations and could return marks instantly.

7.2.2 Further work related to current project

An Online Lecture Theatre

Many of the current online learning facilities have course content available for download. However the content is static, and other than offering a FAQ, there is no facility for students to ask questions about the material.

I propose to develop an online lecture theatre (OLT), i.e. a virtual lecture hall that students can log in to, and where the lecture is controlled by one administrator (the lecturer). The lecturers interface will have facilities to display powerpoint slides, as well as type text, or draw freehand.

The students interface will display the board (i.e. what the lecturer is currently showing) as well as showing the current students logged in. The students will also have the facility to ask questions. When a question is asked, all students will be able to view both it and the answer given by the lecturer. This will create a classroom atmosphere, where students might not lose motivation as quickly as they would when browsing through static online material. Also, all lectures could be saved for future reference for the lecturer.

A complete and fully functional OLT would have huge potential. As well as use in academia for tutorials, lectures and labs, it could be used for talks, online conferences, and meetings. I firmly believe such a tool would be of significant use for e-learning in both the academic and commercial worlds.

7.3 Summary

The overall aim of this project was to produce an e-learning tool that could address one of the major problems with e-learning: motivation. The goal was to use adaptive questioning combined with a staggered learning system to maintain the independent motivation of the students. The classic pitfall

in developing e-learning tools in the past has been to use a fixed level of difficulty. The problem with a constant level questioning is that while some students will progress with ease and quickly become bored, others will constantly score poorly and become demotivated. In the introduction I identified two possible solutions to the motivation problem in e-learning:

- The system should be able to react to the students level of progress, it should be able to challenge the good students and encourage the weak ones.
- The system should be able to aid the students with difficulties they have. If they have a weakness in one area, it should offer some form of help.

The tool developed for this project has a dynamic level of difficulty that is personalised for each student. The level of difficulty can increase if a student is scoring highly to prevent them getting bored or feeling unchallenged. For the weaker students the level can decrease until a point where the student begins to score highly, thus keeping their confidence and therefore motivating them. The marks awarded by the system are, however, fully transparent. When a student is awarded a mark, also displayed is the level at which they achieved that mark. The lecturer can still find the weak students in a class by searching the marks in order of level. This, I believe shows the systems ability to maintain motivation and encourage students to continue using the system.

This tool is also capable of storing hints and reference pages for each question, so that if it detects a student is weak at a particular topic, it can offer advice on the subject. This shows the system's ability to react to students' weaknesses and aid them with appropriate references and indicate what topics they should revise. It is for the above reasons that I believe that this system provides *personalised testing*, which I have identified previously as a reasonable solution to the motivation problem in e-learning.

The project was finished within its allocated time with all requirements satisfied. I gained much knowledge during research and development, and I am happy to consider this project a success. Developing this tool has inspired me to further research in the field of e-learning, and I very much hope to get the opportunity to pursue it. It is my hope that this tool will be used in academic environments, and that the end-users (both lecturers/administrators and students) will be pleased with its performance.

References

- [1] Dr. Brandon Hall, '*Market Analysis of the 2002 U.S. e-learning Industry: Convergence, Consolidation and Commoditization*', 2002.
- [2] Larry Wall, Tom Christiansen, Jon Orwant *Programming Perl (3rd Edition)*, O'Reilly Publications, California.
- [3] Alligator Descartes, Tim Bunce *Programming the Perl DBI* O'Reilly Publications, California.
- [4] Paul DuBois, *MySQL and Perl for the Web*, New Riders Publishing, Indianapolis.
- [5] Eric Foster-Johnson, *Perl Modules*, Hungry Minds Inc., Indianapolis.
- [6] Ben Laurie, Peter Laurie, *Apache: The Definitive Guide (3rd Edition)*, O'Reilly Publications, California.
- [7] Saul Carliner, *Designing E-Learning* , American Society for Training & Development.
- [8] John Salvia, James E. Ysseldyke, *Assessment (8th Edition)*, Houghton Mifflin College, Boston.
- [9] Marcy P. Driscoll ,*Psychology of Learning for Instruction (2nd Edition)*, Allyn & Bacon, Boston.
- [10] Ruth Colvin Clark, Richard E. Mayer, *e-Learning and the Science of Instruction* , Jossey-Bass, Indianapolis.
- [11] Larry A. Mallack, '*Challenges in Implement e-Learning*', Management of Engineering and Technology, 2001.
- [12] Chassie, K., *The allure of e-learning* , IEEE Potentials , Volume: 21 Issue: 3.

- [13] Hamlet, R., '*Unit testing for software assurance*', Computer Assurance, 1989.
- [14] Offutt, A.J., '*Unit Testing Versus Integration Testing*', Test Conference, 1991, Proceedings.
- [15] Question Mark Computing Limited, London, <http://www.questionmark.com>.
- [16] SkillCheck Professional Corporation, Massachusetts, <http://www.skillcheck.com>.
- [17] Blackboard International Corporation, Amsterdam, <http://www.blackboard.com>.
- [18] Chariot Software Group Corporation, California, <http://www.chariot.com>.
- [19] Riverdeep Interactive Learning Limited, Dublin, <http://www.riverdeep.net>.
- [20] Butler W. Lampson, '*Computer Security in the Real World*', Microsoft 2000, <http://research.microsoft.com/lampson/>
- [21] The MySQL website <http://www.mysql.com>
- [22] <http://www.MySQL.com/information/benchmarks.html>
- [23] The Apache website <http://www.apache.org>
- [24] EPIC Group, 'Motivation in E-learning', pp.3-5.

Appendix A

The User Manual

This section will outline what tools were used, explain their functionality, and explain the importance of each. See table A.1 for a list of the names, version numbers, and, locations of each of the following tools.

1. **Apache Webserver**

This is the webserver that allows the computer hosting the tool to communicate over the Internet. When a user visits the webpage, Apache controls the information flow to and from the user. Apache is the most popular webserver on the Internet, and is available for most platforms.

2. **Perl Scripting language**

The Perl scripting language is a highly important tool for the system. Perl is one of the most popular scripting and CGI programming languages available today. Perl is the language that all the scripts are written in, it receives input from the user, queries the database, and generates web pages for the users.

3. **MySQL**

MySQL is the database driver that is used for storing the information held in the database. MySQL is a freely available high speed database driver that is highly popular and communicates easily with Perl.

4. **MySQL Control Center**

This is an optional add-on to MySQL. The Control Center is a graphical user interface where you can alter the structure of your database. MySQL Control Center reduces the level of SQL knowledge required by the user.

Name Of Tool	Version Used	Available From
Apache Webserver	2.0 for Windows NT	http://www.apache.org
Perl Scripting Language	5.005_03 for MSWin32	http://www.ActiveState.com
MySQL database server	3.23.53-max-nt	http://www.mysql.com
MySQL Control Center	0.8.6-alpha	http://www.mysql.com

Table A.1: The required programs and tools

A.1 Installation Procedures

In this section I hope to give as precise a description of the installation procedures as possible. Correct installations are important for effective performance.

A.1.1 Installing apache

To install Apache, go to the website, and download the most recent **stable** release for the operating system you are using. I developed this tool on a Windows NT 4.0 platform, and I download the appropriate version at the time. The installation procedure is reasonably straightforward. You will be asked for information such as name, email address, etc. The email address you give should be a suitable lasting email address for you, as it will be displayed in all Server error pages. (E.g. Server error, please contact *destraynor@yahoo.ie* for further details.)

A.1.2 Installing Perl & its modules

Installing Perl (only if necessary)

From the website, select “Perl”, then under “resources” select “activeperl”. This version is standard across platforms, and comes with an installer. Installation is relatively short but may require a reboot to finish. You should install to the default directory C:\perl. If you choose not to, you will have to change the 1st line of every perl script to point at your chosen location. You should also ensure the Perl directory is in your classpath. See “environment variables” under properties “of My Computer” for details of your classpath.

Installing the required modules(necessary)

From the command prompt type ‘ppm’. This will start the Perl package monitor which will locate and install perl modules for you. To search for a file in the archive type ‘search *filename*’, to install one type ‘install *package-name*’. Locate and install the following modules:

DBI.pm The database driver module.

DBD-mysql The MySQL interpreter.

Digest.pm The Encryption Module.

These files can also be downloaded from the Comprehensive Perl Archive Network (<http://www.cpan.org>).

A.1.3 Installing MySQL

Download and unzip the binary distribution for MS Windows. Run setup.exe. This will install the MySQL program on your computer. To install MySQL as a regular service on your computer type 'mysqld-nt -install' followed by 'NET START mysql'. The next time the machine is booted the MySQL service will be started. Ensure the directory containing MySQL is in your class path for ease of use.

A.1.4 Installing MySQL Control Center

Select MySQL Control Center from the webpage, download the installer file, and run it. If you have already created a database, you will need to connect to it using the username and password. No passwords or usernames are set until you create your first database. If you have yet to create a database the control center will aid you greatly, saving you writing SQL for table creation etc. An illustration of this graphical user interface can be seen in Fig. A.1.

Figure A.1: MySQL Control Center

A.2 Installing the tool itself

This section explains how to install the automated tester, given that all the previous programs are installed and running.

A.2.1 Installing the scripts

Table A.2 contains a list of all the scripts. These all belong in the Apache/cgi-bin directory.

Also required in the cgi-bin is the file 'config' which contains the global environment variables. The file is a '|' separated line of variables, and can

Name Of Script	Function
adduser.cgi	This script takes a user's details and calls storeuser.cgi.
please_login.cgi	This script enables a user to login.
reload.cgi	This script reloads the questions database.
store.cgi	This script stores an answer when given.
storeuser.cgi	This script places a user in the database.
testv2.cgi	This script selects and asks the questions.
viewall.cgi	Views all students from a given class.
viewdate.cgi	Views all test results from a given date.
viewuser.cgi	Views all details regarding one user.
pass.pl	Checks if a password is valid (called by please_login).
test.pl	This is a script that states the environment variables of the system.

Table A.2: The required scripts and their functions

easily be queried by running the file test.pl. This file will have to be edited, as it holds amongst other information, the database server location, the password, and the administration computer IP address.

A.2.2 The html files

There are three html files which should be placed in the directory 'Apache htdocs'. There is an index page (index.html), welcoming the users to the site, an administrator page (admin.html) from which the administrator can run administration scripts, and there is an error page (error.html), where the students are taken to if the tool is not switched on.

A.2.3 The database tables

There are five tables which must be created for the database. These are the answers table (see Fig A.3), the marks table (see Fig A.4), the questions table (see Fig A.5), the sessions table (see Fig A.6), and the students table (see Fig A.7).

1. **answers**

Name of field	Represents	Type
answer_id	The unique answer id	int(11)
answer	The text of the answer	blob
question_id	The unique question id it relates to	int(11)
feasibility	The measure of how feasible this answer is	double
position	Where the answer should be located	enum(positions)

Table A.3: The answers table

2. marks

Name of field	Represents	Type
email	The users email address	VARCHAR(255)
date	The date of the test	date
number_questions	The number of questions taken	int(11)
mark	Their score	int(11)
level	The finishing level of the user	int(11)

Table A.4: The marks table

3. questions

Name of field	Represents	Type
question_id	The unique question id	int(11)
type	The type of the question	VARCHAR(255)
question	The text of the question	blob
link_hint	Link to a hint	VARCHAR(255)
link_answer	Link to be given once question is answered	VARCHAR(255)
level	The level of difficulty of the question	double

Table A.5: The questions table

4. sessions

Name of field	Represents	Type
IP	The IP of the user logging in	VARCHAR(20)
rand	Their current random number	int(11)
time	The time of the login	int(11)

Table A.6: The sessions table

5. students

Name of field	Represents	Type
email	The users email address	VARCHAR(100)
first_name	The users first name	VARCHAR(100)
surname	The users surname	VARCHAR(100)
year	Year they are registered with the tool	VARCHAR(4)
course_number	The number of the course they are enrolled in	VARCHAR(10)
cao_number	The users CAO number	VARCHAR(10)
class	The class they are in	VARCHAR(100)
misc	Miscellaneous Information	VARCHAR(40)
password	This stores a hash of the users password	VARCHAR(40)

Table A.7: The students table

A.3 Testing the Install

To test the install, you must perform the following steps.

1. Ensure the tool is turned on, and that your webserver and database server is up and running.
2. On the administration page (admin.html) select ‘create a new user’, and enter some sample details.
3. When the user has been created, select ‘view details about a new user’, and enter the username. You should see all the information that you entered earlier. This tests that the scripts can read and write to and from the database.
4. Ensure that there are some questions in the database. The ‘reload.cgi’ will read in a html file full of questions and enter them in the database. For an example of such a file, see questions.html.

5. Visit the script 'please_login.cgi' and login using your username and password.
6. Take a full test, and view results.
7. Verify these results from the administration page using either viewall.cgi, viewdate.cgi or viewuser.cgi.
8. Turn the tool off and attempt to login. It should fail.

This should perform a proper test on all scripts, and will establish whether the tool is working correctly.

Appendix B

New Material and Skills Acquired

This chapter outlines the materials learned and skills acquired throughout this project. In the first section I address the theoretical knowledge I gained as a result of background reading and research, and the section second outlines what practical skills I gained as a result of implementation.

B.1 Materials Learned

This project required me to become familiar with the background concepts in e-learning, and the problems inherent in it. I studied many papers and texts on developing an e-learning product ([7],[8],[10],[11],[12]), and I became very familiar with the product domain as a result.

Another important subject I researched was the psychology involved in both teaching and testing [9]. This was an important aspect as it was required for designing a tool which aims to maintain students motivation.

B.2 Skills Acquired

While developing this project I became very familiar with CGI programming ,and also with the database language SQL.

This project has substantially improved my programming ability and also my ability to realise requirements with software development. The following is a detailed list of the skills gained.

B.2.1 Perl

At the beginning of this project my knowledge of the Perl language was limited to simple text processing and regular expressions. During development however, I acquired a lot of knowledge about the more popular Perl modules. I am now familiar with the Perl CGI module (for web based scripting) and also its database interface modules. As a result of this project I am now very familiar with using scripting languages to interact with both the user and a database, and as these are the primary concepts for developing most websites, this knowledge will undoubtedly be useful.

B.2.2 MySQL

I had previous experience with SQL before beginning this project, but it was mainly limited to simple queries on sample databases. This project required me to design a database structure that would accurately model the system and fulfill the requirements. While designing the database I gained much knowledge of the Entity Relationship model, and also greatly added to my knowledge of SQL.

B.2.3 Apache

This project required me design a web based tool. The first stage in doing so is creating an area that is web accessible. I had no previous experience with web servers, their installation, or their configuration at the beginning of this project. I gained a vast amount of knowledge of the Apache webserver by using it to host the online tool I developed.

B.2.4 LaTeX

LaTeX is an interface to TeX (a typesetting tool developed by Donald Knuth), which is highly suited to writing academic papers. LaTeX was recommended by my supervisor at the beginning of this project I have used it to write every document since. I had no previous experience with LaTeX, and consider a highly valuable skill to have gained.

B.2.5 Project Management

As a result of undertaking this project I have gained invaluable experience in project management. I have learned how to allocate time, perform task

analysis, following software development cycles, and most importantly design, implement and test a tool based on a requirements set.

It is clear from the above skills and materials that the undertaking of this project caused me to acquire many skills all of which will be highly useful throughout my academic career.

Appendix C

Project Plan and Tasks assignment

This section shows the task breakdown of the project, and details the time allocated to each task and the order in which they must be completed.

The first step in planning the project was to break it down into a finite list of simple tasks. This was easier to do, once I had some ideas about how to design the project. Once I had a list of tasks, I then had to decide the best order in which they must be completed and allocate a suitable amount of time for each of them. The tasks are lettered, with explanations and dates assigned in Fig. C.1.

I then drew a Gantt chart for my project. A Gantt Chart shows each significant step of a project as a line on a time-based chart. The Gantt chart also clearly illustrates the three stages in this project, design, implementation, and documentation. The Gantt chart can be seen in Fig. C.2

The project plan as it is shown was followed reasonably well, certain tasks had to be re-ordered and dates changed. The most notable difference between the development of the project and the project plan was that documentation began earlier than allowed for. The plan did not allow time for a requirements document to be drawn up and agreed on at the beginning, or requirements analysis documents or design documents for example. To compensate for this, several aspects of coding and testing were postponed until february and carried out at points when I was awaiting feedback on other work. Another example of this is how the requirements were altered slightly during development, and as some tasks became more complicated, they required more time allocated.

I believe that the task analysis, and its corresponding project plan were reasonably accurate estimates, whilst the development may have deviated slightly from the plan, it was for the most part as a result of altering require-

Figure C.1: The Tasks breakdown for development of the project

Figure C.2: The Gantt chart representing the completion of the product over time

ments or necessary documentation work.

Appendix D

Sample data for inputting questions.

This section shows an example of how the lecturer would input questions to the tool. The format chosen for question input was chosen to be html, as it gave the lecturer the advantage of being able to see how questions will look in a browser before uploading them. An example of the html input, can be seen in Fig D.1, and a sample of its corresponding output in a browser can be seen in Fig D.2. When the lecturer/administrator adds a question (as shown in Fig D.1) to the system, the html preamble tags are automatically stripped off and the remainder incorporated into the question database.

Figure D.1: A sample of the html input from the lecturer

Figure D.2: What the lecturer can view before submitting the questions