

Attendance Monitoring Of Students

Paul Mowat

0404695



The Robert Gordon University

School of Computing

Supervised By

Mr Niccolo Capanni

Declaration

I confirm that the work contained in this report has been composed solely by myself. All sources of information have been specifically acknowledged and verbatim extracts are distinguished by quotation marks.

Paul Mowat

0404695

Abstract

Throughout the course of an academic course there are many classes in which students are required to attend. The lecturer in charge must record each of the student's attendance for each of these classes. This takes a considerable amount of time away from the actual teaching of the subject. It also requires a substantial amount of work for both the lecturers, and also the administrative staff. Without attendance being recorded then it is likely that students will miss more and more classes and not realise how much subject material they have missed.

The project has been developed to automatically record a student's attendance when they log into a computer system within a class they are taking. The system has the ability to analyse a student's attendance data and display information about their attendance directly to them. If a student's attendance is poor then they will be able to see it against the class's average and hopefully this will help them improve their overall attendance. The system also has the ability to integrate with other systems by exporting attendance data as comma separated value files.

The completed project aids university staff in recording attendance data, and also frees up a lot of time, which was previously used to record, collate and process attendance information. This system does all of this automatically and also more records attendance information more precisely as it cannot be faked by student's friends. In short this system provides a quicker, more efficient, more accurate way to record student's attendance.

Contents

Word Count: 11295

1 - Introduction	1
1.1 - Project Overview	2
1.2 - Project Justification	3
1.3 - Personal Motivations.....	4
2 - Requirements Analysis	5
2.1 - Project Aims and Objectives	6
2.2 - Functional & Non-Functional Requirements.....	8
3 - Research	10
3.1 - Overview Research.....	11
3.2 - Impact Attendance Has On Further Progression	11
3.3 - Analysis of Current System.....	12
3.4 - Analysis of Similar Systems	14
3.5 - Technology Research	15
3.6 - User Analysis	18
3.7 - Software and Operating System Selection.....	19
3.7.1 - Server Side Scripting.....	19
3.7.2 - Databases	21
3.7.3 - Operating Systems	23
4 - Design	25
4.1 - Design Methodology	26
4.2 - Systems Analysis.....	27
4.3 - Site Map.....	30

4.4 - Interface Design	31
4.5 - Database Design.....	36
4.6 - Server Design	43
5 - Implementation	44
5.1 - Implementation Overview.....	45
5.2 - Problem Analysis & Solutions	55
5.3 - System Testing	59
5.4 - Test Results	60
5.5 - Critical Evaluation and Conclusions	62
6 - Reflection	65
6.1 - Future Developments.....	66
6.2 - The Project: A Retrospective View.....	68
6.3 - Acknowledgements	69

1 - Introduction

1.1 - Project Overview

The current system for student attendance requires weekly production of a paper based attendance log, which is then copied into the electronic log system; if errors are made then any subsequent amendments must to be carried out manually, which can take a substantial amount of time. Access to this information requires an administrative request, which when added together will waste substantial amounts of valuable time.

This method is very high maintenance, inflexible and as already pointed out very time dependant. Therefore there is need for an electronic equivalent system. With this in mind the purpose of this project is to make attendance monitoring more efficient, effective and also less time consuming to Staff, Students and University administration.

Attendance monitoring is directly used by all Universities to highlight problems with student performance, and to act as an aid in dealing with any problems encountered. The current system does nothing to encourage attendance except by using the fear of punishment. This often happens too late as the student has either failed the course or already graduated with a low grade.

With an electronic system in place the student will have access to how well or how badly their attendance is, in comparison to the overall class percentage. If any student has exceptional attendance then they could also gain appraisal. However, if any student's attendance is poor then this information could possibly give them that extra bit of motivation to get their attendance back up to the target percentage. This in turn will possibly help them gain higher grades and an overall better degree classification.

1.2 - Project Justification

The main aim of this project is to make attendance monitoring easier for University staff, this will be achieved by producing an automated attendance monitoring system. At the moment lecturing staff have to spend considerable time in lectures, tutorials and labs taking attendance. Then on a weekly basis this information is collated and inputted into the system by administration staff. This has many head-aches for lecturing and administration staff as it takes up a considerable amount of their time, which could be used for other tasks.

The current system also has problems when it comes to getting a precise attendance; people come in late, leave early or have friends sign them in. The proposed system will help stop that from happening as the students attendance is recorded automatically when they log into their computer account.

Below is a list of advantages and disadvantages for the proposed system.

Advantages:

- Improved accuracy of attendance.
- Less time spent by staff during class taking attendance.
- Easier monitoring of potential problems.
- Help encourage students with low attendance to attend more.
- Improved academic appeal for students, attendance can be used as measurement of performance and effort.

Disadvantages:

- Computer system can crash and possibly lose data.
- Must have sufficient security protecting data.

1.3 - Personal Motivations

After having successfully completing a HND in Software Development at Fraserburgh College, and then the degree year at Robert Gordon University I decided to come back and complete the honours year. Having excelled in the Internet Based Programming module last year, and having a large interest in web development and intranet based systems I felt I should undertake a project in this field.

The project offered the opportunity to work on a large scale intranet application which requires a large database, which will help increase my knowledge of databases greatly. The project will also require the use of technologies that I previously have no experience in. Due to this I will be able to increase my skillset, which in turn will help aid my search for employment after graduating.

2 - Requirements Analysis

2.1 - Project Aims and Objectives

Aim

- Design and develop an automated attendance system to aid University staff in monitoring student's attendance.

Objectives

The system should:

- Support multiple administrators and lecturers with different privilege levels.
- Allow administrators and lecturers retrieval of relevant attendance records.
- Allow exporting of attendance into a format that can be integrated with the current system
- Allow the administrator to add/edit/delete and view.
 - Students.
 - Courses.
 - Modules.
 - Classes.
 - Lecturers.
- Allow the administrator to import a list of students.
- Allow automated entry of student attendance.
- Provide feedback of attendance average to student.
- Interfaces are simple to use, consistent and behave in a consistent manner.

The aim of this project is to provide an electronic system that will aid University staff with the recording and monitoring of student attendance. It is important that this system is robust, secure, stable and easily upgradeable; for the inclusion of new features.

When a student logs into their network account the student interface will automatically find out which class that student is meant to be attending and then record their attendance for that class. The system will provide three types of user interface; administrator, lecturer and student.

The administrator interface will allow those with administrator privileges to add, edit, delete or view records in the database. Further tools, such as the ability to import a comma separated value (CSV) file list of students will also be included.

The lecturer interface will provide lecturers the ability to view or amend the attendance of classes that they lecture. They will also be able to view details of any student that undertakes one of their classes or to cancel a class.

The system must also be able to output data in a format that can be used in conjunction with the current method used for monitoring student attendance.

2.2 - Functional & Non-Functional Requirements

Functional Requirements

1. Student attendance automatically recorded and entered into database
2. Authorized users can add, view and amend information in the database
3. Import file that contains a list of students to be added into the system.
4. Provide feedback of attendance average to student.
5. Interfaces should
 - a. Provide a clear, consistent layout and load quickly.
 - b. Provide feedback to the user if needed.
 - c. Minimize amount of user input required.
 - d. Clearly displayed search results
6. Performance Criteria
 - a. Server side processing to be kept at a minimum.

Non-Functional Requirements

1. Development Constraints
 - a. Access to project supervisor limited to 30 minutes per week
 - b. University opening hours limit access to library and labs
 - c. Time spent carrying out coursework for other modules
 - d. Exam revision and preparation in January 2005 means little work completed in this time (as noted in Gantt chart).
 - e. All project work must be completed and submitted by the 29th of April 2005

- f.** The server side applications required by this project are not widely available on the university computers. Therefore they must be installed onto a separate machine connected to the university network.

2. Quality Control

- a.** System will adhere to current (2005-2006) W3C standards for web development
- b.** Code will contain descriptive comments
- c.** Macromedia Dreamweaver will be used to create default template documents, which are then edited accordingly.
- d.** Project will be managed using Microsoft Project 2003
- e.** Minutes will be taken for all meetings that are conducted

3 - Research

3.1 - Overview Research

While researching the project there were 5 main areas that had to be researched. Firstly the impact that attendance had on further progression. Secondly the current and similar available Attendance Systems were analysed. Thirdly different technologies used in existing attendance systems. Fourthly, research into the potential users of the system, and finally research into which software will be used to implement the system.

3.2 - Impact Attendance Has On Further Progression

Students who miss a large percentage of classes are likely to have problems when it comes to the completion of coursework and exams. Whether the absence is caused by laziness or because of genuine reasons, there will most likely be difficulties. No matter what, there will always be a small minority of students who have poor attendance and still however manage to pass with top marks. This could simply be down to previous knowledge of the subject area, or a greater ability to learn material by themselves from online notes.

The co-author of the book "Retention and Student Success in Higher Education" Mantz Yorke (Personal Communication: 5/11/05), when asked the following question "What are your opinions/thoughts on students who have poor attendance but still receive good grades?" answered "A self-confident independent learner may be able to get what s/he needs from electronic sources, books etc and not need to be at formal teaching sessions (for some subjects, at least). Others might need those sessions to help them structure their learning".

This may be the case but in general though there is quite a noticeable gap in grades between those who attend compared to those who do not (Habib-ullah Khan, 2003) and this will stay the same until there is a more improved way to monitor attendance and/or deliver information electronically over the internet. Lack of attendance has adverse effects on progression as students who fail either drop out or have to repeat the year.

3.3 - Analysis of Current System

The system that is currently in use in the School of Computing is a paper based one. At the beginning of each semester the School of Computing clerical assistants produce class registers (Appendix 1) and Microsoft Excel Spreadsheets (Appendix 2). These are created in accordance to the timetables off each course.

Both the class register and spreadsheet file contain the following details:

- Module number.
- Type of class (lab, tutorial or lecture).
- Lecturers Name.
- Day of week, class start and end time.
- List of course id's which attend class e.g. IM4, IST4, and CS4.
- List of students containing matriculation number, first name, and surname.
- List of weeks for semester.

Once these files are created the registers are then collected by the lecturers, and the Excel spreadsheets are placed in an ordered hierarchy within the clerical assistant's computer.

During the class the lecturer hands out the registers to be completed by the students.

However the following problems arise when this method is used:

- Students falsely signing in others.
- Students arriving late and leaving early.
- Students taking extended breaks within labs.
- Time spent by lecturers retrieving registers from the clerical assistants, getting students to complete them and then returning the completed register to the clerical assistants for processing.

Once the class registers have been completed and returned to the office the clerical assistants then collate the registration information and spend a considerable amount of time manually entering data into the corresponding Microsoft Excel spreadsheet for that class.

If a student is absent, then while entering data into the Excel file the clerical assistant places a 1 under the week for which they were absent. For each subsequent time a student is then absent from that class then that number increases by one, for example: 1 to 2 and then 2 to 3.

Once the student meets the target level for poor attendance, then they are flagged up by the spreadsheet. The route leader for the student's course is then contacted by clerical staff, and an email is sent to the student by the route leader to arrange a meeting. If the student does not turn up to the proposed meeting with the route leader then the situation progresses through the ranks and the student is dealt with.

The current system also allows self declaration of absence (Appendix 3). This means that if a student knows they will be off for some reason or another then they can declare it before hand. This is done by visiting the office and completing a self declaration form, this form is then inputted into the system by the clerical assistant.

3.4 - Analysis of Similar Systems

While researching into automated attendance I came across some systems that are similar to the system proposed for this project. Most of these systems have however been designed with business usage in mind and use barcode readers to keep track of employees.

However there is one system developed by Tokario Ltd (2004) called the TokAM – Attendance Management System that differs from these by focusing on attendance monitoring for colleges and universities. This system has been designed and developed for over nearly nine years with the aid of many colleges within the UK. Tokario Ltd also claims over seventy further education colleges are using its TokAM software within the UK alone.

The TokAM system also offers the following functionality:

- Paper registers which are scanned into the system with a 98% no-correction rate.
- Direct data entry into the system.
- Entry via online web registers.
- Multiple customizable reports.

Alternative systems have been created for schools, one of which has been developed by a company called Synel (2005). This system however is not flexible enough to be implemented within a college or university, because it only records attendance based on morning and afternoon and not specific classes.

3.5 - Technology Research

At the moment there are many different types of technologies used to aid in monitoring attendance and time-keeping. These have to be flexible so that they can be easily implemented by different organizations and establishments.

Below is a list of different types of technology currently used, a brief description, its advantages and also its disadvantages.

Barcode Readers

A simple card that contains a barcode is used to identify an individual; this card is swiped through a terminal. The terminal then sends the interpreted data to a computer that is connected to it, and the computer processes that information.

Advantages:

- Easy and intuitive to use, little learning needed.
- Quick to use.

Disadvantages:

- Can be stolen or lost.
- Students can still lend their card to friends, and have them sign-in for them.

Fingerprint Recognition

According to Idteck (2005a) "Fingerprint recognition is the technology that verifies the identity of a person based on the fact that everyone has unique fingerprints"

Advantages:

- Everyone's fingerprints are unique to them.
- Equipment is now low priced compared to a few years ago.

Disadvantages:

- Some people have damaged fingers.
- Some people may not want their fingerprints stored.
- Can possibly be faked.

Iris Recognition

Iris Recognition is the scanning of the human iris that resides within the eye. A picture of the iris is taken, and this is then verified against the person's eye each time they require authorization (Idteck 2005b).

Advantages:

- Fully developed 18 months after birth.
- More unique than fingerprints.

Disadvantages:

- Invasive Method.
- Equipment & Hardware is costly.

Signature Recognition

Signature recognition involves taken a signature and matching it against an existing signature that already resides in a database (Bioenable 2005).

Advantages:

- People are already used to using signatures e.g. Credit Cards & Checks
- Invasive, compared to other technologies.

Disadvantages:

- Signatures can be faked.
- Signatures must be consistent or there may be problems with verification.

3.6 - User Analysis

The user analysis was undertaken to determine who would be using the system. The result of the analysis would be that there were three groups of users.

- Administration
- Lecturers
- Students

The first group Administrators will have complete access to all of the system by using the administration interface. This group will be responsible for entering and editing of system data. They will also have access to all areas of the system.

The second group Lecturers will have access to some of the same features as Administrators. However they will only be allowed to view the data and not edit it in anyway. Also a Lecturer can only view details of classes that they actually Lecture.

The third group Students will only have access to the Agent located on each computer. The agent will record the student's attendance; also the student can view their average attendance for the current class they are currently attending.

3.7 - Software and Operating System Selection

To successfully implement this project three main software applications are required.

Firstly an operating system is required to run the server on which the system is located.

Secondly a scripting language is required to communicate with the database, server and to manipulate pages. Finally a relational database is required in order to store and retrieve the required information.

3.7.1 - Server Side Scripting

The four main scripting languages out there are PHP, ASP, JSP and Coldfusion. They all have different syntax styles; require different operating environments and costs.

PHP is the leading scripting language used on millions of websites around the world (THE PHP GROUP, 2006). PHP is designed to be HTML embeddable, simple to learn and also has a common syntax style with other programming languages such as C and Java. The PHP scripting language is open source, available for free and has a large active online community that provides support.

The PHP language has excellent database support for a large variety of databases, such as MYSQL, PostgreSQL, Access, Microsoft SQL and Oracle. It also has a large function set to provide common tasks such as reading/writing files, reading/writing cookies and manipulating data. PHP is also very portable and can be used successfully with most current web servers.

ASP is a commercial scripting language available from Microsoft, and as such it has a large amount of support available. ASP uses a syntax that is similar to the Visual Basic programming language. It however does have one main downfall, this is that it requires a Microsoft Windows based server (Microsoft Corporation, 2006). To use the full development potential of ASP an IDE is required. One such IDE is Microsoft Visual Studio; which has a substantial cost of approximately £1800. However Microsoft do now offer an extremely cut down IDE called Visual Web Developer for free.

JSP is a Java based equivalent to PHP and ASP. It is developed by Sun Microsystems and uses a dynamic page development environment that can fully interact with the complete Java programming language. The development is totally platform independent and is free to use. The major draw back with JSP however is due to its extensive functionality, which takes a great deal longer to learn. With this in mind and due to the timescale of this project, it has been decided that JSP goes far past the requirements of this system.

Coldfusion is a tag based language developed by Macromedia. It is completely different from the other languages listed above as it is not actually a programming language but a set of tags that can be integrated into html code to allow certain tasks to be completed. This tag based system is very easy to learn, however at the time of writing there were only Coldfusion is also extremely expensive to purchase and to use, as it requires a dedicated server. With this in mind it cannot compare to languages such as PHP and JSP which offer more functionality and are free.

To complete this project it has been decided that PHP best matches the requirements, it's free, offers good database support and is relatively easy to learn.

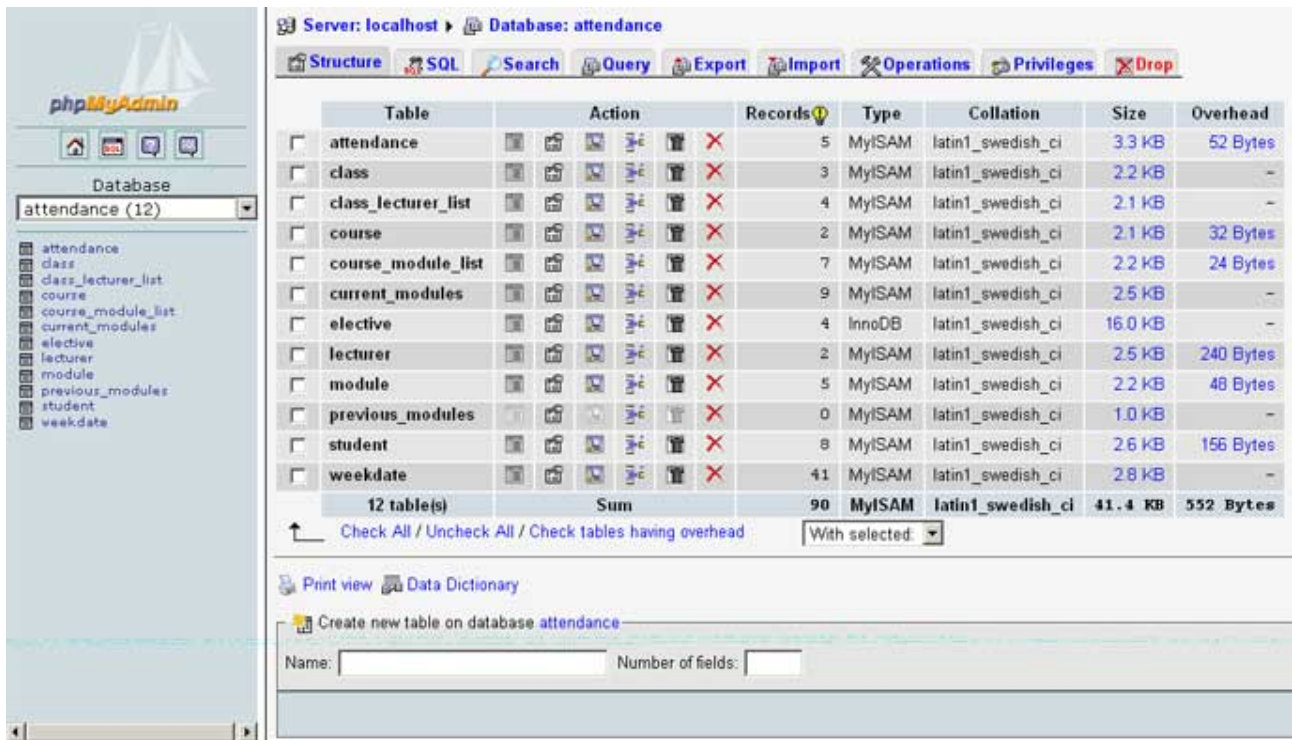
3.7.2 - Databases

The selection of a database is a very important part of the project. This is because all the required data will be stored and retrieved from within it. The database must be able to handle multiple concurrent users, have a simple to user administration interface and also be robust.

MYSQL is a free open source database that is very popular among the web developer community. It can be run on a large variety of operating systems (MYSQL ABa, 2006) and supports a large amount of concurrent connections. This typically ranges from around five hundred to one thousand depending on the available hardware and operating system (MYSQL ABb, 2006). MYSQL also has the ability to support many programming and scripting languages such as PHP, ASP, JSP and Coldfusion with build in or downloadable extensions.

Administration of MYSQL uses a command line interface by default. This interface works however it is hard for inexperienced users to use, as they do not remember commands. With this in mind there are many third party graphical user interfaces available to help make administration simpler and quicker. One of the most popular of these is PHPMYADMIN (2005) which uses the PHP scripting language to interact with the MYSQL server. This can be seen in Figure 3.7-1 below.

Figure 3.7-1 - PHPMYAdmin Interface



Oracle is an enterprise database system that has an extremely wide range of features. It is available in many different versions. Oracle Inc (2005) offers developers a license that they can use while developing or prototyping an application. However when that application is completed then any subsequent use requires a license for the appropriate version. The standard version retails at £8,714 (Oracle Inc, 2006b) and the enterprise version retails at £23,236 (Oracle Inc, 2006c). With these costs in mind it was deemed that Oracle would not be suitable for this project.

Microsoft Access is probably the most well known database as it is bundled along with Microsoft Office. It can also be purchased separately at a cost of £157.57 (Amazon, 2006). The main problem with using Microsoft Access is it has problems supporting a large number of concurrent users which this project will require (Bertrand Aaron, 2006). With this in mind it was decided that Microsoft Office would not be suitable for this project.

After taking the previous three databases into consideration it was decided to go ahead and use MySQL as the relational database management system.

The reasons behind this decision are as follows:

- It's free and open source.
- Integrates extremely easily with a number of scripting languages.
- Has support for a wide range of operating systems.
- Large active online community to provide support if required.

3.7.3 - Operating Systems

The two main server operating systems are either Microsoft Windows Server or a Linux distribution. These two operating are extremely different in the way they operate and behave. Microsoft Windows has a far greater market share when it comes to personal computers. However when it comes to web servers then Linux along with Apache have a substantial 70% market share in the web server industry (APACHE SOFTWARE FOUNDATION 2005).

A main concern when choosing Microsoft Windows as a server operating system is security. This is primary because Microsoft Windows has a much larger market share in the desktop pc industry and provides a lot of tools that allow feature rich applications. If these tools are configured incorrectly then it is possible for hackers to access the system and cause mayhem. Microsoft Windows also has a large cost associated with it, to purchase a single licence for Server 2003 it can cost anything from £297 to £760 depending on the version.

On the other hand Linux has many different distributions such as Red Hat, SuSE, Fedora and Mandrake. All of these distributions are provided free, however some of the companies do charge for support. This however is not required for the average user due to the large active Linux community online. Linux is also much more secure than Microsoft Windows because it is open source, continually developed and also has a much simpler operating core (kernel).

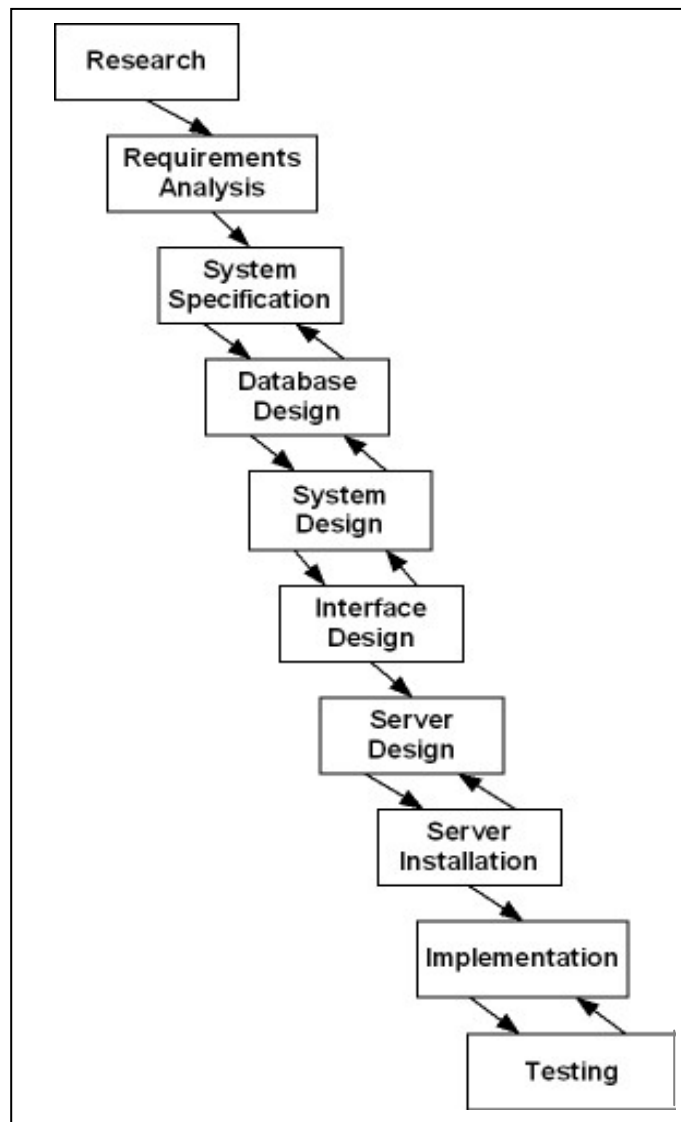
With the security concerns and licensing feeds in mind it was decided to use a Linux distribution on the project server. Fedora was the chosen distribution because it has a simple graphical install process, unlike some other distributions that require manual compiling of the kernel. It also comes with built in configuration options when installing that allow a server to be set up with the minimum of ease.

4 - Design

4.1 - Design Methodology

To model the development lifecycle of the project the SSADM Waterfall model was used. This model is designed to be used on large scale projects and is used in conjunction with a Gantt chart. The Gantt chart was used to keep track of the tasks required, and the time required to undertake them. The waterfall model seen in Figure 0-1 was used to provide a structure in which the system would be designed.

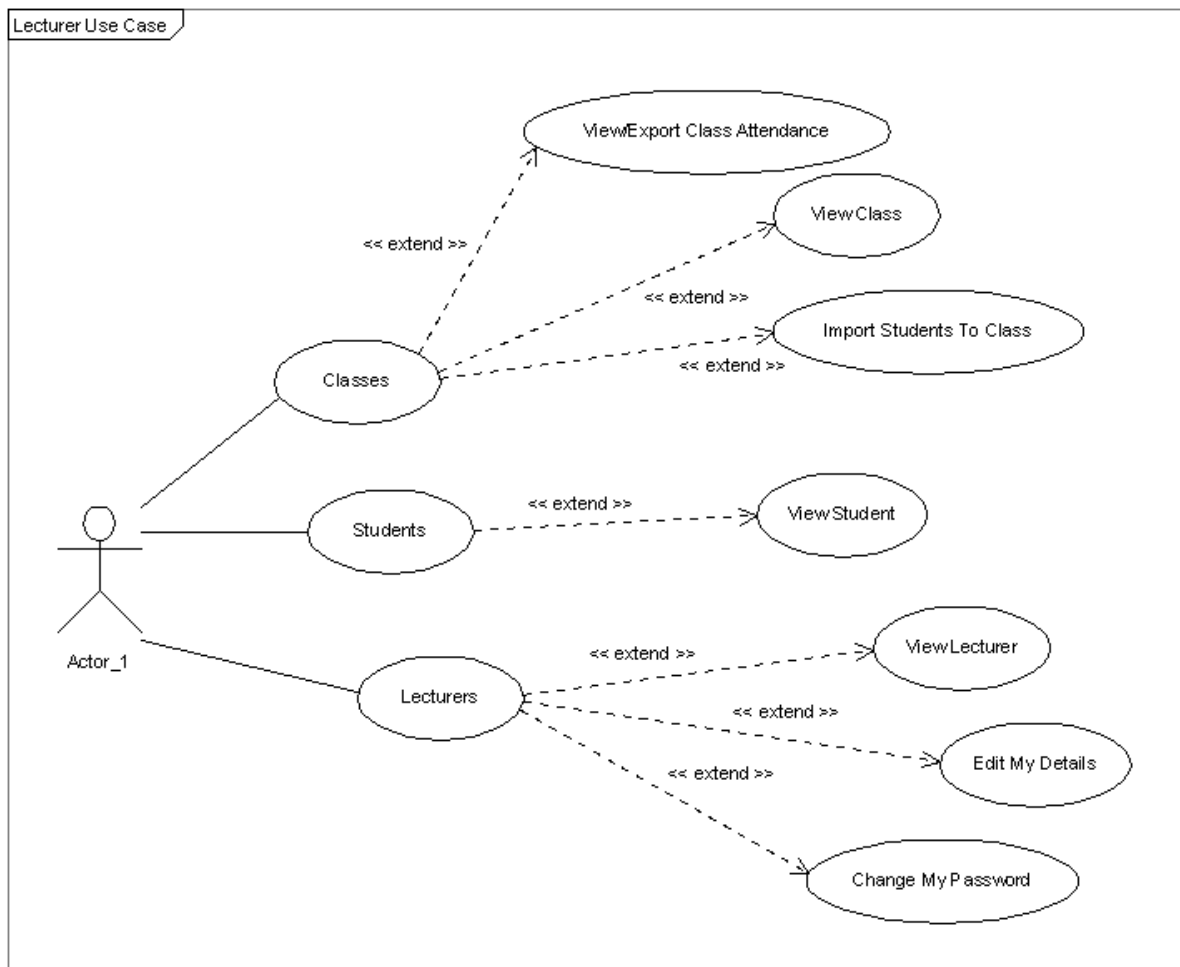
Figure 0-1 – Waterfall Model



4.2 - Systems Analysis

The model used to display the system and its components entirely was the UML case tool. This allowed to have a clear and easy overall view of the system and what can be done. It is used to show all the possible actions that can be used by the user. Figure 4.2-1 below shows the Use-Case diagram for the lecturer section of the system.

Figure 4.2-1 – Lecturer Use Case



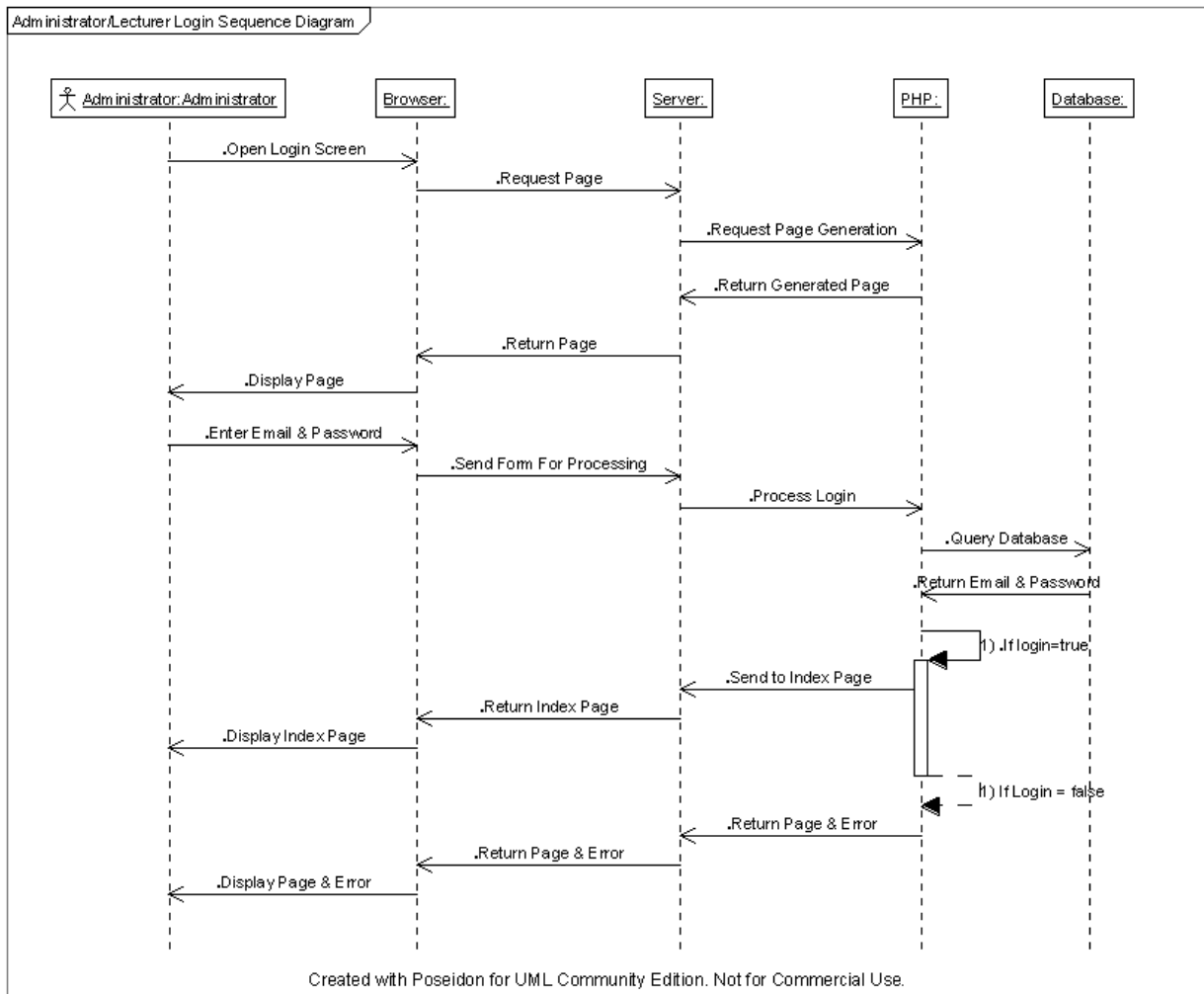
Created with Poseidon for UML Community Edition. Not for Commercial Use.

The overall interaction of the system needed to be represented as well. To do this the UML sequence diagram was used. This diagram can show the interaction between the users and the system. However it can also show the interaction between the system components and how they interact together, making it a good overall choice. This can be seen in Figure 4.2-2, which shows the UML sequence diagram for administrators or lecturers logging into the system.

UML sequence diagrams were found to be extremely useful in designing the system as they gave a clear and concise visual image on how the user will interact with the system. This allowed the implementation of the project to be much easier, and also allowed for sections of the system that are similar to be merged; which in turn helped cut down the overall system implementation time.

Further system designs can be viewed in Appendix C.

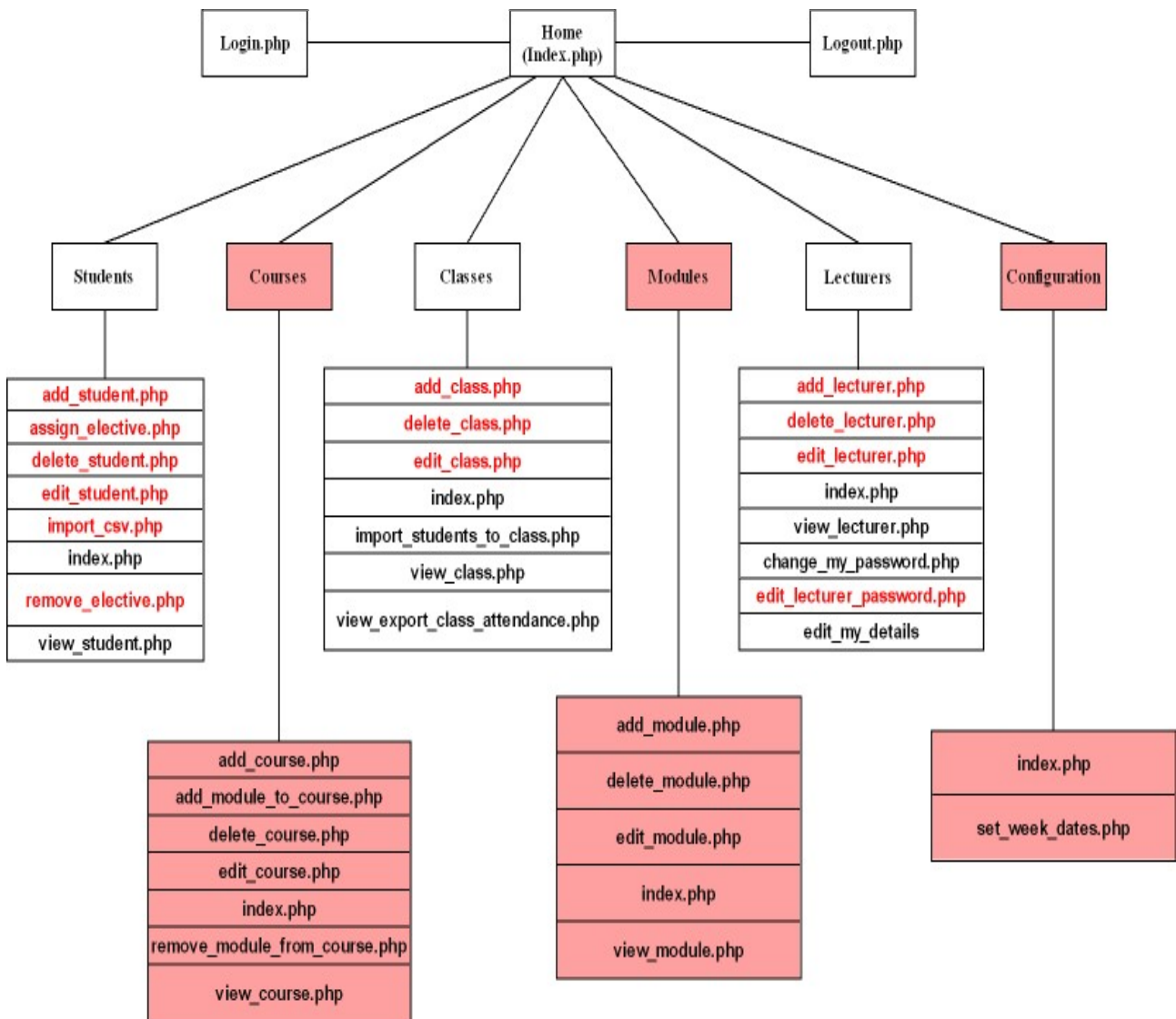
Figure 4.2-2 – Administrator/Lecturer Login Sequence Diagram



4.3 - Site Map

While the site is extremely dynamic and does not contain a single static page, it is possible to create a site map. The site map shows exactly how each page of the system links, and also the structure of the system. The sections which are marked red are parts of the site that Lecturers cannot view or access. All other parts of the system can be accessed and viewed by both Lecturers and Administrators.

Figure 4.3-1 – Site Map Diagram



4.4 - Interface Design

As stated in the requirements analysis, there are requirements that must be met by the user interface. These requirements are to provide a clear and consistent layout, provide feedback to the user if needed, minimize amount of user input required and clearly display search results.

Firstly the interface must be clear, consistent and quick to load. This will help to increase the intuitiveness of the interface, and let users to use the system with a minimal amount of training. To ensure that the interface loads as quickly as possible a minimal amount of graphics were used.

Secondly when the user completes a significant operation they will be asked whether or not they wish to continue. This process is in place to allow the user to change their mind before the operation is finally submitted.

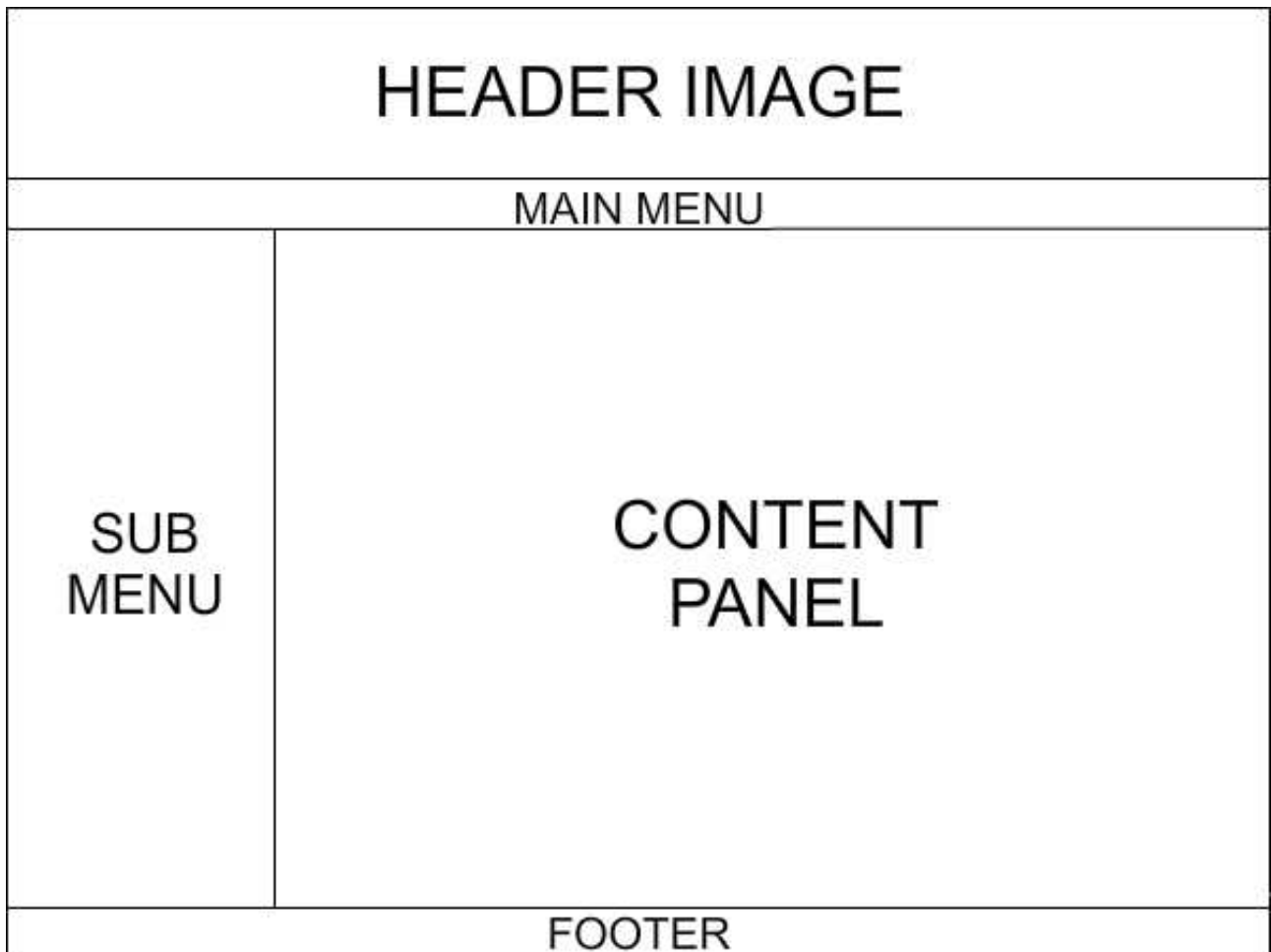
Thirdly the interface is designed to cut down on the amount of data the user has to enter. This has two key benefits, firstly it cuts down on the amount of work the user has to do, and secondly it helps in validating user inputted data.

Finally when a user performs a search, the search results must be displayed in a clear and simple way. The interface will display search results within a table, this will allow the user to quickly and easily see the returned results. When a user moves the mouse over a table row, that row will be highlighted to make it more visible.

With the requirements of the interface now clearly defined, the next stage of the design was to establish exactly what the structure and the content of each page would be.

With the system requiring three interfaces, the Administrator, Lecturer and Student interfaces. It was decided it would be best to keep these interfaces very similar. Each page will be constructed of a header image, main menu, sub menu, content panel and footer. An example of this structure can be seen in Figure 4.4-1

Figure 4.4-1 – Page Structure



The main differences between the page structures for the different interfaces are the menus. Both the main menu and the sub menu change from section to section. An example of this can be seen by comparing Figure 4.4-2 with Figure 4.4-3 . The elements found on the page also depend on what task that page is trying to accomplish. This can also be viewed also by comparing Figure 4.4-2 and Figure 4.4-3. Figure 4.4-3 shows the standard design for the Lecturer section of the interface. As you can see they are nearly identical, but the menus change accordingly.

Figure 4.4-2 - Administrator Interface



The screenshot displays the 'Attendance Monitoring' administrator interface. At the top, there is a header with the title 'Attendance Monitoring' and a clock icon. Below the header is a navigation menu with tabs for 'Home', 'Students', 'Courses', 'Modules', 'Classes', 'Lecturers', 'Configuration', and 'Logout'. The 'Courses' tab is selected. On the left side, there is a sidebar menu with the following options: 'View Course', 'Add Course', 'Edit Course', 'Delete Course', 'Add Module To Course', and 'Remove Module From Course'. The main content area is titled 'Add Course' and contains the following text: 'Please use the fields below to add a course. All fields are required.' Below this text are three input fields: 'Course ID:' (a text box), 'Name:' (a text box), and 'Duration:' (a dropdown menu with '1' selected). At the bottom of the form are two buttons: 'Add Course' and 'Reset'. The footer of the page contains the copyright notice '© 2006 Paul Mowat'.

Figure 4.4-3 – Lecturer Interface

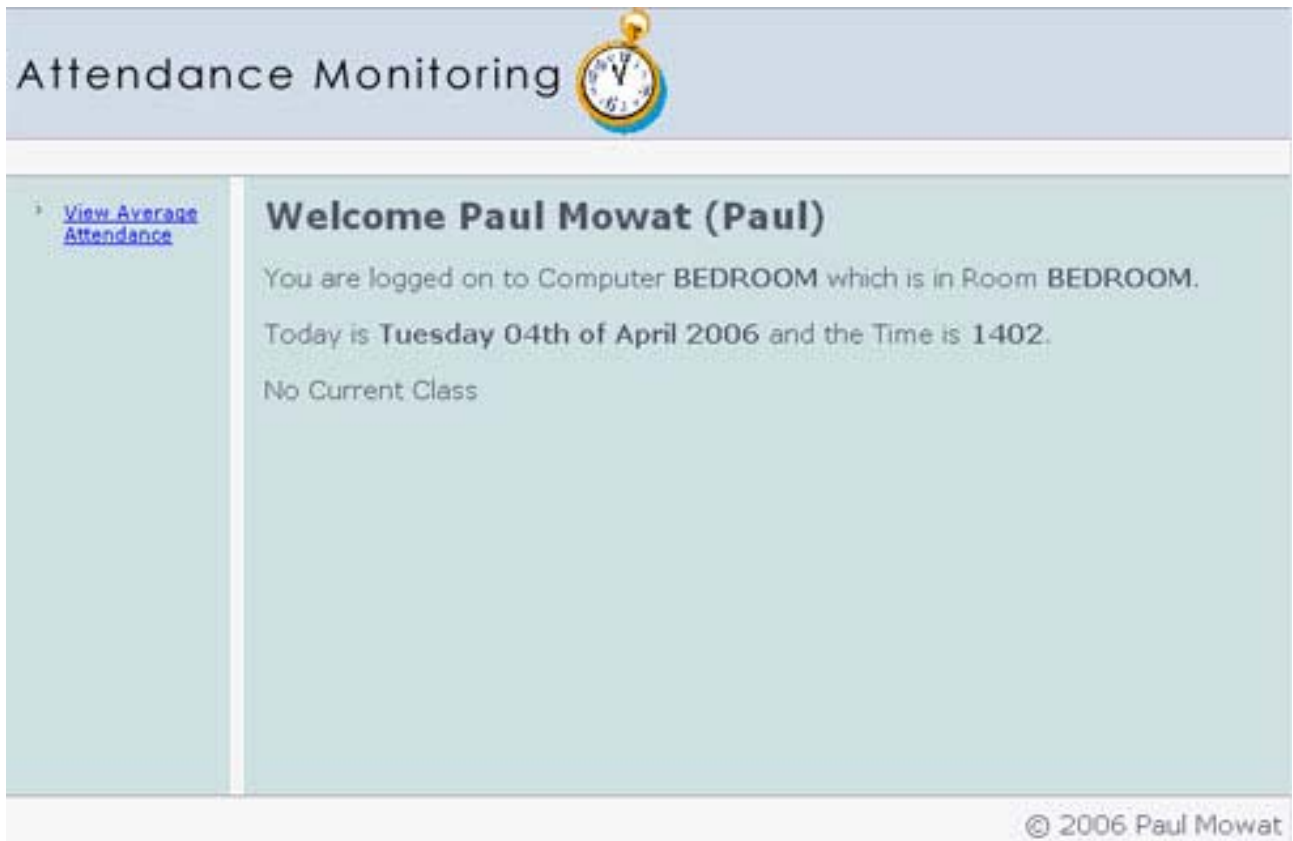


The final interface that had to be developed was the Student interface. Again it was decided to keep this interface similar to the Administrator and Lecturer ones. The main reason behind this decision was to cut down development time. The biggest difference between the Student interface and the others is that the Student interface does not contain a main menu. An example can be seen in Figure 4.4-4.


The interfaces were designed using Adobe Photoshop and used as a background for coding the interface in the implementation stage. Images of the actual system are available in the Implementation stage.

A complete set of Interface designs can be viewed in Appendix D.

Figure 4.4-4 - Student Interface



The image shows a web-based student interface for an attendance monitoring system. The header is a light blue bar with the text "Attendance Monitoring" and a pocket watch icon. The main content area is light green and contains a welcome message for "Paul Mowat (Paul)". A sidebar on the left has a link for "View Average Attendance". The footer is a light grey bar with the copyright notice "© 2006 Paul Mowat".

Attendance Monitoring 

[View Average Attendance](#)

Welcome Paul Mowat (Paul)

You are logged on to Computer BEDROOM which is in Room BEDROOM.

Today is Tuesday 04th of April 2006 and the Time is 1402.

No Current Class

© 2006 Paul Mowat

4.5 - Database Design

One of the main parts of the system is the database backend. The database backend will store all the information required by the system to operate. During the design of the database some differences in the way the database was being designed were found. These differences were due to the author being a direct entry student and having been trained to design databases differently. This was discussed with the project supervisor who approved this method of database design for the project.

Using the information gathered from the background research and the systems analysis and design it was possible to extract the necessary information required for the database. This data was then normalized to remove duplicate entries and to help make the database for efficient and robust. Once the data had been normalized there were 11 tables remaining.

Each of the tables was given a primary key to ensure every data entry has a unique identifier from all other records in the same table. The attendance table also required a unique key to be set. The unique key consisted of two fields, firstly AttendanceClassID and secondly AttendanceMatricNumber. The unique key stops any duplicate entries being inserted into the table that have an AttendanceClassID and AttendanceMatricNumber that already exist.

The table's structure and fields are listed on the next page. Figure 4.5-1 (Page 41) shows the Entity Relationship diagram for the database.

Figure 4.5-2 (Page 42) shows a table normalization sample and finally Figure 4.5-3 (Page 42) shows a sample Data Dictionary Entry.

The full database design including the Entity Relationship diagram, full table normalization and the data dictionary can be found in Appendix E.

Table: Attendance (Contains all the attendance data for students attending classes) {

AttendanceID – PK

AttendanceClassID – ID of the Class

AttendanceMatricNumber – Matriculation Number of Student Attending Class

AttendanceGroup – Class Group if Applicable

AttendanceWeek1

AttendanceWeek2

AttendanceWeek3

AttendanceWeek4

AttendanceWeek5

AttendanceWeek6

AttendanceWeek7

AttendanceWeek8

AttendanceWeek9

AttendanceWeek10

AttendanceWeek11

AttendanceWeek12

AttendanceWeek13

AttendanceWeek14

AttendanceWeek15

}

Table: Class (Contains all data about a Class) {

ClassID - PK

ClassModuleID – Module in which Class belongs to

ClassDay – Day of the week in which Class is on

ClassStartTime – Time Class starts (24h)

ClassEndTime – Time Class ends (24h)

ClassRoom – Class Room

ClassType – Type of Class e.g. Lecture, Tutorial, Laboratory

ClassStatus – Whether Class is On or Off

ClassSession – Session Class is scheduled for e.g. 2006-2007

ClassSemester – Semester in which Class will run

}

Table: Course (Contains all the data about a Course) {

CourseID - PK

CourseName – Name of Course

CourseDuration – Duration the Course Lasts For

}

Table: Module (Contains all the data about a Module) {

ModuleID - PK

ModuleName – Name of Module

ModuleRevision – Revision number of Module

}

Table: Student (Contains personal data about a Student) {

StudentsMatricNumber – PK – Matriculation Number of Student

StudentsName – Full Name of Student

StudentsEmail – Email address of Student

StudentsCourseID – ID of Course Student is undertaking

StudentsCurrentYear – Year of Course Student is currently in

StudentsGraduationYear – Year in which Student Graduates

StudentsDirectEntry – Whether or not the Student is a Direct Entry

}

Table: Lecturer (Contains personal data about a Lecturer) {

LecturerID - PK

LecturerTitle – Title of Lecturer e.g. Mr, Dr, Prof, Miss

LecturerName – Full Name of Lecturer

LecturerEmail – Email address of Lecturer (Acts as login to system)

LecturerOffice – Lecturers Office

LecturerPosition – Lecturers Current Position e.g. Lecturer, Senior Lecturer

LecturerWebsite – Lecturers website address

LecturerTelephone – Lecturers telephone number

LecturerAdminGroup – Group of lecturer, 2 = Lecturer, 3 = Admin

LecturerPassword – Lecturers password to login to the system (Encrypted)

}

Table: Class Lecturer List (Contains a list of all Lecturers currently teaching a class) {

ClassLecturerListID - PK

ClassLecturerListClassID – ID of Class which Lecturer is teaching

ClassLecturerListLecturerID – ID of Lecturer teaching the class

}

Table: Course Module List (Contains list of modules which are assigned to Course Year) {

CourseModuleListID - PK

CourseModuleListCourseID – ID of Course

CourseModuleListYear – Year of Course

CourseModuleListModuleID – ID of Module assigned to Course

}

Table: Current Modules (Contains a list of all modules a Student is currently undertaking) {

CurrentModulesID - PK

StudentsMatricNumber – Matriculation Number of Student

CurrentModulesList – Comma Separated List of Modules Student is Undertaking

}

Table: Previous Modules (Contains a list of modules a Student has previously completed) {

PreviousModulesID - PK

StudentsMatricNumber – Matriculation Number of Student

PreviousModulesList – Comma Separated List of Modules Student has completed

}

Table: Elective () {
 ElectiveID - PK
 ElectiveMatricNumber – Matriculation Number of Student
 ElectiveModuleID – Module Number of Elective
 }

Figure 4.5-1 - Entity Relationship Diagram

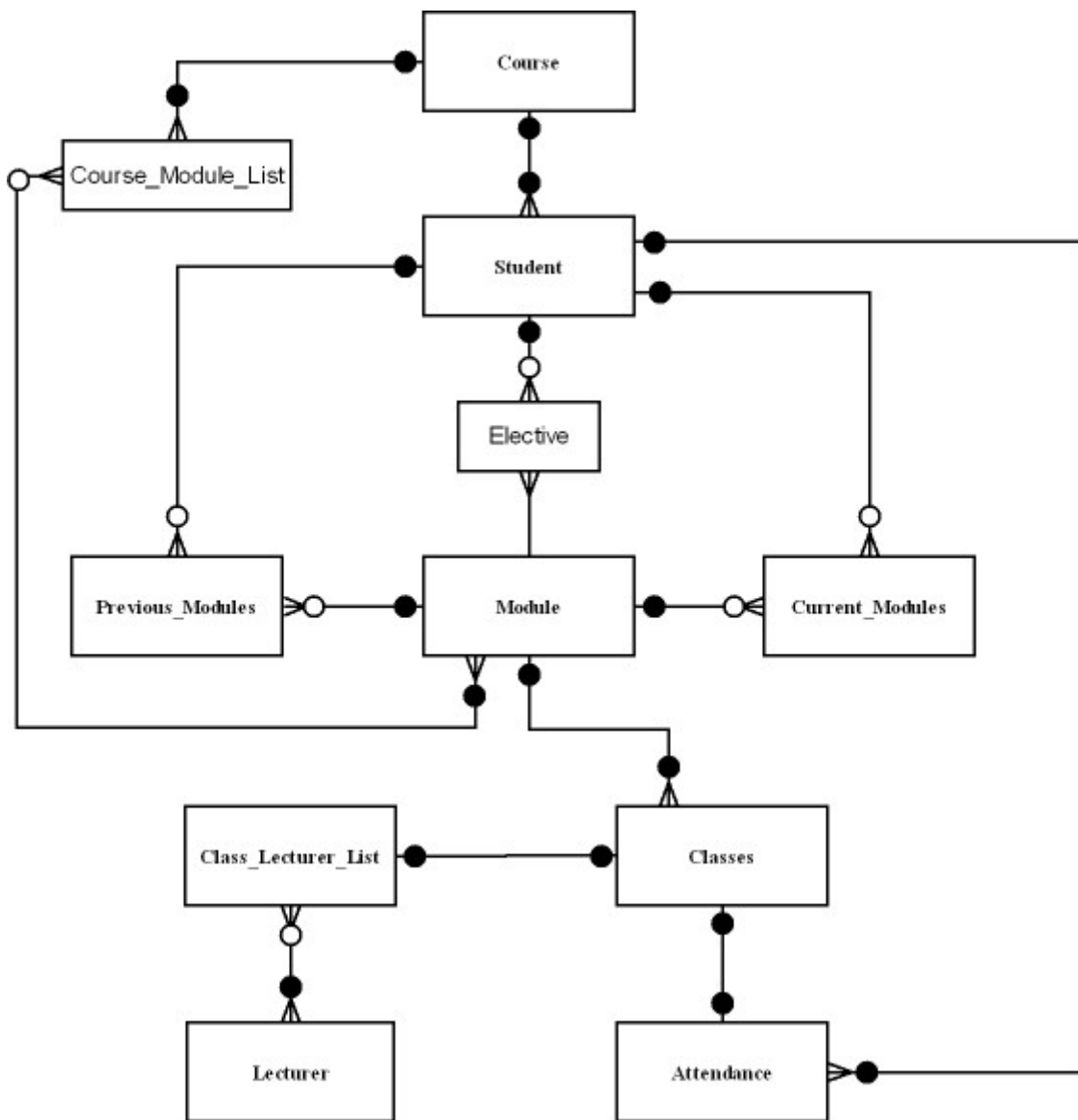


Figure 4.5-2 - Sample Table Normalization

UNF	1NF	2NF	3NF
<u>ModuleID</u>	<u>ModuleID</u>	<u>ModuleID</u>	<u>ModuleID</u>
ModuleName	ModuleName	ModuleName	ModuleName
ModuleRevision	ModuleRevision	ModuleRevision	ModuleRevision
<u>ClassID</u>	<u>ModuleID</u>	<u>ModuleID</u>	<u>ModuleID</u>
ClassDay	<u>ClassID</u>	<u>ClassID</u>	<u>ClassID</u>
ClassStartTime	ClassDay	ClassDay	ClassDay
ClassEndTime	ClassStartTime	ClassStartTime	ClassStartTime
ClassType	ClassEndTime	ClassEndTime	ClassEndTime
ClassStatus	ClassType	ClassType	ClassType
ClassSession	ClassStatus	ClassStatus	ClassStatus
ClassSemester	ClassSession	ClassSession	ClassSession
ClassLecturerID	ClassSemester	ClassSemester	ClassSemester
	ClassLecturerID	<u>ClassLecturerListID</u>	<u>ClassLecturerListID</u>
		<u>ClassID</u>	<u>ClassID</u>
		ClassLecturerID	ClassLecturerID

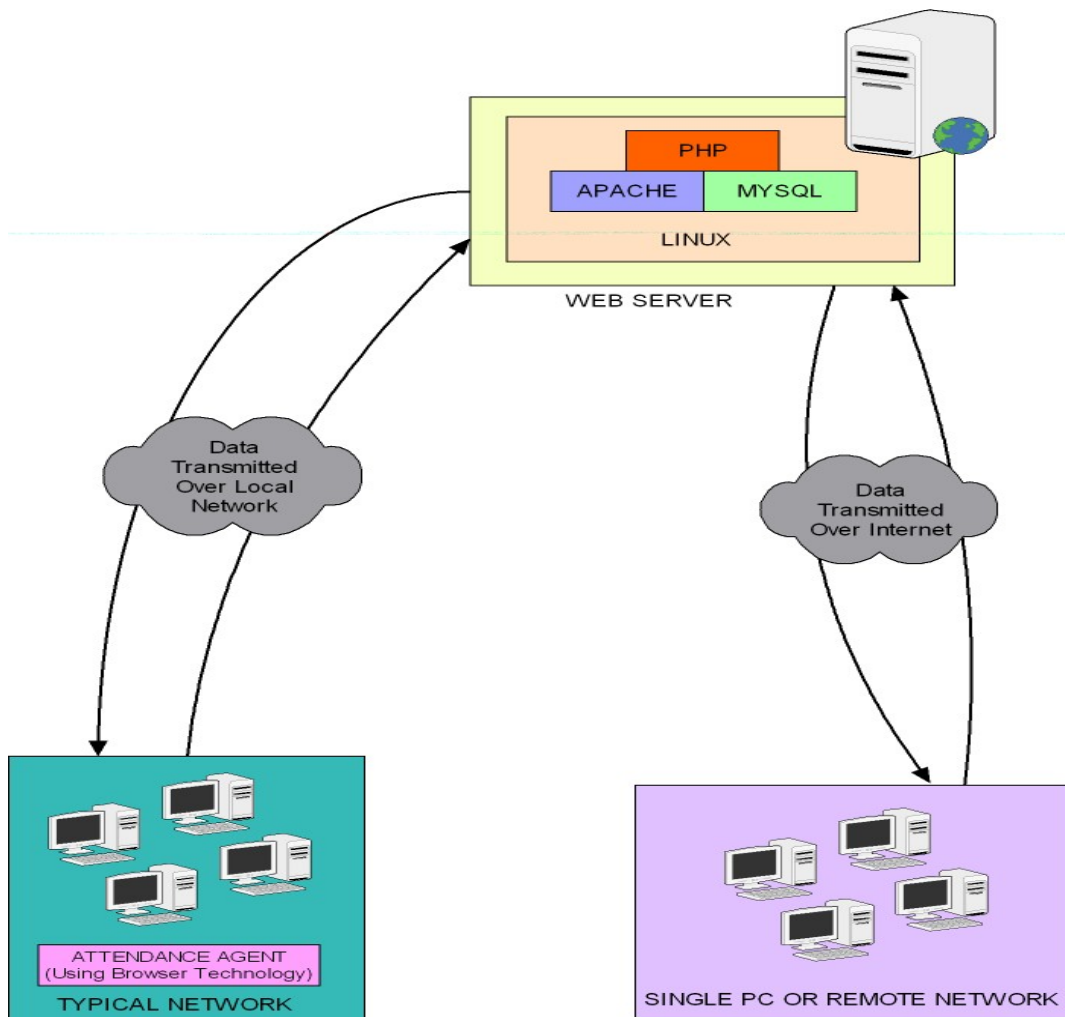
Figure 4.5-3 – Sample Data Dictionary Entry for Table Module

Attribute Name	Description	Data Type	Length	Null	Default
<u>ModuleID</u>	Primary Key	Varchar	10	No	
ModuleName		Tinytext		No	
ModuleRevision		Tinyint	2	No	1

4.6 - Server Design

For the system to work efficiently a dedicated server was required. This server had to include all of the software selected in the Research stage, which included Fedora Linux, PHP, MySQL and PHPMYADMIN. The server also needed to be connected to the School of Computing network, only then could it communicate with all of the other required computer systems. Figure 4.6-1 shows a graphical representation of the server and how it communicates with networked computer systems. Also please note that it would be possible for a Lecturer or Administrator to connect to the server over the internet, however this server is located on a local network and doesn't require this functionality.

Figure 4.6-1 – Server Design



5 - Implementation

5.1 - Implementation Overview

The implementation of the system lasted for around 11 weeks from the end of January to the End of March 2006.

The implementation started by installing the required software onto a server, which had been provided by the School of Computing IT staff. The server that was provided was a standard School of Computing system which had been completely wiped.

Before any practical implementation could be started Fedora 4 the operating system and all the other required software had to be installed. To install Fedora 4 an installation guide created by Stanton Finley (2006) was followed step by step. This guide also detailed how to install all of the other required software on Fedora 4.

With the server now in place it was possible to start implementing the system. To start with a XHTML template was developed to match the interface designs. This template was created using standards compliant XHTML and CSS code. To position elements on the XHTML page, CSS positioning was used. CSS positioning was used to layout the page instead of tables as this allowed the structure to be completely separated from the presentation/style. Therefore if a change was required to the layout it only has to be applied to the one CSS file instead of every XHTML page and the change will take effect.

Once the template had been completed the system structure was then created to match the systems sitemap as found in the design section. With the structure in place it was then possible to create all the required pages using the XHTML template.

Now with the system structure and pages now in place the next step was to create a central configuration file. This configuration file allows the systems administrator to set parameters required by the system. By using the central configuration file these parameters can then be used throughout the entire system with ease. The configuration file also acts as a loading tool for other required files.

To use the system, a database connection was required. A database connection file was created to handle this. To connect to a database some parameters need to be set in order to work correctly. The parameters that were required are the name of the database server, the user's username, the user's password and the database to be used. These parameters were set within the configuration file, and the database connection file was then loaded from within the configuration file.

To maximize coding efficiency and to reduce the possibility of bugs within the system a central function file was then created containing common functions used between multiple pages. This function file was then loaded into the configuration file above which allows it to be accessed by any page within the system.

Now with the system structure and pages now in place the next step was to incorporate a security system. Initially a basic security system coded in PHP was implemented. This system requires the user to enter their email address and password to login to the system. The user's details were then stored within a cookie on the users computer system. Each page within the system then checks to determine whether or not it exists. If the cookie exists and the details are correct the user can then use the system. However if the details are incorrect or the cookie does not exist then the user is redirected to the login screen.

With this system working successfully it was then developed further to include different user privilege levels and password encryption. The user privilege levels allow the system to differentiate between Administrators and Lecturers, and then display the appropriate options. The password encryption was required to stop the password being saved in a cookie and transmitted over the network as plain text.

With the system now secure, the development of the required sections could be started. The sections required are students, courses, modules, classes and lecturers. Each of these sections requires pages for performing core tasks such as adding, editing, deleting and viewing data from the database.

However some of these core pages needed to act slightly differently. For example the delete method within the Student section allows the user to delete multiple students at once if required. On the other hand the delete method within the Module section only allows deletion of one Module at a time.

These core pages were created for each section and then tested to ensure interaction with the database was correct. Once these pages were complete then basic communication with the database was achievable.

However additional pages were required to make the system fully functional. Firstly, the Student section requires pages that allow the user to import a CSV file containing a list of student details. It also requires pages to assign electives to students or remove electives from students. To import a CSV file using PHP is straightforward as there is a method for reading CSV files built-in. The assign and remove electives pages use the same core code as the standard add and delete pages, but required some alteration.

Secondly, the Course section requires pages to add and remove a module from a course. These pages also use the same core code as the standard add and delete pages, but required slight modification.

Thirdly, the Classes section requires pages that allow the user to import students into a class and to view or export a class's attendance details. To import students into a class the same CSV method as above was used. Once completed then the view class attendance page was developed. This page allows the user to select which class they wish to view or export. If that user is a Lecturer then they can only view classes that they teach, however if they are an Administrator then they can view any available class. Once the user has selected a class to view, they can then directly export a CSV file of that class's attendance details from the same page.

The export class attendance script extracts the attendance information from the database and writes it to a CSV file. This CSV file can then be downloaded for further use. The name of the CSV file is generated by the details of the class, such as module id, lecturer name, day of class,

Finally, the Lecturers section requires pages that allow the Administrator to change a Lecturers password. Also required are pages that allow the Lecturer to edit there own details and password, these pages are however inaccessible by an Administrator. These pages use the same core code for editing pages as the previous pages and have been modified accordingly.

Once all of these sections had been completed, they were then tested to determine whether or not they operated correctly.

Now with the Administrator and Lecturer system complete, the next stage to be completed was the Student Interface. This interface is the most complicated part of the system as it has to be embedded into a randomly generated user agent. The user agent must be able to operate on any computer system located within the School of Computing.

To develop the user agent a technology called HTML Applications (HTA) was used. HTA's allow a combination of scripts and a web page to be turned into a trusted desktop application. Only trusted applications can access the operating system. This is an essential part of the system as the user agent has to access the operating system to find out the Students username, and the name of the computer in which they are currently logged into. An example of a webpage and a webpage that has been turned into HTA can be seen below in Figure 5.1-1 and Figure 5.1-1 respectively.

Figure 5.1-1 - Webpage Before Turned Into HTA

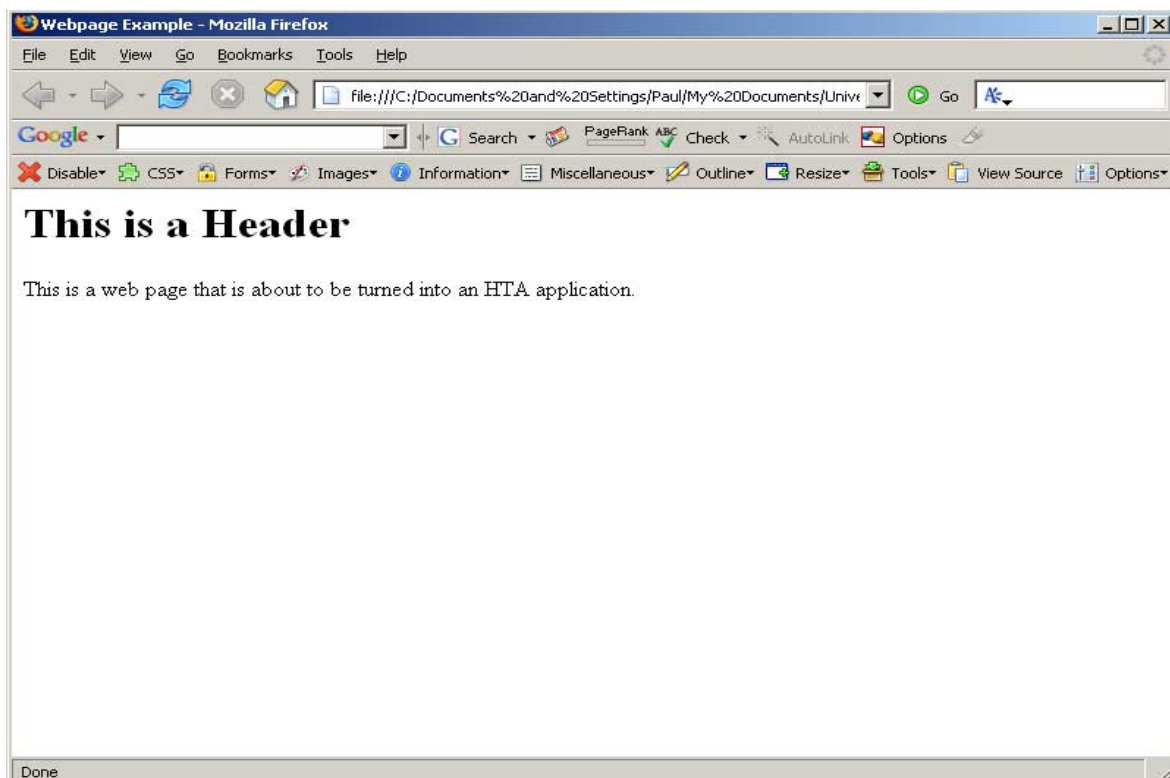
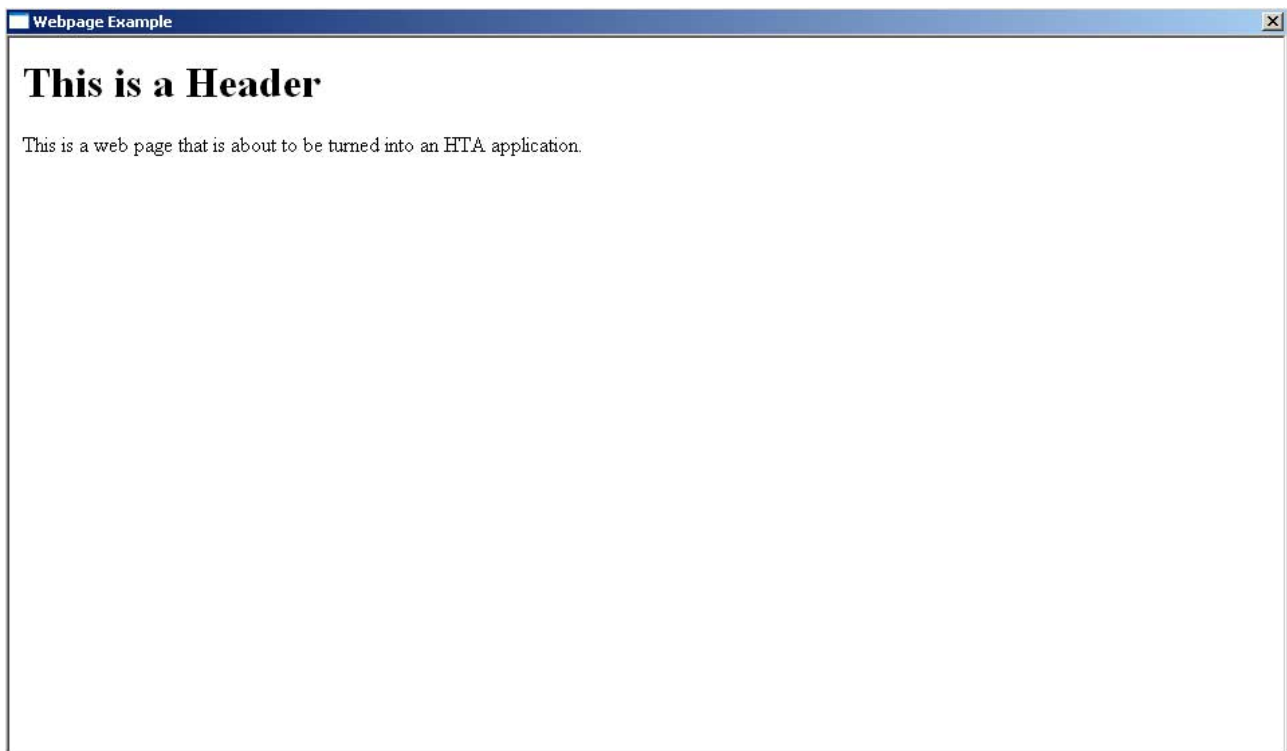


Figure 5.1-2 - Webpage After Turned Into HTA



To turn a webpage into an HTA the following code needs to be placed inside of the head tags.

```
<HTA:APPLICATION ID="oMyNNInterfaces" APPLICATIONNAME="Interfaces"  
BORDER="thin" BORDERSTYLE="normal" CAPTION="yes" CONTEXTMENU="no"  
INNERBORDER="yes" MAXIMIZEBUTTON="no"MINIMIZEBUTTON="no"  
NAVIGABLE="no" SCROLL="no" SCROLLFLAT="yes" SELECTION="yes"  
SHOWINTASKBAR="no" SINGLEINSTANCE="no" SYSMENU="yes" VERSION="1.1"  
WINDOWSTATE="normal"  
>
```

To communicate with the operating system a Visual Basic Script (VBScript) called functions.dat was created. This script contains the functions required by the HTA to access the Students username and computer. Without this script it would be impossible to acquire the required information.

The next step was to allow the Student to access different pages from within the HTA. However with the HTA basically being a static XHTML page this not possible. Therefore a separate technology called XMLHTTP (RDCPRO, 2004) was used. The technology allows data to be dynamically loaded directly into a static XHTML page using the HTTP protocol without requiring the page to be refreshed. This allowed the HTA to incorporate multiple pages as required.

With the HTA now operating correctly it had to be secured. One of the problems with using an HTA is that the source code can be viewed if the file is located. To resolve this problem a HTA generator was created using the Java programming language.

The Java application creates a directory structure to operate within. Firstly, it generates a HTA using a randomly generated file name and then extracts a ZIP file containing any resources the HTA requires. This helps make it slightly harder for the HTA to be found within the system, however the HTA still exists and can be found.

The second security step that was implemented was to execute the HTA automatically after it had been generated. With the HTA now running on the computer system, all of its required resources are stored within the computer systems memory. Once all the resources have been loaded into memory the HTA then deletes the directory structure created by the Java application using a VBScript method contained in functions.dat.

This actually causes the HTA to delete itself and all the resource files it requires. Since the HTA is stored within memory it can continue to operate successfully. This method ensures that the HTA is secured as there is no actual HTA file stored within the computer system, and therefore it is undetectable.

With the HTA generator now working correctly the next stage of the implementation was to create the log attendance script. This script is located on the server and communicates directly with the HTA. It requires the HTA to send it details of the student's matriculation number and the computer id that they are currently logged into. Once these details have been sent the log attendance script then determines the room that the student is currently in. It does this by extracting the room number from the computer id.

With the room now determined the log attendance script can determine whether or not there is a class underway at the current time within that room. If there is a class in the room, then the script checks to see if the student is a member of the class. If the student is a member of the class then the script determines which week of the semester it is and records the student's attendance. If there is no class or the student is not a member of the current class then no information is recorded.

The final part of the implementation required a script to display the class attendance average for a student against the average attendance of the class. This script also requires the student's matriculation number and the computer id in which they are logged into and communicates directly with the HTA.

To determine the student's average the system determines the current week of the semester and then retrieves all the attendance data for all classes before or equal to that

week. Once that data is available the system then performs an addition calculation, which adds all the attendance values together. See Calculation 5.1-1 for an example. This calculation allows the system to see how many times the student has attended that class.

Calculation 5.1-1 – Student Attended Class Total

Weeks of Semester Completed: 5

Weeks 1 and 3 = 1 (Attended)

Weeks 2, 4 & 5 = 0 (Not Attended)

Week1 + Week2 + Week3 + Week4 + Week5 = 2

$1 + 0 + 1 + 0 + 0 = 2$

Total: 2

Another calculation is then performed to determine the student's attendance percentage. This is done by dividing the total from Calculation 5.1-1 above from the Weeks of Semester completed and then multiplying the result by 100. This calculation returns the students attendance percentage out of 100%. See Calculation 5.1-2 for an example.

Calculation 5.1-2 – Student Average Attendance Percentage

Total from Attendance Classes / Current Week Number x 100

Example: $(2 / 5) \times 100 = 40\%$

The next step in the script is to determine the overall class average attendance percentage. This is done by selecting all the students from that class and using Calculation 5.1-1 to find out each Students attended class total, which is then added to the

overall student's total. To determine the total amount of classes that could have been attended a simple multiplication was required, the total amount of students in the class multiplied by the amount of weeks which have past in the semester. The class average can then be worked out using the formula in Calculation 5.1-3.

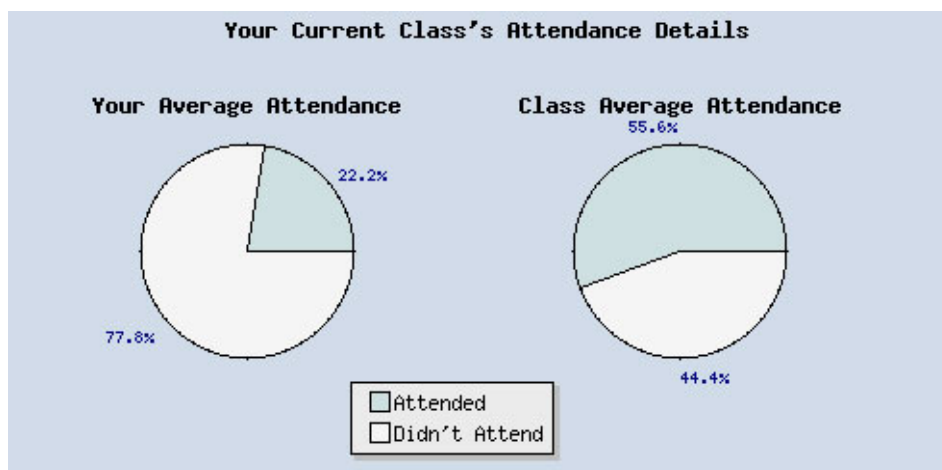
Calculation 5.1-3 – Class Average Attendance Percentage

Overall Student Total / Total Classes x 100

Example: $10 / 20 \times 100 = 50\%$

To display the student's average attendance and the class average attendance a pie chart was chosen. The JpGraph (2005) PHP chart library was chosen to provide this functionality as it provides an existing library that creates a range of different charts. An example of the created pie chart can be seen in Figure 5.1-3 below. For this PHP library to work the GD Graphics library must be installed alongside the PHP installation. This is however usually installed with PHP by default.

Figure 5.1-3 – Pie Chart



5.2 - Problem Analysis & Solutions

Over the course of the implementation stage two main problems were encountered. This section of the report will show what these problems were and how they were resolved.

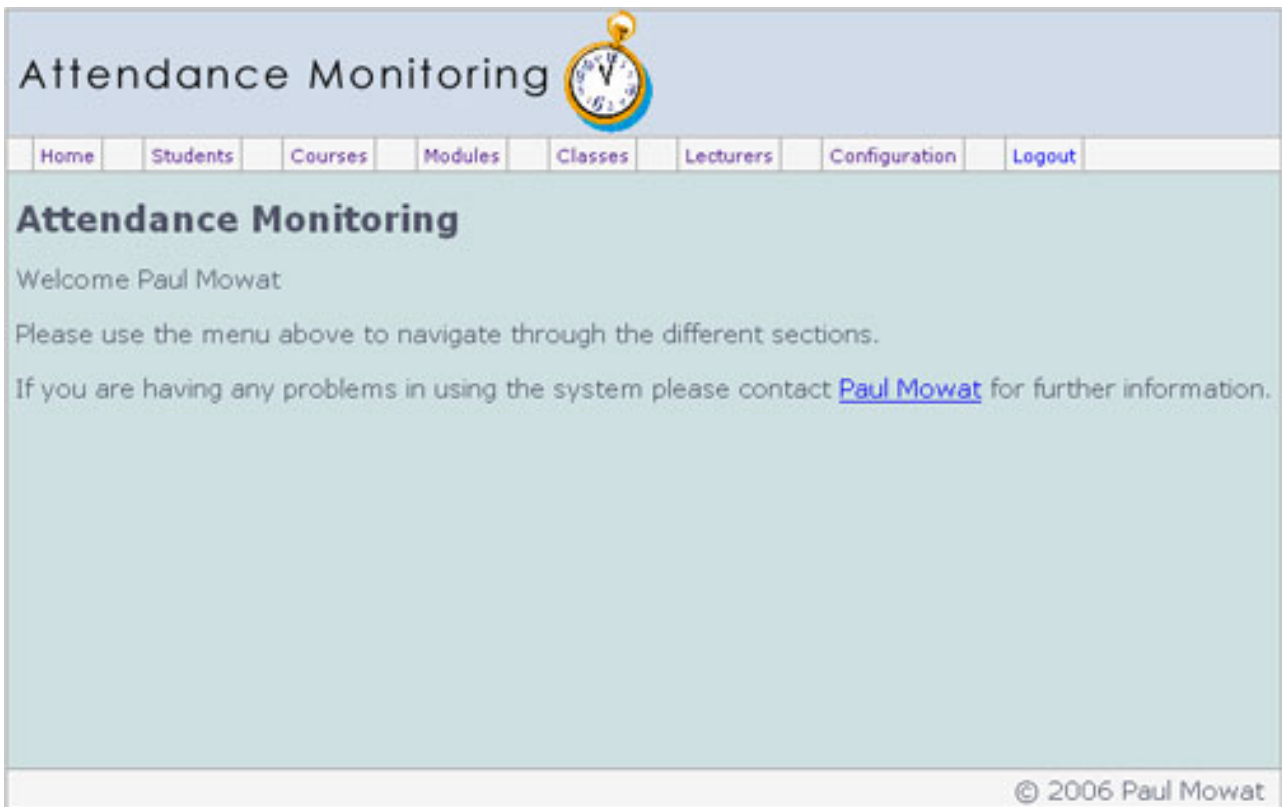
The first problems encountered were with using CSS to layout the pages. The problem was the way in which different browsers implement CSS differently. This has long been a problem of web developers for many years. A simple search to Google for the query “CSS implementation problems” returns nearly 5 million results.

During implementation the system was been tested using Mozilla Firefox and Internet Explorer. Mozilla Firefox displayed the pages perfectly, however when they were viewed using Internet Explorer the height of the page wasn't correct. This problem was having a consistent height in different browsers. This can be seen in Figure 5.2-1 and the way it should be in Figure 5.2-2.

Figure 5.2-1 - Internet Explorer



Figure 5.2-2 - Mozilla Firefox



After some further research it was found that a CSS hack was required to display this correctly in multiple browsers. The Tan Hack (Edwardson Tan, 2002.) is used to manipulate a CSS bug in Internet Explorer. This allows only Internet Explorer to recognize the code. An example of the Tan Hack can be seen in Figure 5.2-3. This example also uses another hack called the Simplified Box Model Hack (CSS DISCUSS, 2006.) to allow different values to be passed to different versions of Internet Explorer. The first value is applied to Internet Explorer 5 and the second one to Internet Explorer 6. The CSS file was then amended to include the hacks and the problem was resolved.

Figure 5.2-3 – Tan Hack

```
* html #container { width:772px;\width:770px;}
```

The second problem was how to find out which week of the semester it is. Without knowing which week of the semester it was then the attendance couldn't be recorded. The problem encountered here was due to the academic year starting in September and running for three semesters. Each of these semesters run for a different period of time, semester one and two run for 12 weeks, where as semester three runs for 15 weeks.

It was decided to solve this problem the administrator must set a date to correspond to the Monday of each semester week. It was then decided to store this information in a database table. Details of the database design can be found in Appendix

With the data stored in a database it allows the administrator to assign the dates for multiple years. It also allows an administration interface to be created to allow the Administrator modify the dates through the systems interface Part of this interface can be seen in the example in Figure 5.2-4.

Now with the interface designed and the data successfully stored in the database. The next stage was to create the script needed to find out the current week of the semester in reference to the current date. The script selects all of the records in the database for that session and then checks each one against the current date. It checks through the records until it finds a record that is greater or equal to the current date and less than the date of the next record. When it finds this record it returns that week number, and the system can then insert the attendance details into the correct week.

Figure 5.2-4 - Set Week Dates

The screenshot shows a web application interface for 'Attendance Monitoring'. At the top, there is a navigation menu with links: Home, Students, Courses, Modules, Classes, Lecturers, Configuration, and Logout. A clock icon is positioned to the right of the title. The main content area is titled 'Set Week Dates' and includes a descriptive paragraph, a note, a session selector, and two sections for setting week dates for Semester 1 and Semester 2.

Attendance Monitoring

Home Students Courses Modules Classes Lecturers Configuration Logout

> [Set Week Dates](#)

Set Week Dates

The setting of these dates are critical to the system. These dates determine which week it currently is, and therefore which week the student is attending. The dates should be set to the monday of each week

Note: This cannot be undone!

Session:

Semester 1

Week 1	<input type="text" value="1"/>	<input type="text" value="January"/>	<input type="text" value="2005"/>
Week 2	<input type="text" value="1"/>	<input type="text" value="January"/>	<input type="text" value="2005"/>
Week 3	<input type="text" value="1"/>	<input type="text" value="January"/>	<input type="text" value="2005"/>
Week 4	<input type="text" value="1"/>	<input type="text" value="January"/>	<input type="text" value="2005"/>
Week 5	<input type="text" value="1"/>	<input type="text" value="January"/>	<input type="text" value="2005"/>
Week 6	<input type="text" value="1"/>	<input type="text" value="January"/>	<input type="text" value="2005"/>
Week 7	<input type="text" value="1"/>	<input type="text" value="January"/>	<input type="text" value="2005"/>
Week 8	<input type="text" value="1"/>	<input type="text" value="January"/>	<input type="text" value="2005"/>
Week 9	<input type="text" value="1"/>	<input type="text" value="January"/>	<input type="text" value="2005"/>
Week 10	<input type="text" value="1"/>	<input type="text" value="January"/>	<input type="text" value="2005"/>
Week 11	<input type="text" value="1"/>	<input type="text" value="January"/>	<input type="text" value="2005"/>
Week 12	<input type="text" value="1"/>	<input type="text" value="January"/>	<input type="text" value="2005"/>
Week 13	<input type="text" value="1"/>	<input type="text" value="January"/>	<input type="text" value="2005"/>

Semester 2

Week 1	<input type="text" value="1"/>	<input type="text" value="January"/>	<input type="text" value="2005"/>
Week 2	<input type="text" value="1"/>	<input type="text" value="January"/>	<input type="text" value="2005"/>

5.3 - System Testing

The testing of the system was an extremely important part of the development process. There were a number of areas which had to be comprehensively tested. Firstly the functionality of the Lecturer/Administrator section had to be tested to ensure correct operation. Secondly the Student section was tested to ensure attendance was being recorded correctly and finally the site had to be tested to ensure the code validated.

The first stage of testing involved using each of the systems functions and measuring the result. This stage was undertaken to ensure that the systems functionality operated correctly. The main tests operated in this stage involved adding, editing and removing information from the database using the Administrator/Lecturer interface. Other tests involved reading and writing CSV files properly.

The second stage of testing involved the testing of the Student interface (HTA) to determine whether or not a student's attendance had been correctly recorded. Also the creation of the average attendance pie chart was tested to ensure the correct figures had been sent and the chart was generated correctly.

The final stage of testing involved validation of the XHTML code and the CSS code. The XHTML code was tested using the W3C Markup Validation Service (W3C VALIDATOR TEAM, 2006) to ensure it meets the XHTML 1.0 Strict standard. The CSS code was tested using the W3C CSS Validation Service (W3C CSS VALIDATOR TEAM, 2006) to ensure it was valid.

5.4 - Test Results

The first stage of the testing was a success as only a few errors were found. The first error was found in the view student's page. The course field wasn't being displayed as the wrong field was being extracted from the database, this was altered and the error was resolved.

The second error was found in the remove elective page. This wasn't removing the elective from the selected student. The problem was that the student's matriculation number wasn't being passed to the delete query. The code was altered to pass the matriculation number to the delete query and the error was resolved.

The third error was found in the add module to course page. The query was selecting all modules whether or not they had been added to the course. This query was changed so that only modules which had not been assigned to the course were selected, this resolved the error.

The fourth error was found in the export class attendance script. Due to the way Microsoft Excel works, it ignores leading zeros which proved a problem with matriculation numbers. To resolve this problem the data being written to the CSV file was surrounded with single quotes. This tells Microsoft Excel to represent that data as a string, this resolved the error.

The second stage of testing was a great success as no problems were found. The Student interface (HTA) handled attendance logging and also removed itself correctly. It also displayed the pie chart properly showing the student and the classes average attendance percentage.

The final stage of testing found that some of pages did not validate correctly. This was mostly due to having tags that were not supported by the XHTML Strict standard. The most common problem found was the language attribute within the script tag which is deprecated in XHTML 1.0 Strict.

When validating the CSS it successfully validated. However there were warnings about some pieces of CSS code. These pieces of code were examined however there was nothing wrong with them. After some research it was deemed that the W3C CSS Validation Service (2005) has many errors (SITEPOINT FORUMS, 2005), and cannot even validate its own CSS code (<http://jigsaw.w3.org/cssvalidator/validator?profile=css2&warning=2&uri=http://jigsaw.w3.org/css-validator/>). It was therefore decided to leave the CSS code in place as the warnings didn't affect the system.

Complete test results for stages one and two can be found in Appendix F.

5.5 - Critical Evaluation and Concussions

This project matches and expands on the original specification. Whilst there are some slight differences, it is felt that the project answers all parts of the specification well. In time available a large amount of functionality required for this project has been implemented. It is the opinion of the author that this project satisfies the given aim in creating an automated attendance monitoring system.

The Administrator and Lecturer area of system fulfils the requirement of multiple users with different privilege levels. When the system is installed onto server a single administrator account is created. With this account any amount of additional accounts can be created. This allows the system administrator to determine the privilege levels of different users.

The system also fulfils the requirement to allow Administrators and Lecturers to view or export attendance records. These attendance records can be either viewed within the system, or they can be exported into a Microsoft Excel document. This document is formatted to integrate with the current attendance monitoring system used by the School of Computing.

The system also provides a complete administration area which interacts fully with the database. This allows the Administrator or Lecturer to add, edit, delete, view or import any data if they have the required privileges to do so.

The requirement to allow the importation of a student list has also been met. A list of Students can be imported into the system directly from within the administration area. This list must be comma separated and formatted correctly. This feature allows the Administrator to enter a list of students quickly instead of having to enter them individually.

The system has also satisfied the requirement of automated entry of student attendance. The Student interface was designed to automatically record the student's attendance when they login to a computer system. The attendance however is only recorded if they log into a computer system which is located within a class that the student is a member of. This functionality allows students who are not members of the class to use any spare computer systems with no problems.

The requirement of providing attendance feedback to the student has also been fulfilled. This requirement was however expanded further by showing the student their average attendance alongside the class average attendance. Nevertheless this proved uninformative as it was not very clear simply by using text. It was therefore decided to use two pie charts as this allows the student to see the attendance averages side by side in graphical format.

Finally the requirement of simple interfaces has been satisfied. This has been achieved by using consistent layouts. By using consistent layouts the user will learn intuitively how the interface works, and be able to use the entire interface with no problems. The interfaces also use similar elements to achieve similar tasks; this also helps the interface to stay consistent.

The main problems found in the implementation of the system were how to find out the current week of the semester and the way that different browsers parse CSS code. These problems were discussed further in the Problem Analysis and Solutions chapter found on page 55.

Out of the entire system it is the author's opinion that the administration area of the system is completed to the highest standard. This area offers total control of the entire system. This allows the system to be administered by any individual on a day to day basis who has no knowledge of the programming languages used. Also by using CSS to layout the system, it is possible to change the entire look and layout of the system by simply modifying the CSS. This allows administrators with knowledge of CSS to create a unique look for the system.

6 - Reflection

6.1 - Future Developments

The project has laid the foundations for many potential projects in the future. The possibilities for the future of this project are extremely diverse. Below is a sample of six possible future developments that could be implemented into this system.

Firstly, it could be possible to add in to the system a visual floor plan. This would allow the administrator to see a plan of each floor. This plan would contain all the rooms within that floor. Each room would then contain a list of all the computer systems that are located within it. The administrator could then browse or search through the rooms and see who is logged in and which system they are logged into. This would

Secondly, another possible development could be to implement functionality that allows the administrator to set which applications can be used in any given laboratory. This would be extremely useful in assessed laboratory conditions and stop people from cheating by using the internet. It could also be used to track which applications are being used most often and to determine whether or not they are required for that laboratory. If they are not required then they could be disabled.

Thirdly, the GPAS printing system could be implemented into the system. This would allow students to view and purchase printing credits directly from within the one application.

Fourthly, a self declaration of absence system could be integrated into the system. This would allow students to declare their absence directly from within the system instead of having to do it manually. This could however cause some problems, such as students entering self declarations on days they skived off. However this could be controlled by ensuring all self declarations are before absence and by having the self declarations approved by their course tutor first.

Fifthly, the system could be expanded to include notification of poor attendee's. The notification would be sent to either the class lecturer or the course leader. This would allow them to keep track of students who could have potential problems with the course and require additional assistance.

Finally, optical character reader support could be integrated into the system. This would allow an Administrator or Lecturer to scan paper based attendance sheets directly into the system. This would save a great deal of time and minimize the possibility of human error.

6.2 - The Project: A Retrospective View

When deciding to undertake this project there were some issues that had to be considered. These issues were what would be learned? What skills will be gained? Is the project unique? Is the project challenging? Is the project achievable?

In undertaking this project it was hoped that there would be the possibility of learning new technologies and programming languages. During the course of the project many new things have been learned. These range from how to set up a server to new programming languages and technologies. The new technologies and programming languages which were learned for the system include HTML Applications (HTA), XMLHTTP and VBScript. The project also helped to expand existing skills in other languages such as SQL, PHP, CSS and Java. With this in mind the project has greatly exceeded any expectations in developing new skills.

The project was definitely worthwhile as it can help to keep a more accurate record of student attendance and hopefully identify students who require help quicker. It also opens up the chance of many future developments, some of which can be found in the previous chapter. Additionally the project is completely unique and there is no other system available today that works in a similar way. With some minor modifications this project could be used in many Universities around the world.

If given the chance of undertaking this project knowing now what I knew then. I would certainly take it on again. The project has raised my design and programming skills to a new level, and has greatly helped in improving my overall knowledge of the project lifecycle.

6.3 - Acknowledgements

Firstly I would like to thank Niccolo for agreeing to be my project supervisor and for suggesting this project. I would also like to thank him for believing I had the ability to complete the project and taking time to exchange many ideas with me

Secondly I would like to thank Marie Duncan for taking time out of her busy day to demonstrate the system currently used by the School of Computing to record and monitor attendance. Without this it would have been impossible to complete this project.

Finally I would like to thank Naomi for her understanding and patience with me during this project, and for always listening to me going on about programming, databases or other technologies she has never heard off.

Thanks everyone, I couldn't have done it without you!