UNIVERSITY OF BIRMINGHAM

School of Computer Science

Timetable Application

David Hughes
793750

BSc in Computer Science

Supervisor: Dr Mark Lee

16/04/2009

TABLE OF CONTENTS

ABSTRACT

This report details the whole process of building an application for Halesowen College. The timetable application built provides members of staff and students with a core system that is accessible through the colleges Intranet. Detail is provided on all aspects of the project including technologies used, justifications to decisions made and details of implementation. The final system allows students to access a copy of their timetable which was not previously possible. It also provides staff with a simpler alternative system to the one they currently have.

AWKNOWLEDGEMENTS

Dr Mark Lee has given valuable time and feedback throughout the course of the project. I would like to thank him for his guidance and supervision.

I would like to thank my client at Halesowen College, Jonathan Priest, for acting as an end user and providing me with the entire set of test data needed.

I would also like to thank Emily Cooper, as well as family and friends for providing me with the support I needed to complete the project.

1. INTRODUCTION

MOTIVATION

This project was chosen as a problem exists at Halesowen College involving timetable access. Students studying A-Levels at the College have timetables that are subject to change from week to week, yet they do not have a suitable way of accessing a copy of their timetable. Members of staff have access to a commercial piece of software called CELCAT. This is primarily used for scheduling by course leaders, but all members of staff have viewing access rights. The drawback to the software though is that it is not simple enough to quickly view a timetable due to all the tools it offers.

At the beginning of every term form tutors print out a copy of each of their tutees. It is then the responsibility of the student to keep it and make their own copy of it. If it is lost, which is a reoccurring issue, the student has to request to have it printed again. It is a large risk to give students access to the CELCAT system as it increases the probability of a student gaining access to information they are not allowed to see.

The primary target users will be the A-Level students, members of staff already have the CELCAT system they can access timetables through, regardless of its complexity. Members of staff are the secondary users. The Intranet is the core system that all members of Halesowen College have access to. For the author, having a wide knowledge in Java programming and previous experience using database systems, and also self taught knowledge in ColdFusion, producing an application that provides students with their timetable through the Intranet is a feasible solution.

AIMS AND OBJECTIVES

The aim of this project is to build an application that is accessible through Halesowen Colleges Intranet that displays a simple representation of a student's timetable for the current week. When completed the system should be able to be integrated into the Colleges current setup. If built correctly it should seamlessly interact with the CELCAT database and the QL database. The QL database is where all students' personal and enrolment information is stored. The application is primarily focused towards A-Level students as opposed to students studying a BTEC for example. The reason being that a timetable consisting of A-Level course lectures is subject to change from week to week. This change does not refer to a room change, although they will be shown in the timetable application, it is referring to an A-Level module having a lecture not occurring every week for the duration of the module.

The timetable system will not require any scheduling capabilities. The data required will be gathered from the database where CELCAT stores its scheduling information as and when needed. In result of this any changes made to the scheduling of lectures in CELCAT by course leaders will reflect in the timetable generated by the new application. In addition any course enrolment changes made in the QL database will also be shown.

How the timetable application is interacted with is a key aspect to its success. It must be simple and straightforward to use. To keep things as simple as possible the application must require no login from the user. To access the Intranet the user must have logged in previously, so the timetable application should use this login information. The login information required consists of only the user ID. From this it is known which information must be pulled from the database. All of the processing will be done in the background without the user's knowledge. Students will have minimal input thus reducing the errors that could occur. A search facility for staff should be available. This will provide staff with basic search capabilities such as searching by student ID, forename and surname. The system will then return matching records from the QL database. A member of staff can then view and print any of the returned student's timetables.

Producing a simple dynamic timetable can be a difficult task. It is important that the timetables generated by the application are not cluttered with irrelevant information that makes it difficult to read. It should be instantly clear where and when a lecture takes place, a user should not have to study the timetable. Any lecture clashes that occur should be shown clearly and not avoid confusion, for example overlapping the lectures. Graphical User Interface (GUI) principles must be following to ensure this is the case.

One of the problems mentioned that motivated the author to take on this project is that students only receive one printout of their timetable. Due to these students not having any access to a system they then have to request another copy to be printed. One of the main aims of the application is to address this by allowing students to print a copy of their timetable.

The timetable shown for a user should only contain lecture information for the A-Level modules that the student is enrolled upon. This requires the author to gain extensive knowledge of how the existing databases are structured. To the reader this may sound trivial but if you can appreciate the size of such databases then you will understand that the structure underneath the data can be very complex. Measures will also need to be put in place to stop a student from gaining access to another student's timetable.

The introduction of such a timetable application to Halesowen College would be a great improvement to what is currently in place. It would address the primary problem by giving students access to a system where they can get their timetable information when they want it. It would provide staff with a simpler alternative to using CELCAT, as it is using the same information but presenting it in a visually simpler way.

REPORT OVERVIEW

Background Research gives an overview of all the research that the author conducted to gain an understanding of the current system at the college, and alternatives that could be used. Research into the technologies available was conducted to which are best suited to the project.

Analysis and Specification provides an analysis of the problem that exists and a requirements definition outlining the functional and non-functional requirements. This section discusses the aims and objectives highlighted in the Introduction further.

The Design section shows the final design that was followed. The design gained more structure as how the implementation would be carried out was thought about. This section shows the design from a high level. It contains the decisions that were taken and justifications.

The Implementation section shows details of the complex parts of the timetable application. It does not show the whole implementation process, only appropriate parts that may not be intuitive.

Testing discusses the testing methods that were used throughout development. A test plan is provided in the appendices.

The Project Management section details how the writing of a project of this size was handled.

The Appraisal discusses the success of the timetable application. It details how the final build compares with the original specification. Performance and robustness issues are detailed.

The Conclusion section discusses what the project as a whole has achieved, and what the author has gained from it. Future enhancements and current limitations of the timetable application are outlined.

The Bibliography shows all of the texts that were cited throughout the report.

The Appendices are referenced to throughout the report. They contain further detail that the reader can access if he or she wishes.

2. BACKGROUND RESEARCH

This section is a collection of research into technologies and existing products that will help with the production of the timetable application. Conducting this research helped to gain an understanding of what technologies may be best suited to producing the application.

HALESOWEN COLLEGE INTRANET

When a web browser is opened within the college domain this is set as the homepage, whether it is a member of staff or student that has logged in. It is the core site that pulls together each department. Familiarity with the site grows through use due to the simple design and layout. Having the application available on the Intranet would mean staff and students would always have access to their timetable. The Intranet is accessible from outside the college's domain to users who can verify themselves with a username and password. This would mean that a student, with an Internet connection, could check their timetable from anywhere externally. It makes sense to make the timetable application accessible through the Intranet for this reason. As well as checking a timetable while at the college, a student would have the ability to check his or her timetable on the weekend for the commencing week.

EXISTING TIMETABLE SOFTWARE

CELCAT TIMETABLER

As previously mentioned Halesowen College already have a piece of software to handle their scheduling. Through further research into CELCAT *Timetabler* it has been found that its main aim is to schedule courses and classes, manage room bookings and prepare timetables (CELCAT, 2008). This software is well suited to doing just that, but when members of staff attempt to simply view a timetable, all the complex features it has to offer can slow the process down. This is true for staff but students do not have a way to access the CELCAT *Timetabler* system at all. SQL Server is the physical location of the SQL Databases containing CELCAT timetables created by staff. This is an advantage as it allows third party software to use the data stored on SQL Server. Research into SQL Server is discussed below under the section database technologies.

IMPERIAL COLLEGE LONDON

The Imperial College in London have taken advantage of the fact that third party software can be written and use the SQL database. An online application has been created, for use at the college, which carries out what the proposed application in this project will. *MyTimetable (student)* (London, 2007) is an online interface that allows a student to log in and gain access to their weekly timetables for the academic year. Only a brief description is available, no detail into its functionality or how it is implemented is provided. This verifies that a solution can be created, but to get it working in conjunction with Halesowen Colleges current systems may be problematic. Imperial College London also describe a similar online applciation for staff but it allows staff access to their own timetables as opposed to students timetables. The proposed timetable application will give staff access to students timetables.

DATABASE TECHNOLOGIES

Analysis into Halesowen College's current setup reveals that two databases exist that the timetable application will need to be able to access if the systems are to seamlessly integrate. These being the CELCAT *Timetabler* database and the QL database. Both of which are stored on SQL server 2005. The CELCAT *Timetabler* database holds every piece of information about events. An event is something at the College that must be timetabled for a student. These events consist of, most commonly, lectures, tutorials or workshops. The database holds the relevant information that needs to be shown on the

timetable such as start time and room number. QL is a system supplied by a company called Agresso. Agresso systems (2006) have described their QL system as a modern student management application within an integrated database environment. The database environment system being SQL Server 2005. The overview paper provided by Agresso contains a diagram showing all the functionality of the system. The functionality the proposed timetable application will need access to is the student tracking. The information stored there that the timetable application will require consists of students' names, ID's and enrolment information.

Two approaches of storing the data needed to generate a timetable exist. Either the author must gain an understanding of how the two currently used databases are structured, and use SQL to extract the required data dynamically. Dynamically is used to mean only gathering data applicable to the student whose timetable has been requested. The second approach is to request Halesowen College to export the relevant information from both databases into one, allowing the author to manipulate the data into what he feels is the most efficient structure.

POSTGRESQL

As the author has no previous experience and knowledge in using SQL Server the second option is considered first. The database system that author has substantial knowledge in is PostgreSQL. This is an open source, object-relational database management system (Holzner, 2008). It is also considered a strong alternative to SQL Server as it contains features that are only usually found in commercial products (Worsley & Drake, 2002). This approach could be met by putting forward the required database information to database administrator at Halesowen College, and request him to extract this data. It can then be imported into PostgreSQL. This would allow the author to easily manipulate the data structure, but a major problem arises, the problem of data duplication. Having exported the data into a PostgreSQL database means it exists in two places. This issue is worsened by the fact that both the QL and CELCAT *Timetabler* databases are constantly subject to updates and inserts. This is backed up by Connolly & Begg (2004), who state that, "an associated problem with duplication is the additional time that is required to maintain consistency automatically every time a tuple is inserted, updated or deleted." Introducing a problem like this into the College's system would create further work, by needing to implement a way to synchronise the databases with duplicated data. For example if a room change occurs in the original CELCAT database it will not be reflected in the PostgreSQL copy unless a synchronisation method is implemented.

MICROSOFT SQL SERVER 2005

The duplication issue that arises with above does not occur if the first approach is followed. It will not result in any duplication as the databases will continue to be used unaffected as they currently do. The completed timetable application will only require read access to the database. This approach introduces the requirement of the author gaining an understanding of SQL Server 2005. SQL Server, like PostgreSQL, is a relational database management system (RDMS). Although using SQL Server has been forced in this project, research has highlighted that SQL Server uses GUI tools to administer databases, which will assist when the author is learning how to use it, as no previous experience is held.

Sticking with SQL Server will also ensure the timetable application dynamically creates a live student timetable. Updates to events in the CELCAT and QL databases will instantly be available when a student or member of staff requests a timetable. This would not be the case in the second approach considered, as the PostgreSQL database would firstly need to be updated to reflect the changes to CELCAT *Timetabler* and QL.

WEB TECHNOLOGIES

MACROMEDIA COLDFUSION MX

It has been discussed that the best place to make the timetable application available is on the college Intranet. The college use ColdFusion to build their web pages. Forta & Weiss (2003) state that ColdFusion is an application server. An application server is the system software that underlies the server-based execution of shared business applications (Microsoft, 2006). ColdFusion's application server was actually the first available in 1995 (Forta & Weiss, 2003). An application server makes applications, usually dynamic ones, available to multiple users, just like a file server delivers files to many users. A ColdFusion application refers to a website constructed of ColdFusion pages, which are processed by the application server.

A Web Server typically returns the contents of a webpage back to the client upon request. The ColdFusion application server parses the page and looks for ColdFusion tags, which instruct the application server to carry out certain functions, such as, execute a query and produce a table containing the results. Any plain HTML read is left alone. When all the ColdFusion tags have been executed the resulting page will consist of plain HTML, dynamically created from the ColdFusion application server, this HTML is returned to the Web Server. ColdFusion contains its own Web Server so it will not be necessary to use another.

As mentioned above, the application server looks for ColdFusion tags within a HTML page. These ColdFusion tags are the second technology ColdFusion provides, the first being the application server itself. ColdFusion tags, formally known as ColdFusion Mark-up Language (CFML) extend HTML. Forta & Weiss (2003) compiled the following list that shows some of the capabilities the CFML offer.

- Read data from, and update data to, databases and tables
- Create dynamic data-driven pages
- Perform conditional processing
- Populate forms with live data
- Process form submissions
- Generate and retrieve email messages
- Interact with local files
- Perform HTTP and FTP operations
- Perform credit-card verification and authorization
- Read and write client-side cookies

The reader should be able to tell from the list that ColdFusion enables programming style functions to be written within HTML web pages. The author has not used ColdFusion previously so if its use is specified as a requirement a better knowledge of how to build applications with it will need to be gained.

PHP

PHP is a powerful server-side scripting language for creating dynamic websites, created in 1994 by Rasmus Lerdorf (Holzner, 2008). An advantage to using PHP over other alternative technologies is that it is cross-platform friendly (Lerdor, Tatroe, & MacIntyre, 2006). PHP code is most commonly embedded within HTML. Its syntax is largely based on programming languages such as C, Java and Perl. The latest release, version 5, has more of an object-oriented programming approach which can be difficult concept to grasp, especially to a user with limited programming experience. The goal of PHP, as stated on its website, "…is to allow web developers to write dynamically generated pages quickly" (PHP, 2009). This is also what ColdFusion allows developers to accomplish. Like ColdFusion processes the CFML tags within the application server, PHP runs on a Web Server, taking PHP code as an input and producing HTML web pages as an output. PHP differentiates itself from ColdFusion by being freely available. This is one of the reasons why it is much more widely used in comparison to ColdFusion.

DISPLAYING THE TIMETABLE

Research into web technologies has shown that it is possible to make the timetable application available on the college Intranet. This section shows the research into what technology could be used to show the visual timetable.

HTML

One option is to display the timetable in pure HTML. The way this would be achieved is through the use of HTML tables. Tables are created in HTML using the `<table>` and `</table>` tags. A table is built up of rows defined by `<tr>` and `</tr>`, within these rows are cells, better described as columns, created using `<td>` and `</td>`. The data to be shown in a table is put into the `<td>` tags, td standing for table data. These cells can contain text, images, lists, paragraphs, forms, horizontal rules, nested tables, etc.

To get a HTML table to generate a timetable would be a simple task. To get it to generate an accurate timetable though is very difficult. Each day on the timetable could have one row. Then every event within that day would be displayed in a table cell. This would be sufficient if the events at Halesowen College occurred at a consistent time throughout the day, such as hourly. Six cells could then be created to represent a six hour day beginning at 9am, if a lecture happens at 1pm on Monday, the details can be inserted into the fourth cell across in the first row. This is not the case. Events need to be shown on the generated timetable accurate to the minute, as a lecture could begin at quarter past 12 and last half an hour, and the next day a lecture could begin at quarter past 12 and last an hour. This cannot be done in a HTML table. To achieve an accurate timetable in a HTML table, cells need to be placed accurately. The only positioning that can be carried out on a cell is defining its width and positions of the contents within itself. To generate an accurate timetable cells would need to be created for every minute. In a six hour day this would be 360 cells, which is difficult to manage, a five day week results in 1800 cells. This becomes impossible to efficiently manage. Apart from being impossible to display any information in this cell it is not guaranteed that the timetable will appear the same on a different web browser.

When constructing a web page it is very difficult to get a web page that works and appears the same in all web browsers. This is due to the way web browsers interpret HTML differently. A web developer that has built and tested a web page only using Microsoft's Internet Explorer cannot expect the page to look the same on another browser. Creating the table in HTML using will require different implementations for different browser compatibly.

Halesowen College use Microsoft Internet Explorer as the main browser, Mozilla Firefox is the second most used throughout the college and is growing. The college also has a select number of Apple Macs which use Safari. The amount of Macs is also growing in numbers.

JAVA APPLET

HTML is primarily used to layout the contents of a webpage. It is used to layout elements such as bulleted lists, tables, graphics and links. Another element that can be included in a HTML page is a Java Applet. An applet is a type of java application that is downloaded and run on the client machine. Only Java enabled browsers can access an applet, which is a problem if the target audience is widespread. This is because it cannot be guaranteed that everyone will have Java enabled. Halesowen College have the Java Plug-in installed so it will be acceptable to use a Java applet within their Intranet site.

Embedding an applet is a simple procedure but getting it to work correctly in practice can be difficult. HTML uses tags to manage elements, for example `<title>` is used to define the title of a web page. `</title>` is used to denote the end of the title. Similarly `<applet>` and `</applet>` tags are used to embed an applet. The tag must include as attributes, where to locate the class files and how to position the applet on the page.

Cornell & Horstmann (2003) state that some of the most successful Java programs are corporate Intranet applications that interface with corpoate information systems. These programs being relatively small, need to interface with databases and are built specfic to a business. The applet mechanism is considered an excellent delivery technology for these programs (Cornell & Horstmann, 2003). The final timetable application will be specific to Halesowen College as it will generate a timetable based on the information stored on their databases, the Intranet is the ideal delivery medium for the application.

A weakness of Java used to be its graphics capabilities (Knudsen, 1999). The introduction of the Java Foundation Classes (JFC) eliminated this weakness. The JFC are a suite of libraries that provide a graphical framework for building portable Java-based Graphical User Interfaces. One of these libraries being Java2D which provides classes for implementing various painting styles, complex shapes, fonts and colours (Eckstein, Loy, & Wood, 1998). The Java2D library could be used to generate a timetable with an Applet as its container within a web page. This would allow an extremely presise timetable to be generated as the Java2D library allows shapes to be drawn to pixel precise co-ordinates. Lines can be drawn at hourly intervals, then for a lecture that starts at quarter past 9, the correct pixel can be calculated for the the start position by splitting the hour space into 60, to represent 60 minutes, and a rectangle can be drawn beginning at the exact position. The length can also be calculated in relation to the width of the timetable. Having the flexibilty of drawing shapes at any position gives the abilty to present a dynamic timetable that not only shows the events that occur, but shows the events exactly where they happen.

Sun Microsystems declared Java as being a language that produces ‘Write Once, Run Anywhere’ applications (Cornell & Horstmann, 2003). As already mentioned above under the heading HTML, Halesowen College use three different browsers. If the timetable is generated in HTML it cannot be guaranteed that it will appear the same on all browsers. Following Suns slogan, writing one Java Applet that draws a timetable using Java2D, it will be viewable from any of the browsers that are Java enabled. It is not limited to browsers either, in theory any device that is Java enabled would be able to display the applet written, such as a mobile phone.

ANALYSIS AND SPECIFICATION

PROBLEM ANALYSIS

Conducting background research into the technologies available and methods of producing the proposed application has given the author confidence that the initial aims can still be met. This section discusses the problems further and recaps the aims with the background research kept in mind. A detailed requirements definition has been produced to give a strict guideline to follow throughout development.

As mentioned in the introduction members of staff use a commercial product called CELCAT *Timetabler* to schedule the timetables. Only course leaders have the access rights to manipulate a timetable, all other staff have viewing and print access rights only. Students do not have access to the CELCAT *Timetabler* system at all. At the beginning of every term a tutor groups form tutor prints out a copy of each student's timetable from the CELCAT *Timetabler* system. If this is lost then a student has to request a member of staff to print another. CELCAT *Timetabler* is quite a detailed system. Most staff members will never use it to its full potential as they only require it for viewing and printing timetables. As CELCAT *Timetabler* is used to schedule the timetables it is safe to assume that no lecture clashes will be entered into the system. With this assumption, the aim of showing clashes in a suitable way will not need to be met.

The problem with the CELCAT *Timetabler* is that viewing student's timetables is quite a tedious task when carried out multiple times. The system has many fine details displayed when viewing a timetable, for example, if a timetable for a particular student is shown, it defaults to show all lectures for every week at the college. The more common case of searching for a student's timetable is when a member of staff or student wants the timetable for the current week. This requires selecting only the weeks to be shown. It may seem like a minute problem to the reader but to a member of staff at Halesowen College it can become tedious having to repeat this for every timetable viewed.

The average student studying for A-Levels is enrolled on four different A-Level courses, each being run by different course leaders, of whom may not have everyday contact with each other. When a course leader makes changes to the timetable for a particular week the students are informed by word of mouth. It is unlikely that students will remember a room change among the workload they have to handle. It is also difficult to track a tutor group as a whole as it is a possibility that each student is enrolled on different sets of A-Levels.

Having a timetable application available to all members of the college through the Intranet will provide a solution to students having to request copies of their timetable. As discovered through research the Halesowen College Intranet page is accessible from outside the college domain. Using the Intranet as the medium for accessing the timetable application as opposed to a standalone application will mean students and staff can access a timetable from anywhere. The applications use will not be restricted internally to the college domain. This may result in students not needing a printed copy of their timetable.

The QL and CELCAT *Timetabler* databases have been discovered to be stored within SQL Server. This enables the author to develop the application to be integrated into the current database setup by writing SQL queries to gather the relevant data. The database administrator will need to give the application SELECT permissions on the two databases, but this is not required until the system is fully tested and working. Due to time scale available the application may not be tested live at the college, but it can be carried if the project is completed with remaining time. Development will take place on offline versions of these two timetables, the offline environment will be set up exactly the same as the live environment; the reasons for this are discussed in the design section under Student Privacy and Data Security during development.

A problem that members of staff have with CELCAT *Timetabler* is that its complexity can often get in the way. The main problem being that when a timetable is opened the whole academic year is shown. The modules that are shown do not necessarily happen every week. It becomes difficult to differentiate which happen on which week. It is possible to show only those on the week it is viewed but this has to be selected for every new timetable displayed. The proposed timetable application will only show lectures that occur on the current week as default.

It has been decided that using a Java applet is the best suited way to display the timetable. This is due to its Java2D package providing the author with the tools to write an applet that generates a pixel precise timetable as well as being appealing and simple to look at.

One feature that Dr Mark Lee suggested implementing is the soundex algorithm. The soundex algorithm was devised to code names recorded in US census records (Black, 2007). This is an algorithm that codes names phonetically. The codes can then be compared to find phonetically equal names. Details of the algorithm can be found in the implementation section. This algorithm can be used in the staff search facility, if a member of staff searches for a student with the surname "Boswell". The search should also return "Buswell" and "Baswol". This will come in extremely useful for members of staff that know a students name but not the spelling, for example with names such as "Ann" and "Leigh".

Further discussions with Dr Mark Lee highlighted a nice feature. As the timetables generated from week to week change, some days will not contain any lectures such as bank holidays. It would be helpful to the user to display somewhere on the timetable that it is a bank holiday, if they view the timetable on that particular week, for example Good Friday.

REQUIREMENTS DEFINITION

The following requirements definition has been put together following the structure outlined in Sommerville's book (Software Engineering, 2004). The requirements definition consists of functional requirements, which detail the services the system should provide, and non-functional requirements, which are constraints that are put onto the systems services or functions.

Functional Requirements

1. The application must generate one timetable showing all A-Levels taken for a given student ID

2. The application shall only show the lectures for the current week
   2.1. The current week refers to Monday – Friday of the week the timetable is viewed.
   2.2. If the timetable is viewed Saturday or Sunday the next weeks lectures will be shown

3. The application must be accessible from the Halesowen College Intranet page
   3.1. It will be accessible via a link
   3.2. When clicked the relevant page will be opened depending on the user logged onto the machine. See requirement 5
   3.3. If a staff member clicks the link it will open the staff search facility
   3.4. If a student clicks the link it will show their timetable

4. The timetable application must have a staff search facility
   4.1. Should contain a search feature to allow staff to search by:
      - Student ID
      - Forename
      - Surname
   4.2. Should contain a sort order option to order the results by any of the fields in requirement 5.
      - Ascending
      - Descending
   4.3. Should return a list of all matching students

4.4. Phonetically matching student names should also be returned when a search is conducted
4.5. A member of staff must be able to view the timetables of the returned students.

5. The application shall not require a user to log in
   5.1. The application should automatically detect the user

6. Students should not be able to gain access to any other timetable but their own

7. Staff should be able to view all students timetables

8. The application shall have a 'you are here' feature
   8.1. Red line showing which timeslot should currently be attended
   8.2. Red line should move across the timetable throughout the day
   8.3. Should only appear on the days and times specified in requirement 2.1

9. The application must allow timetables to be printed
   9.1. Students can print a copy of their own timetable
   9.2. Staff can print a copy of any A-Level students' timetable
   9.3. The printed version should not show the red line in requirement 8

10. The application must be resizable
    10.1. The application must resize the timetable depending on the size of the window

11. To provide a timetable application that can be used on many screen resolutions

Non-Functional Requirements

Usability

12. To provide a simple, easy to use interface for the display of a students timetable
    12.1. Any user of the timetable application should be familiar with how to use it within 10minutes.
    12.2. It should not require any guidance

Efficiency

13. To provide a timetable within 5 seconds of a request
14. 95% of users should be able to use the application with no more than a 10 second response time

Reliability

15. The application must be available for use 95% of the time during the working day

Portability

16. The timetable application must operate on the browsers outlined in appendix B

Standards

There are no requirements to meet standards

Interoperability

17. The application must be able to work with Halesowen Colleges current system

Legislative

18. Only staff can view any students timetable

Delivery

19. The application must be completed by 23rd March 2009

Implementation

20. Any enhancements to the application after implementation should not invalidate requirement 15

Ethical

There are no ethical requirements for the timetable application

3. DESIGN

SYSTEM ARCHITECTURE

Data driven Internet applications can be shown in terms of layered components. A graphical representation of the system architecture for this project has been produced, see figure1. These layered components are abstraction layers. The physical structure will not necessarily consist of three separate components. The timetable application will follow the three tier architecture. In the three tier architecture the user interface (presentation), functional process logic and data storage are shown as three separate layers. These layers can be developed and maintained as independent modules, most often on separate platforms. This allows each of the layers to be updated without any effects on the other layers. For example changing the user interface would not have any effect on the ColdFusion server.

The presentation layer is the graphical user interface. This will be the applet that generates the timetable and the HTML pages that provide staff with a search facility through the use of form submissions. The logic layer refers to the ColdFusion application server. This is where the HTML pages are dynamically created from the CFML using the Result Sets retrieved through data access. The data layer is the data storage layer. This refers to the Microsoft SQL Server 2005 DBMS. This is where the CELCAT and QL databases are stored.

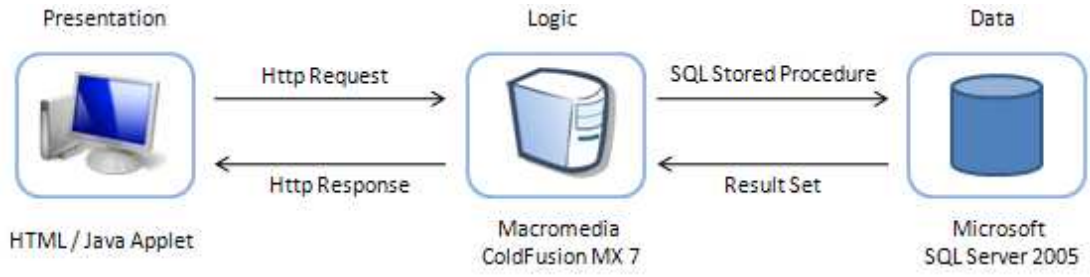


Figure 1: 3-tier system architecture

The two tier architecture could have been employed; this is when the structure consists of a server and clients. The logic is either client side or server side. Traditionally the client implements just the graphical user interface, and the server implements both the business logic and the data management (Ramakrishnan & Gehrke, 2002). The disadvantage of this is there is no abstraction layer dedicated to logic. It is either handled on the client or server. Having the extra abstraction layer allows heterogeneous systems to exist (Ramakrishnan & Gehrke, 2002). Each layer can make use of different software components, allowing each to be updated without effect on the other layers, as previously discussed. The middle logic layer allows access to multiple data sources in the data layer, which is relevant to the timetable application, as there are two databases that need to be accessed. ColdFusion in the middle layer can centrally control the database connections to the CELCAT and QL databases.

DATA LAYER

From now on the design section is structured into three main sections. One section for each of the three layers in the architecture.

DATABASE DESIGN

As discussed in the section titled Background Research, Halesowen College have two databases that contain the data that the timetable application will require. Research has shown that exporting this

data into a separate database will prove to be of no advantage to the functionality of the application. The CELCAT and QL databases will be left unaltered, and no other copies will be made, only offline versions for development discussed below. The proposed timetable application will need to connect to the databases and gather the required data. This requires to author to gain an in-depth understanding of how the databases are structured, and to develop the most efficient way of gathering the data, which requires skills in SQL Server 2005.

STUDENT PRIVACY AND DATA SECURITY DURING DEVELOPMENT

During development the probability of code making undesirable changes to the data is increased significantly. This is why an offline version of the two databases will be created for use during development. This means that if any changes do get made to the data it will not interfere with any other systems. It is illogical to develop a new system against the actual database that is to be used, especially when the database contains sensitive information and other systems rely on it. An example of an undesirable change would be an incorrectly written SQL query deleting numerous records from a table. If this occurred on the CELCAT database then thousands of events could be lost, and since many staff throughout Halesowen College use the CELCAT system everyday for scheduling timetables, their tasks would not be achievable. Once the author has gained an understanding of the databases a request can be put to Halesowen College for the tables required. The database administrators will need to export the data so it can be imported into the SQL Server installed on the author's machine. Another reason for the request being made is that sensitive data exists in the QL database. The database administrator will need to edit all of the data so that the author, or anyone viewing it, cannot trace back to a student. The only identifiable data that is required by the timetable application is a students ID, name and their course enrolments. Anyone reading the data should not be able to identify where a student is at any time, and it is for this reason the administrator will edit the data. Only he will know which records in the offline version correspond to the real data. The CELCAT data can be exported as it is, because no sensitive data is stored. It is a large database holding events with their corresponding locations and times. The student data used in development will be edited so the real events occurring will not be showing an actual students timetable.

For this project the author has been supplied with the database data, so the initial stages of database design is unnecessary. The physical design of the databases is shown in the following two sections to show the reader the databases structures. The reader must keep in mind that the databases do not contain just the following tables. The author is only showing the tables and fields within them that are required by the timetable application.

CELCAT *TIMETABLER* DATABASE

The CELCAT Timetabler database consists of 133 tables in total. Altogether the database is just over 400mb in size. The reader should appreciate that this is a vast amount of data. The author has extensively studied the database structure to locate which tables the timetable application will need to access; Figure 2 below shows these tables, the data fields they hold and the relationships between them.

Fields with a key icon next to them in Figures 2 and 3 indicate Primary Keys in the tables.

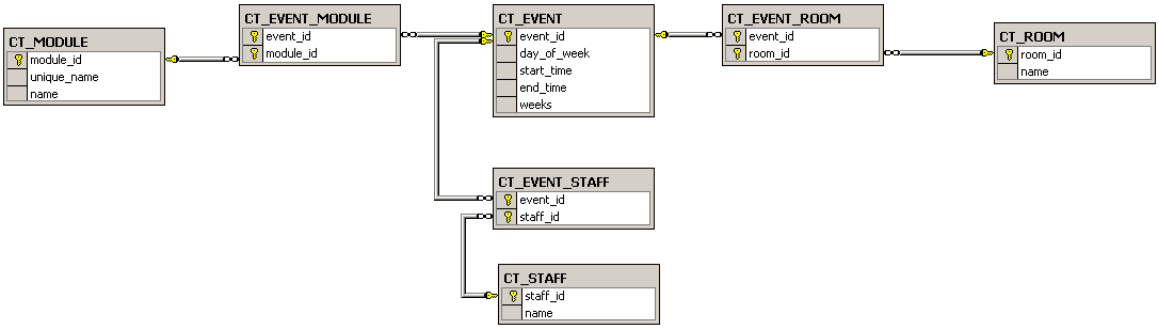


Figure 2: CELCAT Timetabler Database - physical design

To gather the relevant data from this database the `unique_name` for a course must be located first from the QL database, see QL Database below. The name for a course is collected from the CT_MODULE table where the `unique_name` is matched to the one being searched, the corresponding `module_id` is then searched for in the CT_EVENT_MODULE table to find the `event_id`. A module (known as a course such as computing) has a `module_id`, this module then has many events, which are the lectures that occur throughout the academic year for the module. Each of these events are stored in the CT_EVENT table, CT_EVENT_MODULE is an intersecting table matching modules to events. The start and end time etc are then stored in the CT_EVENT table. To find out where an event occurs and which member of staff teaches it, two more intersecting tables are provided, CT_EVENT ROOM and CT_EVENT_STAFF. These match `event_ids` to the relevant records in CT_ROOM and CT_STAFF, which hold, respectively, staff and room information.

QL DATABASE

The complete schema for the QL database is a 400 page document. The database is an ERP system developed by Agresso. Its core functions consist of curriculum definition, admissions, exam processing, attendance marking as well as student information storage. Figure 3 below shows the tables and fields, and the relationships between them, that the timetable application will require access to.

Figure 3: QL Database - physical design

The STMBIOGR table holds all of the students' biographical details at the college with their corresponding student ID as an identifier. The STMAOS table is designed to hold students area of study enrolments. All four data fields that will need to be used are combined as a unique key. This is because a student can be enrolled onto more than one course. The course is identified by an `AOS_CODE` and an `AOS_PERIOD`. The `AOS_CODE` is a code that identifies a course, for example A2MCOP is A-Level computing. The `AOS_PERIOD` is a code that identifies the group, more than one group exist for a course, for example one the computing groups has the `AOS_PERIOD` 09B1. The `AOS_CODE` concatenated with the `AOS_PERIOD` is used as a unique name for a course in the CELCAT *Timetabler* database (called a module in the CELCAT database). The `acad_period` denotes the academic period that the student is taking that particular course. This is because a student can take the same course in more than one academic year, for example GSCE mathematics retakes.

The QL database is used to collect the courses that a student is currently enrolled to; the timetable information for these courses is then gathered from the CELCAT *Timetabler* database, which is discussed in the section above.

SQL AND STORED PROCEDURES

To gather the data discussed in the above section SQL will be used. SQL – Structured Query Language is a language used to interact with databases. It gives the user the capability to insert, retrieve, modify, and delete data in a relational database. It is the language that most databases support. SQL Server 2005 uses a version of SQL called Transact-SQL. It provides the user with a basic set of commands that can be used to perform complex queries. Transact-SQL is an extension of the language defined in the SQL standards published by the International Standards Organization (ISO) and the American National Standards Institute (ANSI).

Stored procedures are precompiled versions of Transact-SQL statements. Stored procedures can take in parameters, just like a programming method. The parameters can then alter the query to tweak what is being carried out. A stored procedure can have as many as 2100 parameters. A stored procedure is stored under a name and run as a unit. To run a stored procedure the EXECUTE command is called followed by the name of the procedure, and any parameters to be passed. A Transact-SQL statement is compiled when it is called from a program or by user input. This can take up processing time. A Transact-SQL statement that is run multiple times can be run as a stored procedure.

The main concerns when collecting the data is speed and efficiency. The data must be returned as quickly as possible without any large processing overhead. A query may not seem that long, for example a 5 line query that collects names of all students on a particular course. The network traffic increases when 5000 clients run the same query throughout the day, which can amount to a lot of congestion. Stored procedures are designed to decrease this congestion on the network. Rather than having clients executing queries, the queries can be kept server side in the form of a stored procedure. When data is gathered from the databases at Halesowen College the same SQL query would be run for every timetable generated. To gather the timetable data from the two databases the only information required is a students ID. This can be passed as a parameter to a stored procedure, which then returns all the data exactly the same as running a Transact-SQL query would.

Another advantage of using stored procedures over pure Transact-SQL queries involves compiling (Gunderloy, Jorden, & Tschanz, 2006). When an SQL query is executed the SQL Server reads the query and looks at things such as JOIN's and WHERE clauses and compares that query with all available indexes to see which indexes will function best, if any exist. When the Server knows which index functions best an execution plan is stored in memory. A Transact-SQL query is compiled almost every time it is run (Gunderloy, Jorden, & Tschanz, 2006). As already mentioned stored procedures are precompiled so they already have an execution plan waiting in memory, meaning hey execute quicker.

The stored procedures written for the timetable application will be modular. If for some reason an error is occurring, the query can be tested and edited server side. It will only need to be changed once, rather than searching through the client side code for every occurrence of the SQL. This also means that a query can be written as a stored procedure and called from multiple places within client side code without having to write out the whole SQL. In summary stored procedures separate server side functions from client side. Stored procedures do not need to be used for every query but for the timetable application it is the most efficient method.

Stored procedures will need to be written for the following:

- Gather all a students enrolments and the corresponding lecture information from CELCAT for a given student ID
- Gather all matching students for a given search criteria for the staff search facility

LOGIC LAYER

The logic layer will consist of the ColdFusion Application server and the built in web server that it provides. The main functionalities of ColdFusion have been discussed previously in the background research section. Forta & Weiss (2002) put together a list of some of the functionality the CFML offers, the main being that reading data from, and updating data to databases can be carried out. ColdFusion also has its own built in web server for handling http requests and responses with clients. This will allow the ColdFusion Server to sit in the middle of the presentation and the actual data.

It is a requirement of Halesowen College that ColdFusion is used for dynamic webpage creation. This will allow the timetable application to be integrated into their current system setup. The alternative to ColdFusion that was considered was PHP, but research highlighted that ColdFusion may be easier to learn as it is an extension of HTML with its CFML, rather than PHP using more of a conventional programming language like syntax.

The Application server will read web pages and execute any CFML language encountered; this may include database access or conditional processing. The result of processing the CFML is a webpage consisting of HTML which can be passed to the web server for deployment to the client browser.

COLDFUSION SESSIONS

One of the requirements of the timetable application is that it does not require the user to login to access their timetable. This can be achieved through the use of ColdFusion session variables. When a user first opens a web browser the Intranet web page is opened. This webpage detects the student ID that was used to log into the client machine and a session variable is created holding this student ID. A session variable is held in the application server's memory and is only accessible by the client application that created it. It is accessible by every ColdFusion page that is running on the client machine. This means that the timetable application web pages can access the session variable. A session variable only exists for a finite amount of time which can be specified and it is also deleted when the browser is closed.

An alternative way to achieve the global access will be to pass the student ID between pages via the URL. This method will be acceptable for staff as they can view any student's timetable. Following this method will enable a student to input another student's ID into the URL and gain access to it which is unwanted. Using cookies is another method that was considered. Cookies are beneficial for storing small amounts of data, which is all the timetable application will need. An advantage to using cookies the proposed environment is their use of use through ColdFusion (Ladd, 2002). A cookie is easily placed on the client machine with an expiration date using the `<cfcookie>` tag. One of the major drawbacks with cookies though is that users now have the option to block server's access to saving cookies on their browsers. This would leave the timetable application unusable to users who have cookies blocked.

Sessions also come in useful for detecting the type of user that has logged in. The timetable application is aimed at two users, staff and students. If a student is logged in then their timetable is shown when a timetable request is made. If a staff member is logged in the student search page is opened, from which a search can take place and a request for a timetable made. To be able to detect whether a user is a member of staff or a student another session variable can be used, this variable will be created when a member of staff is logged in. Then the timetable application can test for the existence of the variable, if it exists the user should be redirected to the staff search page. If it does not exist it can be safely assumed that the user is a student. This is because a session variable is deleted when a browser is closed. Figure 4 shows a high level sequence diagram illustrating how the system checks the user that is logged in though the use of SESSION variables.

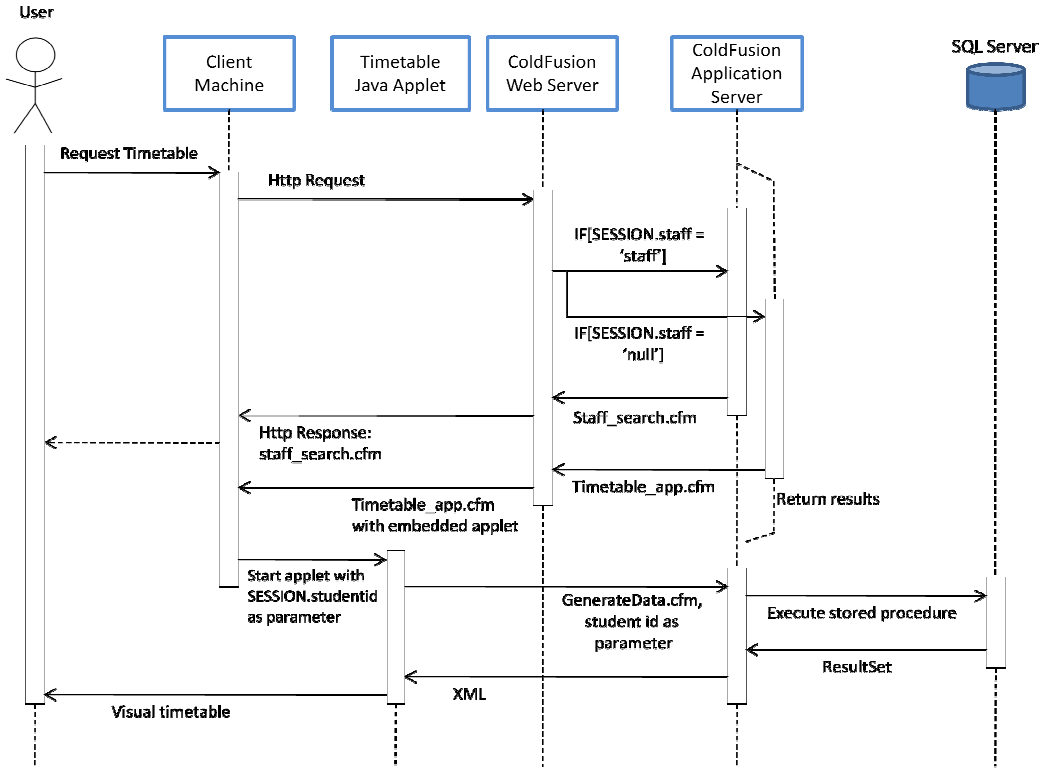


Figure 4: High level Sequence Diagram showing the use of SESSION variables

DATABASE INTERACTION

ColdFusion is the application server at the logical layer, it does not have its own database, but it does allow you to interact with a database the programmer specifies. This is achieved through the use of data sources. A data source is used to interact with a database. A data source holds all of the driver specific information that would need to be supplied every time a connection to a database is made, such as the driver itself, username and password. ColdFusion has an administrator page accessible through a browser on the server machine. From this page data sources can be set. The timetable application will need a data source to the QL and CELCAT databases on the SQL Server.

Through setting database connections this way, ColdFusion allows the author to set and forget the defaults for database access. The connection details can be set once in the administrator and then be referred to via its name from the code.

JAVA APPLET

The actual timetable will be shown in a Java applet that will be embedded within a web page. Due to this the majority of the project involves developing this applet. Java applets are applications that are downloaded to the client browser when accessed from a web page and run client side. For this reason the applet will be discussed in more detail under the section titled presentation layer. As the timetable java applet forms the main GUI of the timetable application, a section is provided detailing its importance. The Java applet is included in this logic layer section as it is the ColdFusion application server that will embed it to a web page.

PRESENTATION LAYER

The presentation layer refers to the client side processing of the GUI. The timetable application will contain two main GUI components, the first discussed below being the HTML forms for the staff search page, followed by the main Java applet that generates the timetable.

STAFF SEARCH

The staff search page allows a member of staff to search for any student that is currently enrolled at Halesowen College. Only members of staff will have access to this page. This is ensured by the use of ColdFusion session variables previously discussed. The search will be achieved through the use of HTML forms produced using the CFML. Ramakrishnan & Gehrke (2002) state that forms are a common way of communicating data from the client tier to the middle tier. The middle tier is the logic layer and the client tier is the presentation layer. Within the HTML form input fields such as text boxes can be inserted, in which the user can enter information. When the form is submitted to the logic layer, the server can pull this information out of the inputs and the application server can pass the information as parameters to the precompiled stored procedures. The results can then be processed by the application server and the generated page passed back to the browser in the presentation layer.

The staff search page will allow a member of staff to search for a student by the following information, student ID, forename and surname. The form will also allow a member of staff to select some sorting options for the results, sort results ascending or descending by either, student ID, forename or surname.

TIMETABLE JAVA APPLET

The Java Applet forms the main part of the project; it will be embedded within a webpage by the Web Server. When it is accessed by the client it is downloaded to the client browser and run. Research identified that Java can allow a developer to write once and run anywhere. This will allow the author to write the applet, and in theory it will run on all the browsers specified by the client at Halesowen College. This will meet requirement 16 without having to write three different implementations.

The logic and data layer deal with passing all the information across to the presentation layer. Stored procedures have been chosen as the method to collect the data. The applet must have access to this data. The way this will be achieved is through the use of the ColdFusion page GenerateData.cfm

GENERATEDATA.CFM

To display a webpage a URL is entered into a browser, the enter key is pressed and the webpage is displayed. In the background the browser makes a connection to the Server and makes a request for the webpage. Assuming that the Server locates the webpage a response is put together and returned to the browser. A webpage consisting of more than just text, which is more often than not, has the extra images and applets etc sent as separate files from the Server that the webpage can refer to. The browser will then make sense of what has been received and do its best to display it. So long as correct HTML has been written the browser will be able to understand and display it. When all of the information has been received the browser – Server connection is closed.

Using this methodology the GenerateData.cfm page can be used to gather data from the SQL Server and pass it back to the timetable applet. The applet will be running client side, so it can make a request to the Server for the GenerateData.cfm page. Java supplies classes for achieving a connection to a host, the host being the Server. The applet will open a connection to the URL:

`http://localhost:8500/finalyrproject/GenerateData.cfm?studentid=`

with a student ID appended to the end through the use of an `InputStream`. The Server takes this as a request and locates the GenerateData.cfm page. This `InputStream` will then be wrapped in an `InputStreamReader` that will read the response from the Server. Discussed above a webpage is passed from a Server as text and any images are sent separately across the connection. The Java applet can read the text in through the `InputStreamReader` and store it within a String variable as no additional files will be sent. Only text in the form of XML will be generated in the response.

The detail of how GenerateData.cfm is constructed will be shown in the implementation section. In the URL request above there is a parameter called studentid passed across from the applet. The GenerateData.cfm will take this parameter and use it as a parameter to call a stored procedure that is pre-compiled on the SQL Server. This stored procedure will collect all the students' enrolments and the corresponding lecture timetabling information. Using CFML the records can be looped through and an XML document produced to give the information a structure so it can be understood by the applet when it has been received.

XML has been used as it is a structured language that can be read by Java, as well as still being human readable. Harold & Means (2001)state that an advantage of using XML is the wide availability of tools parse and write it. This will allow a parser to be written in Java so that the applet can understand and use the data that it has read in. XML uses tags to structure the data, like HTML tags, except the tags are not predefined. A developer can make their own tags to give the data some meaning to make it easier to understand.

An example of the XML that will be produced by the GenerateData.cfm page is shown below:

```
<Student firstname="Efren" lastname="SARGENT" student_id="ben07071174">


        <Lecture title="A2 Tutor Group 014"
        weeks="NNNNNNNYYYYYYYNYYYYYYYNNYYYYYYNYYYYYYNNYYYYNNNNNNNNNNNN">
                <DOW>4</DOW>
                <START>1899-12-30 12:45:00.0</START>
                <END>1899-12-30 13:15:00.0</END>
                <ROOM>5.207</ROOM>
                <staff>Imbimbo: Toni</staff>
        </Lecture>

        <Lecture>
        …
        </Lecture>

        …

</Student>
```

The tags define elements, and these elements can contain more elements as well as attributes. The Student element is the root element, as all of the child elements belong to the Student. The Student element can contain zero or more Lecture elements. Theses Lecture elements represent one lecture that will be need to be shown on the timetable. The Lecture elements then contain the relevant data as elements such as the day of the week, start time and end time etc. The reader should also be able to see the use of attributes in the Student and Lecture elements in the code above. The Student contains the 'firstname', 'lastname' and 'student_id' as attributes. The Lecture contains the 'title' and 'weeks' as attributes. The weeks attribute is a String of 'Y's' and 'N's' referring to 'Yes' and 'No'. One element of the String corresponds to a week at Halesowen College. If the value at a certain week is 'Y' then this lecture must shown on that week and visa versa.

PARSER

Once the connection has been closed between the browser and Server the Java applet will have a String variable containing the XML document which contains all of the information that must be used to generate a timetable. The applet needs to make sense of the data that is contained; this can be fulfilled using a parser. The timetable applet will use the Simple API for XML (SAX) to achieve the parsing. SAX, provides a streaming parser API (Brownell, 2002) and falls under the parser level category. SAX does not give any structure to the data; it is all left to higher levels. This is the opposite in most other parser level API's such as jDOM, which provides a generic object tree data structure. Using an API such as jDOM leaves the developer constrained to the structure that the API has generated. For this reason jDOM is not used. SAX is event oriented, meaning all the structure is built by the higher levels as an event is created. A SAX implementation requires the developer to override the provided methods to define what needs to be done as certain elements are found. SAX reads the XML from a stream, when a starting element is found the overridden `startElement()` method is called and the element tag is matched, and the corresponding code is called to handle it, likewise, when the closing tag is read in, the `endElement()` method is called. This way the timetable application can read the code shown above, and create a new Student object holding the attributes, when the Student element is read. When a Lecture element is reached a new lecture object can be created storing the related data. When the whole XML has been parsed the applet will have objects containing all the data in a structure specific to the program. In contrast jDOM provides methods for traversing the tree structure it creates allowing manipulation of the data, as no data manipulation is required by the timetable applet the jDOM API is irrelevant. The timetable application just requires basic reading of the XML document which SAX is best suited. In terms of memory consumption, SAX is preferred. With many DOM implementations in Java, each byte of the XML can take up to 10 bytes of memory (Brownell, 2002). A 3Mb file can easily take up to 30Mb of memory. With SAX the memory consumption can be out as only the data needed is gathered from the XML.

DATA STRUCTURE

Lecture objects will be created to hold the information representing one lecture. These lecture objects will be stored in an `ArrayList`. An `ArrayList` size is not predefined like an `Array`. This is required as it is not known how many lectures a student's timetable will need to show. A Lecture object will take the structure shown in figure 5. The length variable will be calculated using an algorithm to work out the difference between the start and end times. Figure 6 shows the class diagram for the Student object.

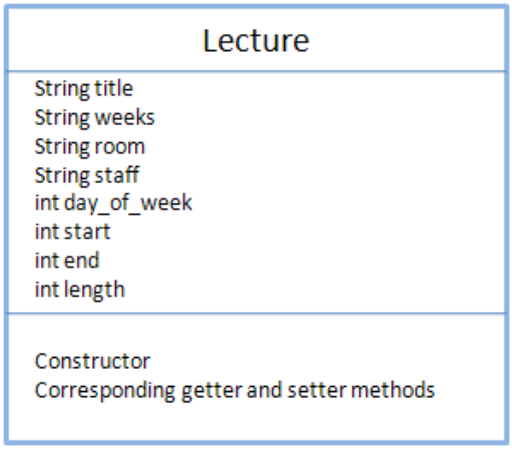


Figure 5: Lecture Class diagram

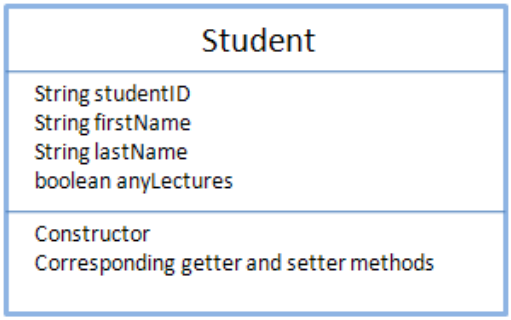


Figure 6: Student Class diagram

The variables in the above classes are defined when SAX hits the element that instantiates the event that declares a new Lecture or Student object.

Using the data that has been parsed and stored a graphical timetable must be generated. This is the only information that is available. Research revealed that using the Java2D classes is the best suited way of accomplishing this. Algorithms to calculate where to draw the actual lectures on a timetable must be written. The detail is discussed in the section Implementation. The following sections discuss the final output of the Java Applet. This is what the user will be interested in.

GRAPHICAL USER INTERFACE (GUI)

The main point to keep in mind when designing the GUI is usability. A system that clients do not want to use is pointless. Especially with the proposed timetable application, it is being designed at built to be used by students and members of staff on a daily basis. If the final system does not have an appealing GUI it is not meeting its purpose. Requirement 12 of the requirements definition would also not be met. Three main GUI goals are defined by Brink, Gergle & Wood (2002) that should be followed when designing a GUI.

SIMPLICITY

The content that is shown on the page is why the user is looking at it. The structure supports the content to make it appealing. Sometimes the structure becomes too much and overpowers the content. Keeping the structure simple is what the author needs to keep in mind. Halesowen College already have a system for staff that is complex. To differentiate the timetable application and make it a success it must be kept simple. It must be clear to a user what component does what. The title must be recognized as a title and that navigational elements are clearly for navigation (Brinck, Gergle, & Wood, 2002). This should be obvious to the user at first sight. It should not require the user to first study the page. Conserving a page to just the required elements will result in an elegant design.

CONSISTENCY

Consistency increases ease of use, reinforces a sense of structure, and decreases learning time associated with navigating your site (Brinck, Gergle, & Wood, 2002). Consistency must apply to the whole site. It should be immediately obvious to a user that they are still on the College Intranet, whether it be on the homepage, staff search page or the College news page. The design of any web pages that will exist in the timetable application will need to be consistent with the structure and elements that are already used. Consistency will have a strong influence on how the user perceives the timetable application. Aligning elements along common axes, using consistent font sizes, and repeating font styles used in the current pages at Halesowen College, will mean users will start to subconsciously predict where elements are. Figure 7 shows some screenshots from the current Intranet pages. This applies to mainly the Search page that will be used by members of staff. The timetable that is displayed to students will not require any interaction, so consistency of elements is not as vital. Simplicity and focus are the key goals with the timetable generation.

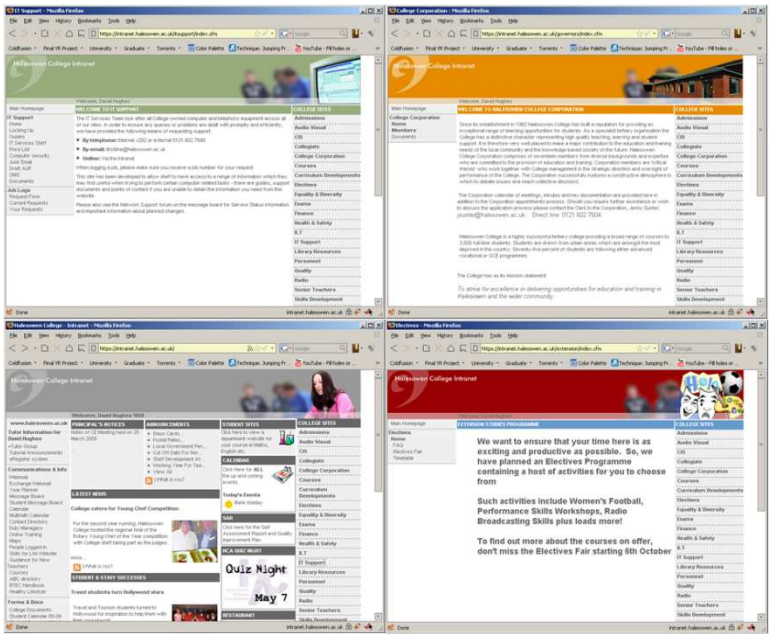


Figure 7: Halesowen College Intranet screenshots

FOCUS

Emphasis on certain areas of the page will draw a user's attention. This can draw a user directly to key information on a page. Whether the point of emphasis is an image, textual information, or lectures on the timetable. This can be achieved through the use of colour, increased font sizes, increasing the size of elements etc. Without the focus, simplicity and consistency are not as effective. All three goals complement each other.

A full understanding of the end user will need to be gathered to achieve a successful GUI. Whether the final design choice is a success or not will be shown through client acceptance.

PRINTING

The timetable application must be able to provide a printout of the generated timetable for the user. The printout will not need to be in colour but it will need to be very clear and concise. The way that the lecture information will be shown on a lecture will need to be different to the how it is onscreen. A printed version can use smaller fonts as they are easier to read on paper than it is onscreen. It must be clear whose timetable it is and the week that it is representing. This should be shown at the top of the timetable.

4. IMPLEMENTATION

The Implementation section will focus on showing how the timetable has been implemented to get from an xml document like the example in figure 8, to a visual timetable shown in figure 9. The whole implementation will not be shown, detail and code examples will be provided where it is needed, such as where the author found the solution difficult, or where complex code has been used.

```
<?xml version="1.0" encoding="UTF-8"?>
<Student firstname="Efren" lastname="SARGENT" student_id="ben07071174">

        <Lecture title="A2 Tutor Group 014" weeks="NNNNNNNYYYYYYYNYYYYYYYNNYYYYYYNYYYYYYNNYYYYNNNNNNNNNNNN">
           <DOW>4</DOW>
                    <START>1899-12-30 12:45:00.0</START>
                    <END>1899-12-30 13:15:00.0</END>
                    <ROOM>5.207</ROOM>
                    <staff>Imbimbo: Toni</staff>
        </Lecture>

        <Lecture title="A2 Tutor Group 014" weeks="NNNNNNNNNNNNNYNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN">
           <DOW>2</DOW>
                    <START>1899-12-30 13:15:00.0</START>
                    <END>1899-12-30 13:45:00.0</END>
                    <ROOM>1.206 (LRC Business: TALC)</ROOM>
                    <staff>Imbimbo: Toni</staff>
        </Lecture>

        <Lecture title="A2 Tutor Group 014" weeks="NNNNNNNYYYYYYYNNYYYYYYNNYYYYYYNYYYYYYNNYYYYNNNNNNNNNNNN">
```

Figure 8: GenerateData.cfm response

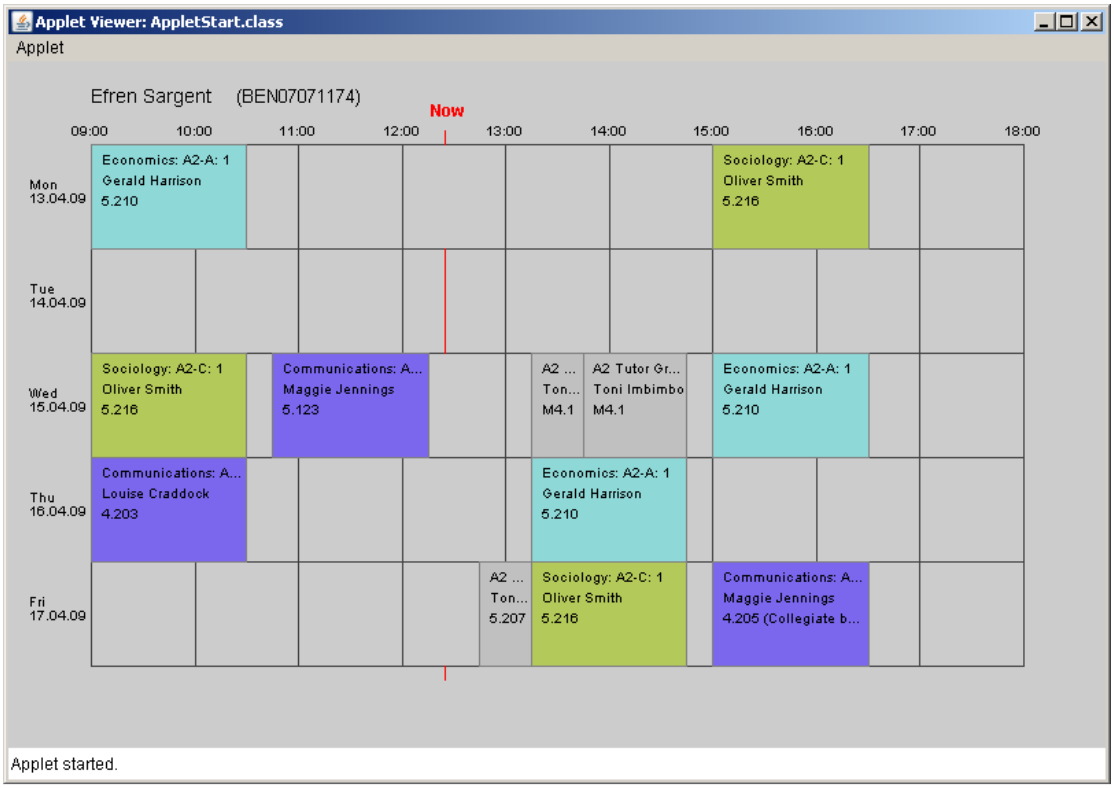


Figure 9: Generated Timetable based on XML snippet in Figure 8

GENERATEDATA.CFM

When a user accesses the timetable the Applet is loaded and makes a request to GenerateData.cfm with a student ID as parameter, this occurs whether it is a student requesting a timetable or a member of staff viewing a timetable from search results. GenerateData.cfm carries out a stored procedure on the SQL Server which returns a record set containing all the lecturers that must be shown on the students' timetable. The CFML then loops around the record set and produces an xml response like the one shown in figure 8.

STORED PROCEDURE

The stored procedure that the GenerateData.cfm page calls is shown in figure 10. The section called database design contains details on how the data is stored in the CELCAT Timetabler and QL databases. This stored procedure collects all the required data following what was discussed in database design.

```
CREATE PROCEDURE FYP.getLectureEventsByStudentID (
@studentid char(11)
)
AS

SELECT CT_EVENT.event_id, CT_EVENT.weeks,
modules.name AS lecture, modules.student_id, modules.forename, modules.surname,
day_of_week, start_time, end_time,
rooms.roomName,
staff.staffName
FROM   CT_EVENT_MODULE,
                (SELECT     STMBIOGR.forename AS forename, STMBIOGR.surname AS surname, module_id, name, STMAOS.student_id, aos_code, aos_period
                FROM        offline_celcat.dbo.CT_MODULE AS CT_MODULE, halesowen_college.dbo.STMAOS AS STMAOS,
                                halesowen_college.dbo.STMBIOGR AS STMBIOGR
                WHERE     (CT_MODULE.unique_name = RTRIM(STMAOS.AOS_CODE) COLLATE DATABASE_DEFAULT
                + '' + RTRIM(STMAOS.AOS_PERIOD) COLLATE DATABASE_DEFAULT)
                AND STMAOS.student_id = @studentid
                AND STMAOS.acad_period= '08/09'
                AND STMBIOGR.student_id = STMAOS.student_id) AS modules,

                (SELECT CT_EVENT.event_id, CT_ROOM.name AS roomName
                FROM CT_ROOM, CT_EVENT_ROOM, CT_EVENT
                WHERE CT_EVENT.event_id = CT_EVENT_ROOM.event_id
                AND CT_EVENT_ROOM.room_id = CT_ROOM.room_id) AS rooms,

                (SELECT CT_EVENT.event_id, CT_STAFF.name AS staffName
                FROM CT_EVENT, CT_EVENT_STAFF, CT_STAFF
                WHERE CT_EVENT.event_id = CT_EVENT_STAFF.event_id
                AND CT_EVENT_STAFF.staff_id = CT_STAFF.staff_id) AS staff,

        CT_EVENT

WHERE CT_EVENT_MODULE.module_id = modules.module_id
AND CT_EVENT_MODULE.event_id = CT_EVENT.event_id
AND CT_EVENT.event_id = rooms.event_id
AND CT_EVENT.event_id = staff.event_id
GO
```

Figure 10: Stored Procedure - getLectureEventsByStudentID

The stored procedure substitutes `@studentid` with the student ID that has been passed as a parameter. Many implementations of this stored procedure were attempted but this resulted to be the most efficient. It is collecting and using data from two databases and nine tables. The nine tables used are put together into derived. The whole stored procedures results are made based on the derived tables. The results are first collected from the derived table queries in the FROM clause, these result sets then act as tables themselves from which the main query can pick the required data. The first derived table contains all of a student's course enrolments from the QL database and the corresponding `unique_names` on the CT_MODULE table in the CELCAT *Timetabler* database. The second contains all of the CT_EVENT table events and the corresponding CT_ROOM names. The third contains all of the CT_EVENT table events and the corresponding CT_STAFF names. The main outer query then brings together all of the data.

CFML

Figure 8 is achieved through the use of the CFML tag `<cfxml>` within the GenerateData.cfm page. The code below in figure 11 demonstrates how it is used.

```
 8 <cfxml variable="TimetableXML">
 9    <Student student_id="<cfoutput>#URL.studentid#</cfoutput>"
10    firstname="<cfoutput>#XMLFormat(accessInfo.forename)#</cfoutput>"
11    lastname="<cfoutput>#XMLFormat(accessInfo.surname)#</cfoutput>">
12    <!--- #XMLFORMAT(string) is used to escape special characters such as (&) --->
13       <cfloop query="accessInfo">
14          <Lecture title="<cfoutput>#XMLFormat(accessInfo.lecture)#</cfoutput>"
15                weeks="<cfoutput>#XMLFormat(accessInfo.weeks)#</cfoutput>">
16             <DOW><cfoutput>#XMLFormat(accessInfo.Day_Of_Week)#</cfoutput></DOW>
17             <START><cfoutput>#XMLFormat(accessInfo.START_TIME)#</cfoutput></START>
18             <END><cfoutput>#XMLFormat(accessInfo.END_TIME)#</cfoutput></END>
19             <ROOM><cfoutput>#XMLFormat(accessInfo.ROOMNAME)#</cfoutput></ROOM>
20             <staff><cfoutput>#XMLFormat(accessInfo.staffname)#</cfoutput></staff>
21          </Lecture>
22       </cfloop>
23    </Student>
24 </cfxml>
```

Figure 11: <cfxml> taken from GenerateDta.cfm

Figure 11 is a section of the GenerateData.cfm page. The full page is available for viewing on the supplied data CD. Lines 8 and 24 contain the opening and closing tags for `<cfxml>`. Everything produced within these tags will result in an XML document that can be parsed by the Java Applet. As seen in line 8 the XML is stored as a variable called `TimetableXML`, this is what is output by the Server. At a glance the reader should be able to see that it already takes the form of an XML document. The interesting part begins on line 13. This is where a loop is started. `accessInfo` is the variable where the record set returned from the stored procedure is stored. The loop will go through every record and carry out the CFML that is between the opening and closing `<cfloop>` tags. Data is drawn from the record set and put into the correct tags of the XML.

```
<START><cfoutput>#XMLFormat(accessInfo.START_TIME)#</cfoutput></START>
```

The code above is taken from Figure 11, line 17. This is how a piece of data is taken from the record set and put into the XML. `XMLFormat()` is a defined function in ColdFusion that takes its parameter and escapes any special characters that occur, such as ampersand '&'. `accessInfo.START_TIME` refers to a column in the `accessInfo` variable record set table. The ColdFusion simply pulls out the data that is stored in the `START_TIME` column of the current record.

JAVA APPLET

The Applet was chosen to be the best suited way of showing the timetable. The applet is embedded within a webpage that opens in a pop up when the timetable link is clicked on. The applet fits to the size of the window. This gives the effect that it is not an applet embedded within a web page but just a dynamic webpage.

DATA PROCESSING

Before the timetable can be drawn the data gathered from the databases must first be processed so that it is all in the correct format.

When the Java applet has read all the xml in it is parsed using the SAX parser written in the `SAX_Lecture` class. This class is available on the data CD for viewing. During parsing it creates a new `Lecture` object for each of the lectures in the XML document. When a new `Lecture` object is created and all the variables declared it is added to an `ArrayList` of type `Lecture`. This allows all of the `Lecture` objects to be structured to allow easy access to the information they hold. When the `start` and `end` times are set in a `Lecture` object by the `SAX_Lecture` class, the data gathered

from the XML must be formatted. SQL Server stores the times in the following format in the CELCAT *Timetabler* database '1899-12-30 10:45:00.0'.These fields are put into the format 'HHMM' and stored as an integer.

If there are any lectures in the ArrayList produced, there are lectures to be shown for the student. These lectures do not necessarily occur on the current week (current week being the week the timetable is requested), so the lectures ArrayList must be filtered to only contain those that need to be displayed by using the weeks variable in a Lecture object. College weeks differ from the current week in the year, so the difference is calculated. Any lecture that does not occur on that week is dropped from the ArrayList. If it is Saturday or Sunday the lectures are filtered to only contain those that occur on the next college week.

For each of the lectures that now exist in the ArrayList, the lecture length is set by calculating the difference between the start and end times. Each Lecture object represents a lecture that needs to be shown on the timetable. Therefore each has an assigned lecturer. The lecturer is stored as a String in the format 'surname: forename'. This is the formatting of the database which when displayed on a timetable is not easy to view. Simplicity is important as previously discussed in the design, so an algorithm has been written to switch the surname and forename around and remove the ':'. This leaves the String ready to display.

JPANELGUI() – JAVA2D

Now the data has been processed and reduced to just what needs to be shown on the timetable. The JPanelGUI() class deals with the drawing of the timetable through the use of Java2D. It extends JPanel() to allow the author to override paintComponent() to draw the timetable to a JPanel() within the Applet container

Firstly all of the global variables are set, these include the width and height variables, these are the first thing the paintComponent() method carries out. The paintComponent() method is overrides the JPanel's paintComponent() method and is called when the applet is first run and every time the applet is refreshed. There are two cases when the paintComponent is recalled. The first is from within the applet, such as when a button is pressed. The second is from outside the applet, such as when a screen is resized or a user scrolls the mouse on the webpage. The paintComponent() method executes all of the contained code when it is recalled. Everything is drawn dynamically to fit in the window. So the double width and height variables are collected in case the window is resized, when the paintComponent() redraws it will draw according to the window size. This includes increasing / decreasing the grid size, lecture positions, recalculating the lecture size according to its length, font sizes, and text positions. The double across and down variables are also set, these contain how many pixels are between the lines on the grid. This is a vital part of the program, as all the lectures are positioned using the across and down variables. If a lecture begins at 10oclock, a rectangle must be drawn beginning at the second block of the grid. So the coordinates are specified using the across value multiplied by how many columns into the grid it must go, plus the X axis margin. Which is also set when setting up the variables, this leaves space for the labels around the grid along with the Y axis margin.

DRAWGRID()

All of the timetable generation is called from within the paintComponent() method. After the global variables are set the grid of the timetable can drawn. A line is drawn simply using the code on line 463 in figure 12. g2 is a Graphics2D object that allows a programmer to render 2-dimensional shapes, text and images. It is an extension of the Graphics class.

```
461     for (int i = 0; i <= 5; i++) {//draws lines going vertical
462
463         g2.draw((new Line2D.Double(x, y, (width - (X)), y)));
464         y = y + down;
465     //System.out.println(y);
466     }
```

Figure 12: drawGrid() code snippet

The `for` loop that begins on line 461 in figure 12 loops 5 times, each time drawing a line beginning at `(x, y)` and ending at `((width – X), y)`. `x` and `y` are variables that are copies of the `X margin` and `Y margin` variables that are used locally in the `drawGrid`() method. The 5 lines drawn are horizontally across the screen beginning at `x` to the `width – X` pixel position. The down value is added to `y` each time so that there is an equal space between each line. The `down` value is dynamic depending on the window size. The code for drawing the vertical lines is the same apart from the `across` value is added to `x` each loop and 9 lines are drawn going down the screen.

The labelling is carried out next to label the days and times on the grid.

DRAWLECTURES()

The `drawLectures`() method is the most involved method that the `paintComponent`() calls. It is worth noting that it is only called when there are lectures to display. A check is first done in the `paintComponent`() method. This is because before it can do any drawing it must first calculate the pixels where the lecture is drawn. A lecture is represented by a rectangle. The rectangle is drawn exactly on the grid where the lecture begins and ends. It is drawn precise to a pixel. The method can be split into two main parts, the processing and the drawing. Figure 13 shows the beginning of the `drawLectures`() algorithm written in pseudocode. Pseudocode has been used to make it easier to understand for the reader and to shorten it down for the purposes of this paper. The full code is available on the data CD.

```
1   FOR (each lecture in lectures ArrayList)
2
3       //calculate how far down on the Y axis to draw in pixels
4       CASE  DayOfWeek  OF
5                   0       : pixelsDownGrid = YaxisMargin
6                   1       : pixelsDownGrid = YaxisMargin + down
7                   2       : pixelsDownGrid = YaxisMargin + (down * 2)
8                   3       : pixelsDownGrid = YaxisMargin + (down * 3)
9                   4       : pixelsDownGrid = YaxisMargin + (down * 4)
10                  default : pixelsDownGrid = YaxisMargin
11      ENDCASE
12
13      //calculate how far across on the X axis to draw
14      split start time into double hours and double min
15      //hours
16          pixelsAcrossGrid = hour - 9 + XaxisMargin //subtract 9, the interger left is how
17                                                    //many columns on the grid to move in from the XaxisMargin
18
19          WHILE (hours > 0)
20              Compute pixelsAcrossGrid as PixelsAcrossGrid + across //adds the across value for every hour
21              DECREMENT hours
22          ENDWHILE
23      //minutes
24          Compute minAcross as ((across / 60) * min)
25          Compute pixelsAcrossGrid as pixelsAcrossGrid + minAcross
26      //draw Lecture
27      Draw rectangle(pixelsAcrossGrid, pixelsDownGrid, lengthOfLecture, down)
```

Figure 13: drawLectures() pseudocode 1

As the reader should be able to see in figure 13, the processing takes up most of the algorithm. Once the pixel positions are calculated rectangle can be drawn on the screen. The lecture position is all calculated according the across and down values, this ensures that everything is drawn precise. The values calculated are used as coordinates in line 27 to draw the rectangle. The first two refer to the top left hand corner of the rectangle position. The third represents the pixel width and the fourth the

pixel height. `lengthOfLecture` used in line 27 is pre-calculated and stored in the `Lecture` object. The values calculated are also stored in the `Lecture` object so they can be used again in a mouse over function, to calculate whether the mouse is within the lecture rectangle. As well as drawing a rectangle the `Graphics2D` object of the `paintComponent`() is used to fill the rectangle with a colour and to draw a second rectangle to give the lecture a border.

```
1  //set up variables for texual information positioning
2      //dynamic to where the lecture rectangle is positioned on the grid
3      Compute stringXpos as pixelsAcrossGrid + xAxisIndent; //calculates the X axis position of where to draw the strings
4      Compute titleStringYpos as pixelsDownGrid + titleIndent;//calculates the Y axis position of where to draw lecture title
5      Compute staffStringYpos as pixelsDownGrid + staffIndent;
6      Compute roomStringYpos as pixelsDownGrid + roomIndent;
7
8  //draw the Strings
9      //only writes the lecture info when the applet is greater than (400, 400) pixels
10         IF (width > 400 AND height > 400)
11             //passes the string and pixel position values to another method to do the drawing
12             Call writeLecInfo(lecture title, 1, length - xAxisIndent, g2, stringXpos, titleStringYpos, fontSize)
13             Call writeLecInfo(lecture staff, 2, length - xAxisIndent, g2, stringXpos, staffStringYpos, fontSize)
14             Call writeLecInfo(lecture room, 3, length - xAxisIndent, g2, stringXpos, roomStringYpos, fontSize)
15         ENDIF
16 ENDFOR
```

Figure 14: drawLectures() pseudocode 2

Figure 14 shows the second part of the pseudocode that shows how the text is positioned on the screen within a lecture like in figure 9. `xAxisIndent` used in line 3 leaves some pixel space between the edge of the rectangle and where the text is drawn. The values computed in lines 3 – 6 are passed as parameters to a method called `writeLecInfo()`. This method takes as parameters, a string to write to the screen and positioning information that is calculated in the first part. `Length – xAxisIndent` in line 12 – 14 calculates the pixel space the string must fit into within the lecture rectangle. It also takes the `paintComponent() Graphics2D` object.

TEXTUAL LECTURE INFORMATION

`writeLecInfo()` provides a solution to one of the main problems that the author encountered. This was fitting the text within a rectangle that represents a lecture. As can be seen in Figure 9 some of the lectures last only half an hour, so there is not enough space to display all of the information in the rectangle. The font could be reduced to fit it in but it becomes too small to read. The solution was trim off the text that does not fit in and append '…' to the end to inform the user that more information exists. This became difficult though because once a lectures rectangle has been drawn to the screen it cannot be referenced to in any way like a component can. That is the reason the coordinates calculated when drawing lectures are stored within a `Lecture` object. The String to be drawn within a lecture can then be tested to see if it fits within the coordinates. Figure 15 shows the complete code for this method with commenting to aid the reader in understanding it. There are two parts to the method that differs what is executed when the String does not fit within a lecture rectangle. If the `print` Boolean is set to true, meaning that the `paintComponent`() is drawing the timetable ready for printing, all of the textual information must be displayed. If the `print` Boolean is set to false then the timetable is being displayed as normal on screen, if the String does not fit within the given rectangle coordinates, a loop is entered which trims text off the String and appends '…' until a point is reached that the String will fit in.

The `writeLecInfo`() method makes use of a method in the `Process` class called `stringPixelWidth`() which takes as parameters a String and an integer font size. This method contains an algorithm to calculate the pixel length of a String as opposed to the actual character count length. The font size must be passed because a larger font will increase the overall length of the String and visa versa.

```
711  public void writeLecInfo(String text, int textType, double lecLength, Graphics g2, double x, double y, int fontSize) {
712
713      g2.setFont(new Font("ariel", Font.PLAIN, fontSize));
714      String tempString;
715
716      tempString = text;
717      int tempStringLength = process.stringPixelWidth(tempString, fontSize);//gets pixels length of string
718
719      boolean stringTrimmed = false;
720      if (!print) {//draws as much of the string and concatenate "..." to the end
721          //loop through till get to position in the string that fits within length
722          while (tempStringLength > lecLength && tempStringLength > 0) {
723
724              tempString = tempString.substring(0, tempString.length() - 4);
725              tempString = tempString + "...";
726              tempStringLength = process.stringPixelWidth(tempString, fontSize);
727          }
728          g2.drawString(tempString, (int) x, (int) y);
729      } else {//if printing want to draw as much within lec then move down a line and draw the rest
730          String buffer = "";//buffer stored the part of the string that will not fit within lec
731          boolean firstloop = true;
732          while (tempStringLength > lecLength && tempStringLength > 0) {
733              if (firstloop) {//for the first loop can substring to the end of the tempString.
734                              //but after that want to remove the "-" that gets concatenated
735                  stringTrimmed = true;
736                  buffer = tempString.substring(tempString.length() - 2, tempString.length()) + buffer;
737                  firstloop = false;
738              } else {
739                  buffer = tempString.substring(tempString.length() - 2, tempString.length() - 1) + buffer;
740              }
741              tempString = tempString.substring(0, tempString.length() - 2);
742              tempString = tempString + "-";//adds a "-" to the end to show that the rest of the text is on the next line
743              tempStringLength = process.stringPixelWidth(tempString, fontSize);
744          }
745
746          if (stringTrimmed) {//if the string has been trimmed down for printing
747              g2.drawString(tempString, (int) x, (int) y);//draw both tempString
748              g2.drawString(buffer, (int) x, (int) y + 7);//and buffer
749
750              //need to increment the Y axis indents if the string has been moved to a new line
751              switch (textType) {//textType corresponds to what the string is, lecture title, room etc..
752                  case 0:
753                      break;
754                  case 1://lecture title
755                      titleStringYpos = titleStringYpos + 10;
756                      staffStringYpos = staffStringYpos + 10;
757                      roomStringYpos = roomStringYpos + 10;
758                      break;
759                  case 2://staff teaching lecture
760                      staffStringYpos = staffStringYpos + 10;
761                      roomStringYpos = roomStringYpos + 10;
762                      break;
763                  case 3://room number
764                      roomStringYpos = roomStringYpos + 10;
765                      break;
766              }
767          } else {//otherwise just draw the tempString
768              g2.drawString(tempString, (int) x, (int) y);
769          }
770      }
771  }
```

Figure 15: writeLecInfo() code

LECTURE COLOURING

When the event information is gathered from the CELCAT *Timetabler* database it is already ordered ascending by lecture title. Knowing that every time event information is returned it will be in this order an algorithm has been written that simply checks if the next `Lecture` objects title in the `drawLectures()` for loop is different from the previous. If it is then a new colour is set, if not then the colour remains the same. This results in a colour coded timetable. This can be seen in appendix C that shows a screenshot of one of the timetables generated. All of the Economics lectures are one colour and all of the Sociology another.

HERE NOW FEATURE

Shown in appendix C is a screenshot that has been generated from some XML produced by GenerateData.cfm for a given Student ID. A red line can be seen going through the lecture on Wednesday just before 12 o'clock. This is the time and day the timetable was viewed. The red line

shows the current time and day on the timetable. With this it can be seen at a glance the lecture that should be attended. The `paintComponent()` method redraws the timetable every refresh. The applet refreshes every time the operating system instantiates a refresh, for example, when a window is resized, dragged or uncovered. The applet has been implemented to refresh every time the mouse is moved. The primary reason for this is for the mouse over feature discussed in the next section. Having the `paintComponent()` redraw every refresh gives the effect that the red line is moving throughout the day.

With the amount of redrawing that happens due to all the refreshes it was a risk that screen flicker may occur. No flicker occurs though as it the processing and drawing is not memory intensive, only basic shapes are being used to draw. If components, such as buttons, were being created and added to the JPanel then the memory consumption would increase, which would increase the chances of screen flicker happening.

The line will not draw if it is a weekend or the time that the timetable is viewed is greater or less than the times shown on the grid. This keeps the timetable professional looking. There will be no point displaying the line after the grid has finished. The algorithm used gets the current time and tests whether it must be shown on the timetable. If it is greater than 6 pm or less than 9 am then the algorithm will halt.

MOUSE OVER FEATURE

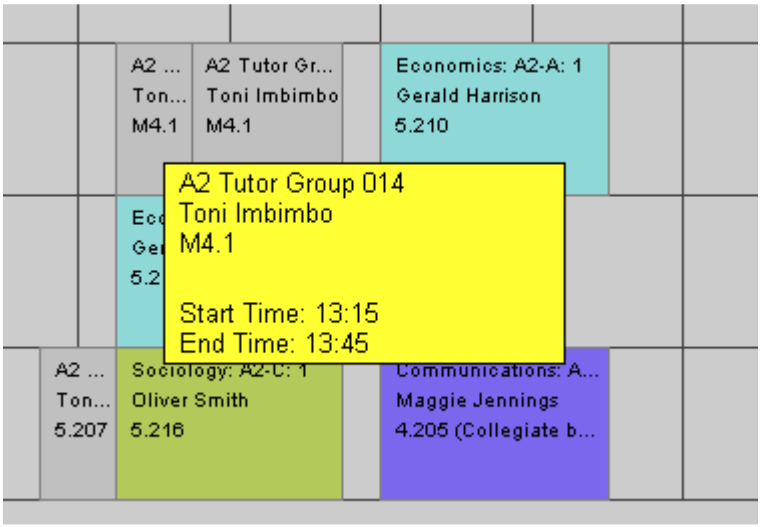


Figure 16: Screenshot demonstrating the mouse over feature

The main problem encountered when drawing the lectures to the JPanel is fitting the text within a rectangle. As discussed under the section drawLectures() the text that needs to be shown is trimmed until a String is left that does fit with '…' appended to the end of it. The mouse over feature allows a user to hold their mouse pointer above a rectangle to display the rest of the information. This is similar to the tool tip concept available in Java. A tool tip allows a user to hold the mouse pointer above a component and informative information is shown, similar to the way in figure 16. The problem with this is that no components are used. Once a rectangle is drawn to the JPanel there is no way to refer back to it. This is the reason the coordinates calculated for a rectangle are stored in the corresponding `Lecture` object.

The mouse over function works by getting the mouse pointer coordinates and checking if they are within any of the `Lecture` objects rectangle coordinates stored in the Lecture `ArrayList`. If they are, a rectangle is drawn to accommodate the textual information and the `writeLecInfo()` method is called with the text information to be displayed for that lecture as a parameter. This information box will stay displayed as long as the mouse pointer remains within the coordinates of the lecture rectangle. As soon as it moves out from the rectangle the timetable is redrawn with the

information box removed. A check on the mouse pointers coordinates must be carried out every mouse move. Also a timer is used that resets every time the mouse moves. If the timer completes then an `actionPerformed` method collects the mouse pointer coordinates, and checks if they are within a lecture. This will mean the user will need to pause above a rectangle for the specified time to see the information box. In earlier builds of the timetable applet the timer was not used, it resulted in an information box being displayed for every rectangle the mouse went within when a user drags their mouse across the screen.

PRINTING

Requirement 9 states that the final application should allow a user to print a timetable. The timetable application has been implemented to print a timetable in black and white. There is no need for it to be printed in colour. The reason for it being available on the Intranet and written in Java is so that it can be accessed from anywhere, therefore the amount of timetables printed at Halesowen College will be reduced. The `paintComponent()` method also carries out the drawing of the timetable when it is printed. When a user requests a printout a Boolean print variable is set to true and the applet is redrawn. When `paintComponent()` carries out the initial setting up of the global variables a check is done to see if print is true. If so, rather than setting the `width` and `height` to the container width and height, they are set to the size of an A4 page. This will mean all methods that use these variables will draw their parts to fit an A4 page. Methods that render the timetable differently will contain a check at the beginning to determine how to drawn. For example, the `writeLecInfo()` will display all the text within a lecture rectangle when it is being printed by putting the information on a new line. This is the only way to get it to fit within the lecture. It is difficult to read it onscreen when formatted this way but with a printout it is suitable. An example printout is provided in appendix E.

ERROR MESSAGES

The only time an error message will need to appear within the Java applet is when no lectures are available to display. This can be caused by either no lectures exist for the current week or no lectures exist at all. Once the lectures have been filtered to the current week it is not known if none exist because there are none to display at all or whether it is just for the current week. A check is done as soon as the XML has been parsed. This is shown in figure 17. It contains the method within the `Process` class that is called from `AppletStart` to do the check. A Boolean variable is set in a Student object that can be accessed to determine which error message to show. If no lectures exist in the lectures `ArrayList` but the Boolean variable in the student object is true then it is just the current week that no lectures happen. If the Boolean is set to false then it is for the whole year. The error message:

'No Lectures to display for the Student ID: …'

is shown in the same format as figure 18. This could be due to the ID being incorrect or that the student no longer attends the college but exists in the database still. This cannot be distinguished so this error message should give the user reason to investigate if the Student ID is correct.

```
52      public void lecturesForStudent(ArrayList<Lecture> lectures, Student student) {
53          if (lectures.size() == 0) {
54              student.setAnyLectures(false);//sets to false if no lectures for student
55          } else {
56              student.setAnyLectures(true);//sets to true if no lectures for student
57          }
58      }
```

Figure 17: check to see if any lectures exist

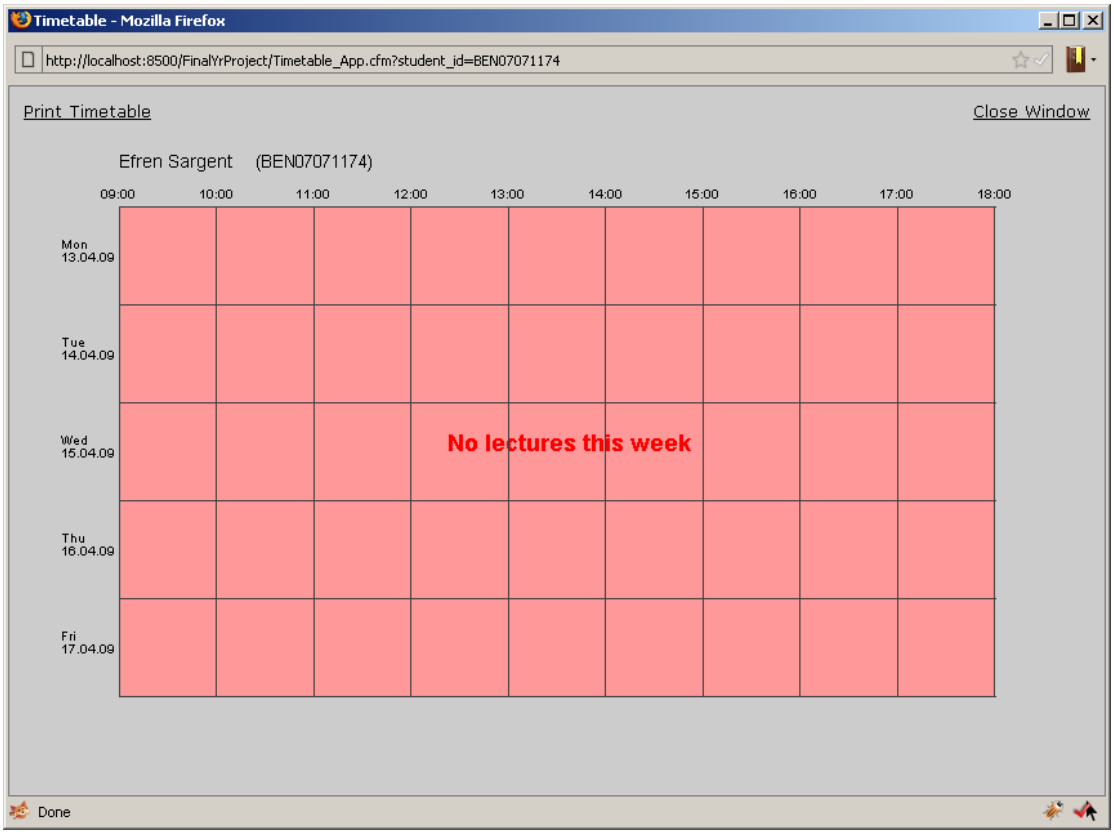


Figure 18: Error message, No lectures for the current week

SOUNDEX

As discussed in the problem analysis section of this paper Dr Mark Lee suggested using the soundex function in the staff search facility. Due to time restrictions the soundex function built into SQL Server has been used. The way the algorithm works is that the first letter of the name is kept. All occurrences of the following letters should be removed: A, E, I, O, U, W, Y and H. The remaining letters are assigned a number following the table below.

Letter	Number
B, P, F, V	1
C, S, K, G, J, Q, X, Z	2
D, T	3
L	4
M, N	5
R	6

Table 1: Soundex code table

The first four numbers form the soundex code for the name. If a name has any double letters they should be treated as one. In the name Lloyd the second 'l' should be disregarded.

Figure 19 shows part of the stored procedure that the staff search page calls when a member of staff searches for a student by name. The full stored procedure is on the data CD provided. The soundex function takes a parameter and calculates the soundex code. This stored procedure is comparing the code for the name passed as a parameter to the results that are found in the QL database. So if 'Lee' is searched for, the results consist of 'Lee Pacheco' and 'Leo Garrett'.

```
SELECT LTRIM(RTRIM(forename)) AS forename, LTRIM(RTRIM(surname)) AS surname, student_id
FROM halesowen_college.dbo.STMBIOGR AS STMBIOGR
WHERE @forename = STMBIOGR.forename
AND @surname = STMBIOGR.surname
OR soundex(@forename) = soundex(STMBIOGR.forename)
OR soundex(@surname) = soundex(STMBIOGR.surname)
ORDER BY
CASE WHEN @sortColumn ='student_id' THEN student_id END ASC,
CASE WHEN @sortColumn ='surname' THEN surname END ASC,
CASE WHEN @sortColumn ='forename' THEN forename END ASC
```

Figure 19: Stored Procedure using Soundex function

SECURITY

Requirement 6 states that a student should not be able to be access another student's timetable. This is carried out through the use of ColdFusion session variables. At the top of an Intranet page within the code is CFML setting session variables. Figure 20 shows an example for a students Intranet page. Their student ID is set in line 3 and the staff variable is set to null in line 4. These session variables are then accessible by other ColdFusion pages.

```
1 <!--- SESSION variables that can be accessed by the client from other CF pages
2         for the timeout duration specified on the server --->
3 <cfset SESSION.studentid = "dee07068776">
4 <cfset SESSION.staff = "null">
```

Figure 20: ColdFusion SESSION's set

Figure 21 shows the CFML surrounding the HTML that creates the link to the timetable. Conditional `<cfif>` statements are used to determine whether a link should be provided to the staff search page or direct to the timetable depending on the value of session.staff.

```
585     <cfif SESSION.staff eq "null">
586     <li><a class="leftmenusmall" href="JavaScript:popUpWindow('Timetable_App.cfm')">Timetable</a></li>
587     <cfelse>
588     <li><a class="leftmenusmall" href="Timetable_Search.cfm">Timetable</a></li>
589     </cfif>
```

Figure 21: ColdFusion SESSION used to determine link

```
58 <td align="center">
59     <cfapplet appletSource="timetable" name="timetable" studentid=#SESSION.studentid# staff=#SESSION.staff#>
60 </td>
```

Figure 22: ColdFusion SESSION passed as parameter

In the ColdFusion page that embeds the timetable applet the CFML pulls the student ID from the `session.studentid` variable and passes it to the applet as a parameter, the applet then uses it to gather the data from GenerateData.cfm. This prevents a student from having any input into the system. The session variables are stored server side and are processed by the ColdFusion application server.

5. TESTING

Sommerville's three stage testing process has been used throughout the development of the timetable application. These are component, system and acceptance testing (Sommerville, 2004). Each of which are discussed below, it is also shown how they are applicable to the timetable application.

COMPONENT TESTING

This type of tested is carried out during implementation on individual components. When a method is written it is tested to make sure it completes what it is intended to. All methods are tested independently without other methods. This type of testing has been carried out throughout and can be seen commented in parts of the code provided on the data CD. A basic example of this involves the `stringFirstUppercase()` from the `Process` class. It is used to convert a String that is all uppercase to a String that only has the first letter uppercase and the remaining lowercase. This is because SQL Server stores student's surnames in uppercase. A main method has been written in the `Process` class that creates a `Student` object and sets its surname to uppercase. It then calls the `stringFirstUppercase()` and passes the student surname. The result is printed out. If it is in the correct format the method works, and will work on any other String that is passed to it uppercase.

This type of testing has been done throughout the build of the project and testing can be seen commented out among the code.

SYSTEM TESTING

System testing is where components that have been tested independently are brought together to test if they function as expected. This type of testing highlights interaction errors between methods in the code. It is also used to check whether a system meets its functional and non-functional requirements. In very large applications, subsystems can be built and tested that consist of individual components. When the subsystems are tested they can be brought together to form the final system.

A test plan is included in appendix F that shows the testing of the system to validate whether the functional and non-functional requirements are met.

ACCEPTANCE TESTING

The final stage in testing is acceptance testing. This is where a system is tested with data supplied by the client rather than data generated by the developer. The author was supplied with test data from the beginning to build the system around. System and component testing was carried out with this data.

Acceptance testing continues until the system has the clients desired output. The system can then be accepted by the client. Appendix G shows a letter from the client saying that they are happy with the final build. This verifies that testing has been a success and that the clients' requirements are met.

6. PROJECT MANAGEMENT

SOFTWARE PROCESS MODEL

The software process model followed throughout the course of writing the timetable application was the iterative model. This model breaks the specification, design and implementation down into a series of increments that are developed in turn (Sommerville, 2004). This allowed part of the timetable application to be built and presented to the users at Halesowen College, the specification can then be tweaked to suit their needs and changes re-implemented in the next increment until it is accepted by the client. The first increments developed acted as a prototype that meet the users needs, this enables the user to see what they defined in the beginning and allow them to rethink and re-specify any changes. There are two other models that could have been followed, the first being the waterfall model, which requires a set a detailed requirements before any development can commence. Any changes to the requirements require rework of requirements, design and implementation (Sommerville, 2004). With the incremental model, any changes that are to be made to requirements can be reflected in development of the next increment. The second alternative model is the evolutionary model. Sommerville (2004) outlines that following the evolutionary model though can lead to unstructured code that is difficult to maintain due to allowing the requirements and design decisions being delayed.

STAKEHOLDER MANAGEMENT

A stakeholder is a person or a group that will be affected by a system, directly or indirectly (Sommerville, 2004). For this project the stakeholders are as follows:

- Client at Halesowen College (also representing staff)
- Author (also representing students at Halesowen College)
- Project Supervisor (Dr Mark Lee)

It is the author responsibility to arrange regular meetings with the client and the project supervisor to discuss and demonstrate progress. Regular meetings with the client allow each increment of the development to be demonstrated and accepted.

If the completed software were to be integrated into Halesowen Colleges system and used, the stakeholders would be as follows:

- Lecturers
- A-Level students
- Network administrator
- Database administrator

TIME MANAGEMENT

It was vital that the time spent on the project was managed carefully. A work breakdown structure was produced which shows a high level view of what needed to be completed. In each of the increment breakdowns a requirements and design review take place. The report details the final requirements and design that were followed. The work breakdown structure is shown under appendix H. To further manage the time available a Gantt chart was produced which is provided in Appendix I. This Gantt chart shows high level milestones that needed to be met to ensure the project was finished on time. Supervisor Meetings held on a regular basis ensured that too much time was not spent on unnecessary parts of the project. They also allowed for constructive criticism and feedback to be gained.

RISK ASSESSMENT

Table 2 below shows the risks that were identified, the probability of them occurring, and the impact they will have on the project. The overall risk score is the probability multiplied by the impact. This gives an overall indication of the impact of the risk.

Identifier	Risk Description	Probability (1=low, 5 = high)	Impact (1 = low,5 = high)	Overall risk score
1	Developer (Author) may encounter illness or injury causing the project progress to decline	2	4	8
2	Data could be lost during the course of development	3	5	15
3	Other work commitments could interfere with the progress of the project	3	3	9
4	Learning new technologies could take longer than originally expected	3	2	6
5	Lack of communication with project supervisor leads to project loosing focus	2	3	6
6	Lack of communication with client results in software not meeting user requirements	2	4	8
7	Underestimated timescales leads to project falling behind schedule	2	3	6

Table 2: Risk Assessment

RISK MANAGEMENT

Table 3 below shows the strategies that have been put into place to handle one of risks shown in Table 2.

Risk	Risk Strategy	Strategy Type
1	• Allow flexibility in time plan for the risk	Risk Acceptance
2	• Copies of the project must be stored on mediums other than the place of development. • Use subversion to introduce a central storage repository	Risk Avoidance
3	• Effective time management and prioritisation will reduce the effects of this occurring	Risk Reduction (Probability)
4	• Intense research into this area will give insight to which parts of the technologies need to be learnt • Highlight alternatives in research in the event the technologies result in being to problematic	Risk Reduction (Impact) Contingency Plan
5	• Weekly meetings with supervisor will allow collaboration about progress made and the planned development	Risk Avoidance
6	• Regular meetings with the client will allow feedback to be gained from progress made • Where meetings cannot take place collaboration through email will be used	Risk Avoidance
7	• Being realistic when time management is planned, will not lead to underestimating tasks • Flexibility in the time management will allow for some fallbacks • Appropriate research will give the author more knowledge about what is being undertaken	Risk Reduction (Probability) Risk Acceptance Risk Acceptance

Table 3: Risk Management

APPRAISAL

SPECIFICATION ANALYSIS

The requirements definition specified in the problem analysis section has been followed throughout the whole process of the project. The design and implementation were carried out to meet this requirements definition as accurately as possible. It was also referred back to regularly to make sure the process was not loosing scope.

The timetable application works well for A-Level students. All of the A-Level are displayed on the timetable in a simple to view way. It does not work so well for students taking other courses such as a BTEC. The lectures display fine on the grid, but they do not have any colouring. This is due to way CELCAT *Timetabler* names the events. A BTEC is just one course broken down into modules, these modules are given the course name in CELCAT Timetabler, so when the timetable application is checking if a new colour is to be used, it is right to stay with the same one.

Timetables change from week to week. The timetable application takes account of this and produces a timetable showing only lectures that occur on the week the timetable is viewed. During a weekend the next week's timetable is shown. In problem analysis a feature that was discussed with Dr Mark Lee was to display information on bank holidays etc. This would make it clear why lectures were not on that day. This was not implemented due to time constraints. It would be nice future enhancement. The implementation of the functional requirements was prioritised.

The timetable application has been implemented to be accessed from the college Intranet and to work with Halesowen College's current system setup. This was focused on throughout the project and has been a success, although it has not been tested live on the college system. In theory the timetable application should be able to be installed at the college and work without any problems. ColdFusion and SQL Server were used, and the data within the testing versions of the databases was not changed. Because this has not been tested, requirement 17 from the non-functional requirements cannot be validated. This is due to time constraints, by project completion there was not enough time to carry out this test.

A screenshot of the staff search facility can be seen in appendix D. From the screenshot it can be seen that the functionality outlined in requirement 4 has been met. The soundex function built into SQL Server allowed the author to return phonetically matching student names. The use of ColdFusion session variables ensures only staff have access to this search facility ensuring that only staff can view can all students timetables. The staff search facility was designed to be consistent with the Intranet the college already have. The importance of this is discussed in the design section. The layout and colours consistently flow with the rest of the design. This validates requirement 12. Having the page consistent with the Intranet will increase usability due to familiarity.

Efficient use of ColdFusion Session variables resulted in a system that does not require a user to log in, meeting requirement 5. They also proved as an effective method for redirecting a user to the relevant page when they request a timetable without their knowledge, meeting requirement 3. Members of staff are directed to the search page and students are shown their timetable. Sessions also allowed the author to restrict a student from gaining access to another student's timetable, or the staff search facility.

Provided in the appendix is a printout of one of the timetables printed by the timetable application. This feature turned out better than expected. It was difficult to get the positioning correct for an A4 page, but in the end the final printout looks very professional and acceptable to the client. The main problem discussed in the implementation is fitting text within a lecture. As can be seen on the printout it has been handled well.

Requirement 10 states that the application must be resizable. This proved to be more difficult than originally anticipated due to an applet being used. When an applet is embedded within a webpage it must be supplied with a size. A way to adjust the applet when the window is resized could not be found. The applet itself remains static on a webpage. The contents of the applet pick up the dimensions of the containing window and redraw the timetable accordingly. This is acceptable when the window is minimized. When the window size is increased greater than the applet size the timetable draws outside the borders of the applet, which does not look nice. To get around this a piece of JavaScript was written to disable resizing on that page. This works in Microsoft Internet Explorer, but not in Mozilla Firefox. The browsers settings need to be changed to allow JavaScript to override the resize functionality. This is not an option. The way the applet works in Firefox is that it redraws the timetable if the window dimensions are smaller than the applet, but it does not redraw any bigger than the applet.

The Java Applet that was written can be displayed on many platforms, including the ones in appendix B which are a requirement. Using a Java Applet means the colleges IT Support team do not need to worry about updating the browsers; so long as Java is enabled the applet will still function.

PERFORMANCE

The way that the system generates a visual timetable from just XML works well in terms of performance. Considering the applet is redrawing the timetable on every refresh instantiated by the operating system and the user, it does not lag or flicker. This is due to only simple shapes that are not memory intensive being used for generation of the timetable. The application would be slower if components such as JLabels were used. The initial loading of the applet is dependant on the client machine it is run on, this is because applets are downloaded and run on the client side. If the applet is accessed over a slow network, it will take longer to initially appear.

It is difficult to test the efficiency of the stored procedure for the staff search as Halesowen College supplied test data for only 1000 student. The system worked fine with 1000 students. Every search worked correctly, and results were shown instantly.

The CELCAT *Timetabler* database was provided in full. The CT_EVENT table which holds every event consists of just above 11000 records. This did not slow down the query at all. SQL Query Analyser returns that the cumulative client processing time is 15 milliseconds for the stored procedure used to gather the events from a given student id. This is good performance. The CT_EVENT table will stay around this size so performance of the query will not be affected in future.

ROBUSTNESS

The timetable application has been designed and implemented to produce a visual timetable given the lecture information in the form of an XML document. The structure of the XML is strict. Any unexpected If an XML document that does not match what the timetable applet is expecting the SAX parser will fail. It is safe to assume that the XML document will be correct, due to the GenerateData.cfm page creating it. During development no problems have been found with the parsing of data and creation of XML.

The timetable application is not portable. It would not run on another colleges system unless the data is stored in exactly the same structure as Halesowen Colleges. It has been built specifically to work with the way they store their data. If the structure of either CELCAT *Timetabler* or QL changes then the stored procedure gathering the data would need to be rewritten.

CLIENT SATISFACTION

Appendix G shows a copy of a letter from the client at Halesowen College that has aided the author throughout the project. He has acted as an end user of the system and provided me with efficient

feedback at each increment of the application that was presented to him. The letter outlines what he likes and dislikes about the timetable application. Overall the client is satisfied with the final result and how well it meets its requirements.

7. CONCLUSION

PROJECT ACHIEVEMENTS

The final system built provides a solution to the main problem outlined at the beginning of the project. The test plan shows that it functions as intended and the specification analysis in the appraisal shows how it compares to the requirements definition. The acceptance letter received from the client at Halesowen College shows that they are satisfied with the result. The GUI quality has been built to a high standard. This shows especially in the staff search facility. It is visually appealing yet simple and straightforward to use. This is due to the strict following of Brink, Gerlge & Wood's three GUI goals.

A three tier architecture has been implemented in which a variety of technologies have been used, some of which the author has had to teach himself at the beginning of the project. Even though some of the implementation has been based on self taught knowledge the final system is successful.

Working with a real client made the project more interesting and gave the author motivation to work on it. Knowing that the software produced could solve a real problem provided the determination to complete a more than satisfactory application.

Problems not relating to the project did affect the progress made. The main being other work commitments. Time management was difficult during periods of the project due to other work deadlines that needed to be met. Prioritisation and effective time management prevented other deadlines from being too much of a problem.

The duration of the project has been a large learning curve for the author. Two technologies that the author has never used have been self taught from scratch, these being ColdFusion and SQL Server 2005. SQL server configuration and the writing of stored procedures have been the most difficult. Ben Forta's book 'ColdFusion MX 7 Web Application Construction Kit' aided greatly in learning everything needed in ColdFusion, and would be recommended to any reader wishing to gain any insight into this subject.

LIMITATIONS TO THE SOFTWARE AND FUTURE ENHANCEMENTS

It still remains to be seen whether the system would actually work within Halesowen College. The author has enough confidence in the project implementation to say that it would. This is because the project specification was put together with the intention of getting the application working at the college. The implementation has met all of the requirements relating to the colleges systems; therefore it is expected to work. This is not necessarily an enhancement, but given more time it could be seen whether the application works at the college.

The timetable is focused to A-Level students. The system does work for students on other courses but the formatting may not appear correctly. An example for students on BTEC courses is explained in the appraisal. Extending the application to work with all courses would widen the applications audience.

Night classes are a problem for the timetable application because the grid is only drawn till 6 pm. If a student has a night class this lecture will still be shown on the timetable but it will be outside the grid, or outside the applet screen. Given more time the applet could be extended to dynamically draw the grid depending on the lectures that need to be displayed.

Currently when a user prints a timetable it automatically prints to A4, a way of getting the applet to generate timetables for different paper sizes could be implemented.

Discussions with Dr Mark Lee resulted in the idea of showing bank holiday information on the timetable. This would inform the user why no lectures are shown for particular days. Again, due to time constraints this was not implemented. It was an additional idea so priority was given to the initial specification. This feature would increase the user-friendliness of the system. The idea can be extended to show specific dates at the college such as training days and trips, as opposed to just national bank holidays. This will also increase the applications uniqueness to the college.

There are so many enhancements and additional features that could be made like the ones above. More functionality could be given to students, such as the ability to put their own appointments onto the timetables. The building blocks for a much larger system now exist.

8. BIBLIOGRAPHY

Black, P. E. (2007, December 17). *Dictionary of Algorithms and Data Structures.* Retrieved April 10, 2009, from U.S. National Institute of Standards and Technology: http://www.itl.nist.gov/div897/sqg/dads/HTML/soundex.html

Brinck, T., Gergle, D., & Wood, S. D. (2002). *Usability for the Web.* San Francisco: Morgan Kaufmann Publishers.

Brownell, D. (2002). *SAX2.* Sebastopol: O'Reilly.

CELCAT. (2008). *CELCAT online: Timetabler Overview*. Retrieved April 2, 2009, from CELCAT online: http://www.celcat.com/products/timetabler/timetabler.html

Connolly, T. M., & Begg, C. E. (2004). *Database Systems: A Practical Approach to Design, Implementation and Management.* Addison Wesley.

Cornell, G., & Horstmann, C. S. (2003). *Core Java 2, Volume 1 - Funtamentals.* California: Sun Microsystems Press.

Eckstein, R., Loy, M., & Wood, D. (1998). *Java Swing.* California: O'Reilly.

Forta, B., & Weiss, N. (2003). *Macromedia Coldfusion MX Web application construction kit.* Berkeley: Macromedia Press.

Galitz, W. O. (2002). *The Essential Guide to User Interface Design.* New York: Wiley Computer Pub.
Gunderloy, M., Jorden, J. L., & Tschanz, D. W. (2006). *Mastering Microsoft SQL Server 2005.* Indianapolis: Sybex.

Harold, E. R., & Means, S. W. (2001). *XML in a Nutshell.* Sebastopol: O'Reilly.

Holzner, S. (2008). *PHP: The Complete Reference.* McGraw-Hill Osborne.

Knudsen, J. (1999). *Java 2D Graphics.* O'Reilly.

Ladd, E. (2002). *Macromedia ColdFusion MX development.* Indianapolis: Que.

Lerdor, R., Tatroe, K., & MacIntyre, P. (2006). *Programming PHP.* Cambridge: O'Reilly.

London, I. C. (2007, December 10). *Central Timetabling and Survey Unit*. Retrieved April 2, 2009, from Imperial College London: http://www3.imperial.ac.uk/centraltimetabling/aboutctsu/aboutcelcat#MyTimetable_student

Microsoft. (2006, January 1). *Application Server: Frequently Asked Questions*. Retrieved April 12, 2009, from Microsoft: Windows Server: http://www.microsoft.com/windowsserver2003/techinfo/overview/appservfaq.mspx#EBB

PHP. (2009, April 3). *PHP: General Information*. Retrieved April 4, 2009, from PHP: http://uk.php.net/manual/en/faq.general.php#faq.general.differences-45

Ramakrishnan, R., & Gehrke, J. (2002). *Database Management Systems.* McGraw Hill Professional.

Sommerville, I. (2004). *Software Engineering.* Essex: Pearson Education Limited.

Systems, A. (2006). *Agresso QL Students Overview.* Retrieved April 3, 2009, from Agresso: http://www.distinction-systems.co.uk/docs/pdf/QLS_general/QLS%20FE%20overview.pdf

Worsley, J. C., & Drake, J. D. (2002). *Practical PostgreSQL.* O'Reilly.

9. APPENDICES

A. DATA CD STRUCTURE

The data CD provided contains all of the source code written by the author.

Example_Timetable_Printout.pdf is an example of the printout that the timetable application generates for a student ID

Dissertation contains electronic copies of the report as docx and pdf file formats

Java_Source_Code
Contains 6 java files, these are the files that build the Java Applet.

CFusion_Server
Contains a folder called **FinalYrProject**. This folder contains all of the files and folders handled by the ColdFusion Server. The list below details them:

Folder **_mmServerScripts** contains files that the server uses.
Folder **Halesowen College - Intranet_files** contains images that are displayed on the Intranet page.
Application.cfm is a file that stores global settings for the FinalYrProject root of the server.
clear_sessions.cfm deletes SESSION variables staff and student when run.
GenerateData.cfm holds the code that produces an XML document

These next three are the same page intranet page but they set different SESSION variables to simulate users logging in.
Halesowen College - Intranet_ben07071174.cfm
Halesowen College - Intranet_dee07068776.cfm
Halesowen College – Intranet_staff.cfm
IntranetIndex.cfm is a list of links to the above three pages to aid the user.
infoBanner.jpg, script.js, timetableStyle.css are all pages that the web pages access.
timetable.jar is a JAR file containing all source relating to the Java Applet.
Timetable_App.cfm – this is the page that the Java Applet is embedded within.
Timetable_Search.cfm – this is the staff search facility page.

SQL_Server_2005
Contains a detailed database schema spreadsheet, and a folder **Stored_Procedures** that contains text files showing the stored procedures that should exist in the SQL server for data gathering.

To run the code, ColdFusion MX and SQL Server 2005 must first be installed.

The FinalYrProject folder within the CFusion_Server folder must be copied to the wwwroot folder of the server directory.
In the ColdFusion administrator two data sources must be created linking the ColdFusion server to the SQL Server databases.
The Java Applet also needs to be added so it can be referred to by name in the ColdFusion code.

The SQL Server 2005 databases must be created with the readers own data following the database schema supplied in the folder SQL_Server_2005. This is due to Halesowen College not allowing the data to be passed on due to data protection laws.
The stored procedures provided as text files must be created in the stored procedures folder of the offline_celcat database.
A new user role must be created called FYP with SELECT only access to the two databases.

The Intranet pages can then be opened and the timetable link clicked.

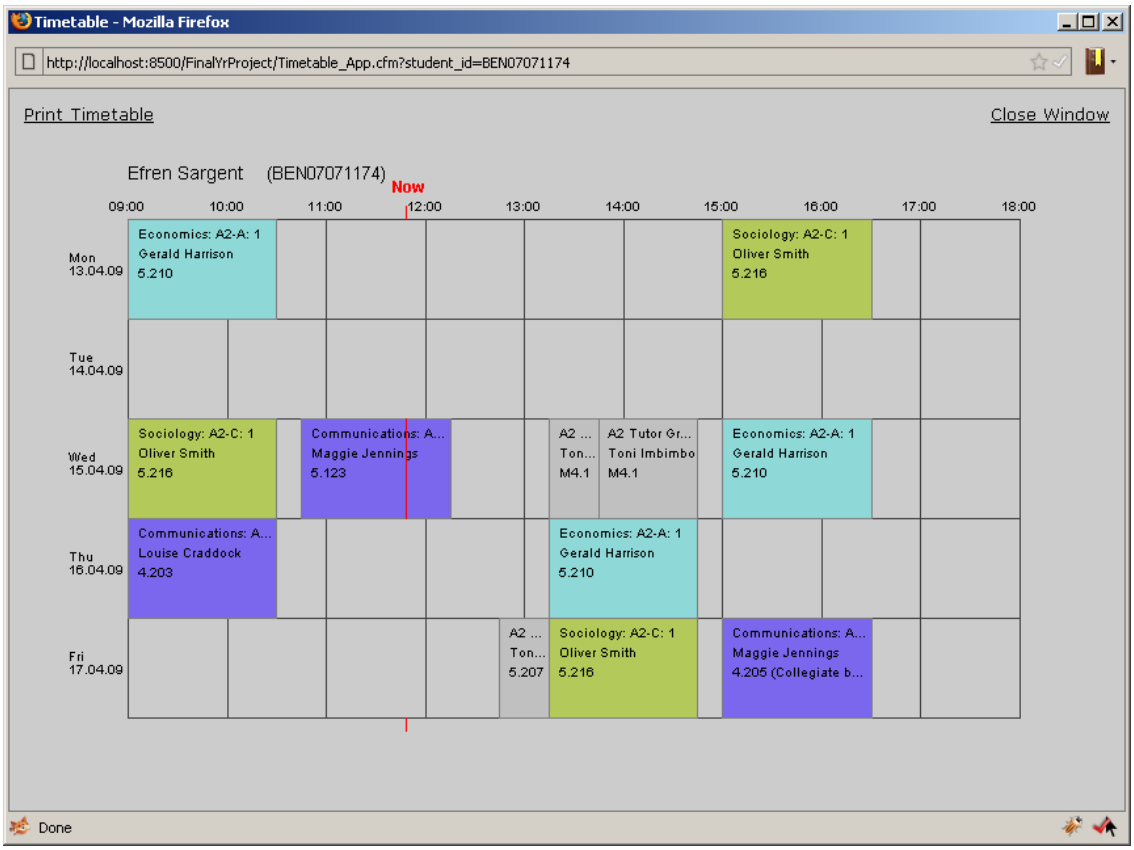
B. BROWSERS

This is a list of the browsers that the timetable application must be viewable within:

- Microsoft Internet Explorer 7
- Mozilla Firefox 3
- Apple Safari 3.1.1

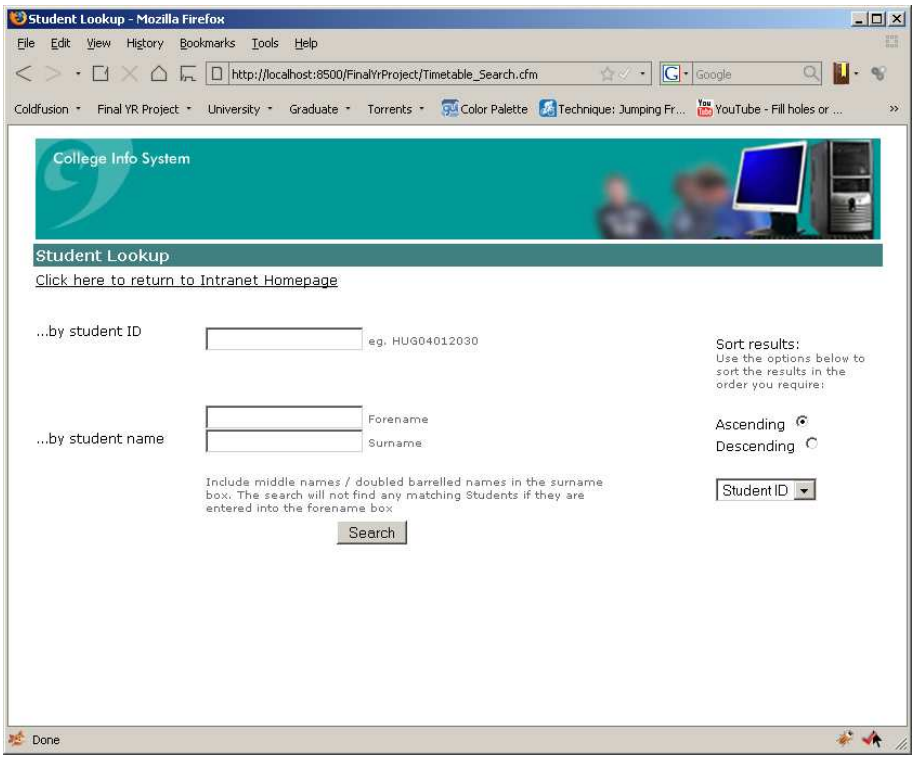
C. TIMETABLE SCREENSHOT

The screenshot below is an example of a timetable that the final build of the timetable application produces. It shows the here now feature and how the text is trimmed to fit within lectures.

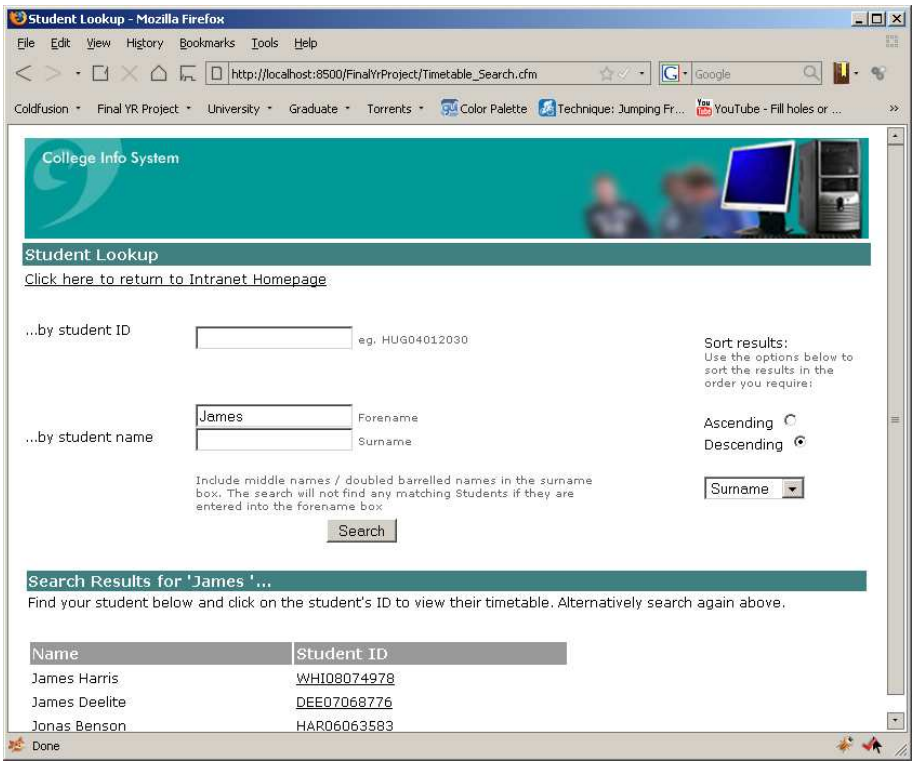


D. SEARCH FACILITY SCREENSHOT

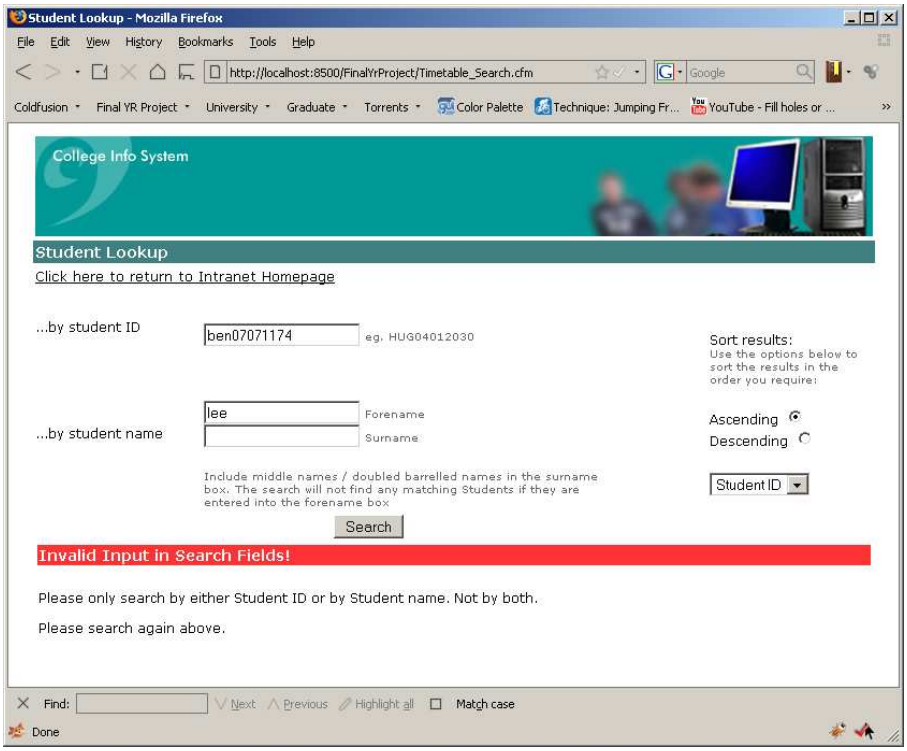
The screenshot below shows the staff search facility.



The screenshot below shows how results are displayed in the staff search facility with the sorting options selected.



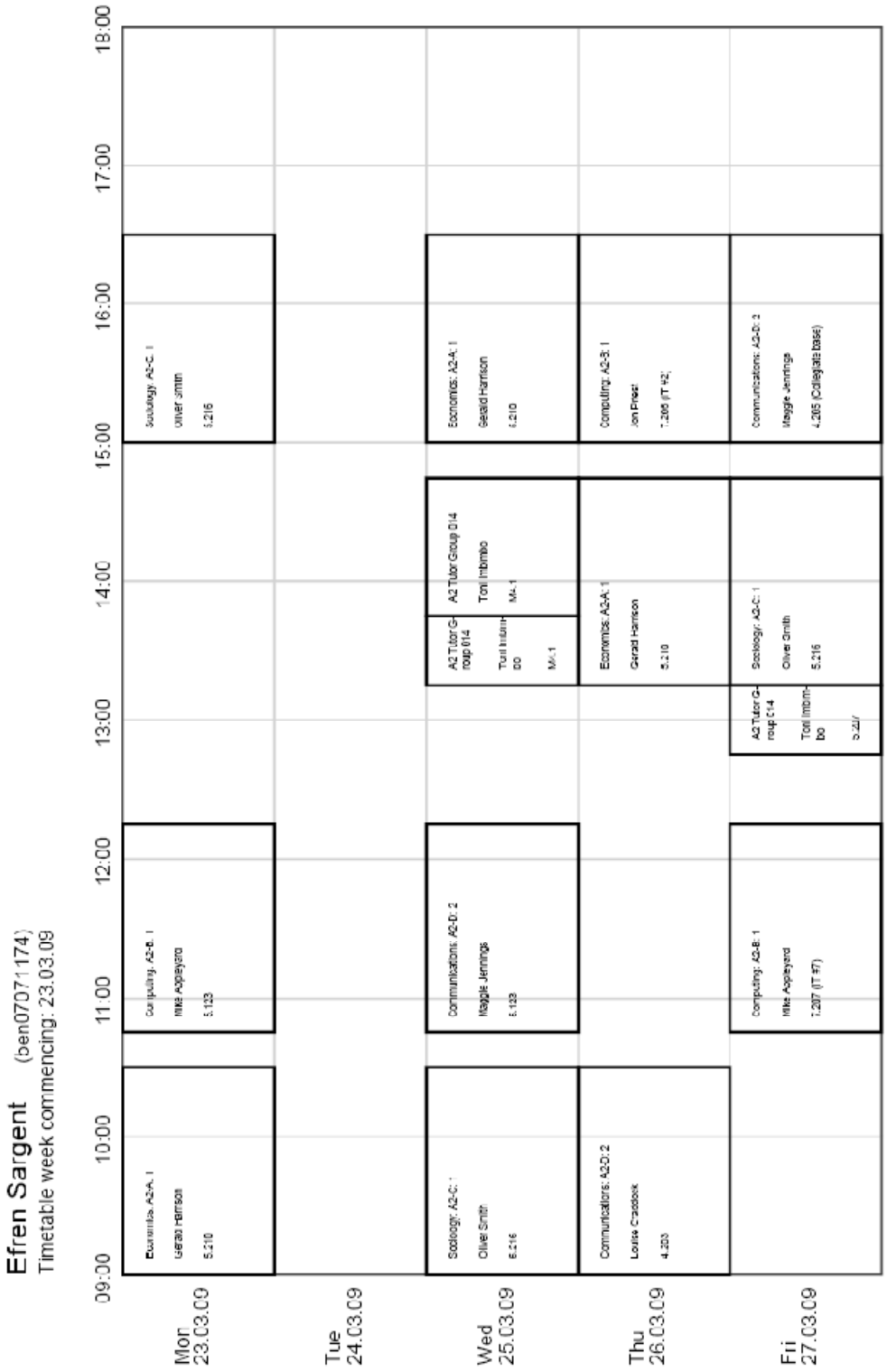
Screenshot below shows the error message that is displayed when a user attempts to search by a student ID and a name.



E. PRINTOUT

A PDF version of the full timetable is available on the data CD titled Example_Timetable_Printout.pdf.

The image below is a screenshot of the pdf to give the reader an idea of what it looks like and how the text is formatted to fit within a lecture.



F. TEST PLAN

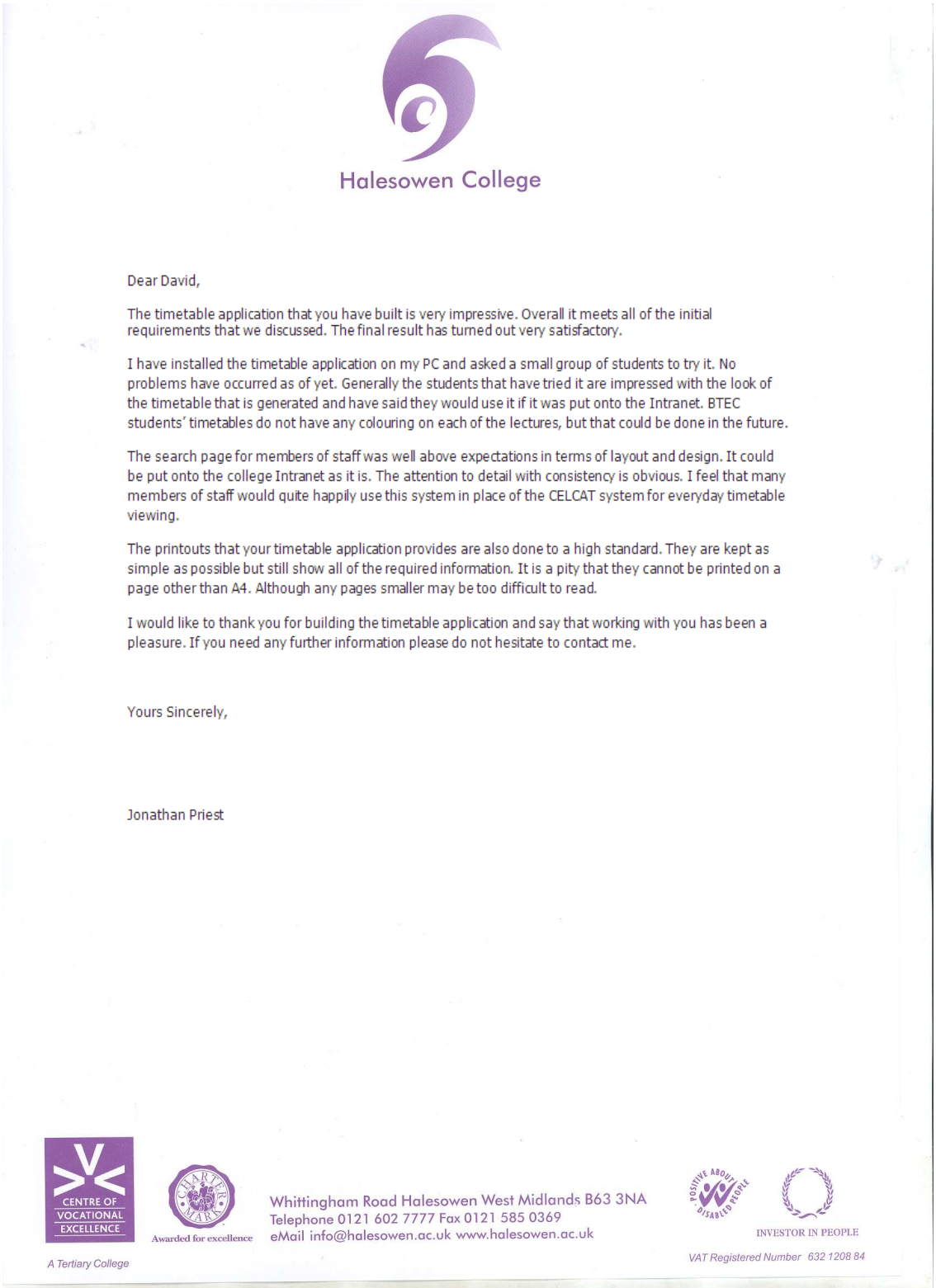
The test plan below tests the final build against the functional and non-functional requirements.

Req.	Validation Criteria/Expected Results	Pass?	Comments
1	One timetable should be shown onscreen. All lectures should be shown for a course that the student is enrolled upon.	PASS PASS	For any given student ID all of the correct lectures are gathered from the databases and shown correctly.
2	A timetable should only show lectures for the current week. The timetable should only show Monday – Friday. Test on Saturday or Sunday, the next weeks lectures should be shown.	PASS PASS PASS	The student ID 'ben07071174' has a differing timetable on some weeks. This information is shown correctly on the timetable. The weekend returns the next weeks lectures.
3	A link on the Intranet should take the user to the timetable. Students directed to their timetable. Staff directed to the staff search page.	PASS PASS PASS	ColdFusion application server works correctly. Conditional processing carried out to determine where to direct a user. SESSION variables also work as expected.
4	HTML forms must allow a user to search by: Student ID – 'ben07071174' should return 'Efren Sargent' Forename – 'Lee' should return two students Surname – 'Smith' should return one student Sort options: Ascending / descending should order results by any column User should not be able to search by student ID and a name field at the same time.	 PASS PASS PASS PASS PASS	Student ID returns the correct student. Forename returns the expected two results and so does the surname. Informative error message displayed when too many input fields are filled in. shown in appendix D.
5	User should not have to log in	PASS	SESSION variables are working. Application server knows if a staff or student is accessing the timetable.
6	When a student is logged they should have no way of accessing another students timetable	PASS	If a parameter is passed through the URL the application ignores it. No way exists for a student to gain access. All processing done in background.
7	The staff search page should return any student searched for and provide a link to their timetable	PASS	The search page queries the QL database, as long as the student exists in there then staff will be able to find them

8	A red line should indicate the current day and time when a timetable is viewed.	PASS	The red line updates throughout the day and shows only on the days specified in requirement 2.1
9	When a timetable is viewed it should have a link to print it. The timetable should not be printable if no lectures are displayed	PASS PASS	The timetable does not print if no lectures exist. The formatting is easily readable. Does not provide a way to print on paper sizes other than A4
10	The timetable must resize when a user resizes the browser window.	FAIL	The resizing of the applet was problematic. In Firefox it can be made smaller. This is discussed in the appraisal section.
11	The application must be viewable on many screen resolutions	PASS	When the screen resolution is large the timetable can take up most of the screen. Due to the problems with resizing it cannot be minimised but it is still viewable.
12	A user should be able to use the application within 10minutes.	PASS	A student only has to click on one to access their timetable. Staff build familiarity with the search page quickly as it is consistent with the Intranet.
13/14	The timetable should appear within 10 seconds of clicking view	PASS	Any timetable that has been viewed has appeared within 10 seconds. This should always be the case as the generation of the timetable in the applet is done client side. The server is just used mainly for data access.
15	System should be available for users 95% of the day. If implementation at the college, its availability time will need to be monitored for the first month.	-	This cannot be tested until it is running at the college
16	The application should show a timetable when it is opened on all of the browsers in appendix B.	PASS	All display as expected
17	Seamlessly integrate into Halesowen Colleges current setup	-	This has not be tested, but assumed to do so because it has been built according the clients specification.
18	Students should not have access to any other student's timetables.	PASS	Testing of requirement 6 proved this.
19	Was the software completed by 23rd march?	PASS	
20	Enhancements cannot invalidate requirement 13	-	Cannot be tested until it is running at the college and enhancements are made.

G. ACCEPTANCE LETTER

The image below is a scanned copy of the letter that the client sent in return to the final timetable application.



Halesowen College

Dear David,

The timetable application that you have built is very impressive. Overall it meets all of the initial requirements that we discussed. The final result has turned out very satisfactory.

I have installed the timetable application on my PC and asked a small group of students to try it. No problems have occurred as of yet. Generally the students that have tried it are impressed with the look of the timetable that is generated and have said they would use it if it was put onto the Intranet. BTEC students' timetables do not have any colouring on each of the lectures, but that could be done in the future.

The search page for members of staff was well above expectations in terms of layout and design. It could be put onto the college Intranet as it is. The attention to detail with consistency is obvious. I feel that many members of staff would quite happily use this system in place of the CELCAT system for everyday timetable viewing.

The printouts that your timetable application provides are also done to a high standard. They are kept as simple as possible but still show all of the required information. It is a pity that they cannot be printed on a page other than A4. Although any pages smaller may be too difficult to read.

I would like to thank you for building the timetable application and say that working with you has been a pleasure. If you need any further information please do not hesitate to contact me.

Yours Sincerely,

Jonathan Priest

CENTRE OF VOCATIONAL EXCELLENCE

A Tertiary College

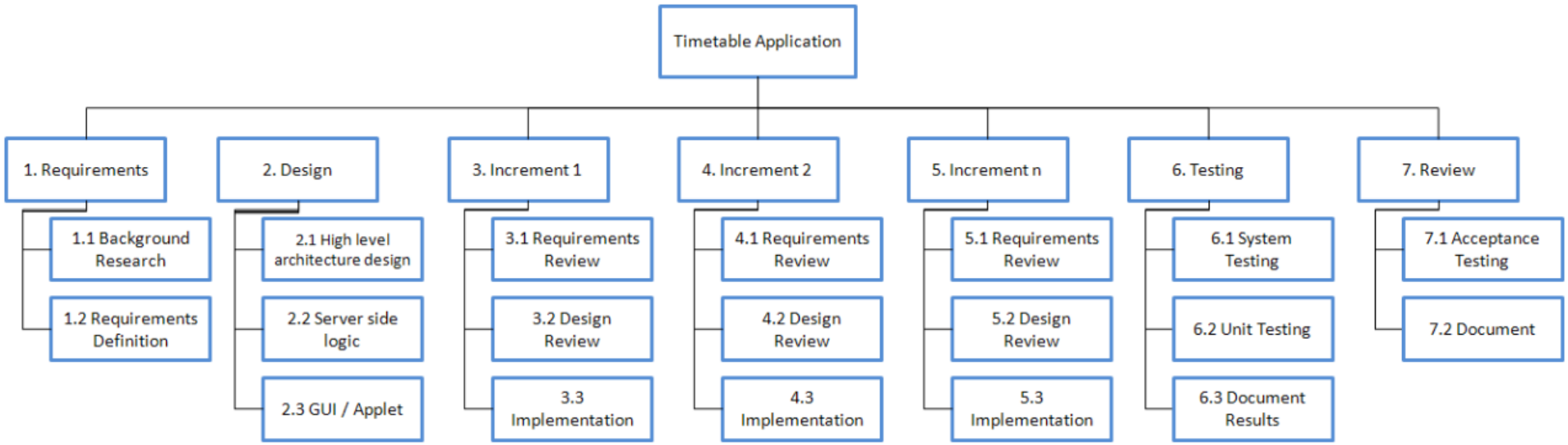
Awarded for excellence

Whittingham Road Halesowen West Midlands B63 3NA
Telephone 0121 602 7777 Fax 0121 585 0369
eMail info@halesowen.ac.uk www.halesowen.ac.uk

INVESTOR IN PEOPLE

VAT Registered Number 632 1208 84

H. WORK BREAKDOWN STRUCTURE



I. GANTT CHART

	29/09/2008	06/10/2008	13/10/2008	20/10/2008	27/10/2008	03/11/2008	10/11/2008	17/11/2008	24/11/2008	01/12/2008	08/12/2008	15/12/2008	22/12/2008	29/12/2008	05/01/2009	12/01/2009	19/01/2009	26/01/2009	02/02/2009	09/02/2009	16/02/2009	23/02/2009	02/03/2009	09/03/2009	16/03/2009	23/03/2009
Research into technologies																										
Planning																										
Teach self ColdFusion and SQL Server																										
Understand database structures																										
Design																										
Implementation																										
Install and configure ColdFusion and SQL Server																										
Import databases																										
Stored procedures																										
ColdFusion pages to collect the data and produce xml																										
Java Applet																										
Parser for XML																										
Data structure																										
Processing of data																										
drawing the grid and labels																										
drawing the lectures																										
here now feature																										
printing																										
Staff search ColdFusion page																										
SESSION variables for security																										
Testing																										
Review project																										
Project presentation																										

Indicates the whole duration of a high level task

Indicates the duration for tasks