



DEPARTMENT OF COMPUTER SCIENCE

## IEP: Interactive Examination Program

James William Burke

---

A dissertation submitted to the University of Bristol in accordance with the requirements  
of the degree of Bachelor of Science in the Faculty of Engineering

---

May 2006 | CSBSC-06

# Declaration

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Bachelor of Science in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

James William Burke, May 2006

## **Abstract**

Over the past decade, technology has grown at a rapid rate in many sectors of business. Daily activities such as banking, shopping and business transactions have become computerised, enabling secure transfer of information across the internet.

The education sector can be described as a business; an ever changing corporation requiring continual documentation and updating of information. As this business grows, so do the demands on the administration systems that form its backbone. An overload of paperwork and an increase in the numbers of admin staff required to combat such problems are two perfect reasons for the incorporation of an automated examination system for educational environments. Time, money and effort could be saved if students were able to sit exams in a computerised environment. Many businesses that are not strictly educational, yet offer some form of examination procedures have adopted this new way of working. The DVLA Driver Theory test for example is now completely computerised, and has been working successfully now for several years.

This thesis outlines the work carried out to produce a complete online examination system for use in a University environment. The finished product, IEP, is a secure, user friendly system that enables lecturers and students alike to participate in and monitor examination progress, minus the usual paperwork

# Table of Contents

Abstract .....	3
Table of Contents .....	4
Table of Figures.....	7
1. Introduction .....	8
1.1 Tasks Involved.....	8
1.1.1 Research.....	8
1.1.2 Design .....	8
1.1.3 Implementation .....	8
1.1.4 Testing .....	9
2 Background.....	10
2.1 Application Research.....	10
2.1.1 Blackboard .....	10
2.1.2 Richard Healy Test Builder .....	10
2.1.3 Other Professional Systems.....	10
2.1.3.1 ExamWeb.....	10
2.2 Security .....	11
2.2.1 SSH (Secure Shell) .....	11
2.2.2 IP Monitoring .....	11
2.2.3 On the day Passwords.....	11
2.2.4 Internet Access .....	11
2.2.5 HyperText Transfer Protocol Secure .....	11
2.2.6 SQL Injection Attack .....	12
2.3 Possible Approaches .....	12
2.3.1 Available Technology .....	12
2.3.2 Web Servers.....	14
2.3.3 Internet Technologies .....	15
2.3.3.1 Web Browsers.....	15
2.3.3.1 Page Formatting.....	15
2.3.3 Conclusions of Final Software.....	15
2.4 Exam Research - Question Formats.....	16
Bloom's Taxonomy .....	16
Question Formats .....	16
3 Specification.....	18
3.3 Requirements .....	18
3.3.2 Functional Requirements .....	18
3.3.2.1 Student Section .....	18
3.3.2.2 Lecturer Section.....	18
3.3.2.3 Administration Section .....	18
3.3.4 Efficiency .....	19
3.3.5 Design Constraints .....	19
3.4 System Environment.....	20
3.5 Interface .....	21
3.6 Security .....	21
SQL Injection.....	21
Bombardment .....	21
Security Procedures .....	21
3.7 Data Transfer .....	22
3.8 Project Schedule .....	22
3.9 Class Design.....	22
3.9.4 Exam Package .....	22
3.9.5 Marking Package .....	23
3.9.6 Tasks Package .....	24

3.9.7 Security Package.....	24
3.9.8 Monitoring Package.....	24
3.9.9 Utilities Package.....	24
4 Implementation.....	25
4.1 Database.....	25
4.1.1 Structure.....	25
4.1.2 Table Relationships.....	26
4.1.3 Connection Pooling.....	26
4.1.4 Database Class.....	26
4.1.5 Database Results.....	27
4.2 Interface.....	27
4.2.1 Navigation.....	27
4.2.1.1 Lecturer Navigation.....	27
4.2.1.2 Student Navigation.....	28
4.2.2 Style.....	28
4.2.3 Layout.....	28
4.3 Calculator.....	28
4.4 Security.....	29
4.4.1 Security Realms.....	29
4.4.2 HTTPS.....	29
Server Certificate.....	29
Configuration.....	30
4.4.3 Security Validation.....	30
4.5 System Access.....	30
4.5.1 Index.....	30
4.5.2 Login.....	30
4.5.3 User JavaBean.....	31
4.5.4 Settings Bean.....	31
4.6 Student Section.....	31
4.6.1 Welcome.....	31
4.6.2 Start Exam.....	32
4.6.3 Exams.....	32
4.6.3.1 Exam Back End.....	32
4.6.1.2 Layout.....	33
4.6.1.3 Setup.....	35
4.6.1.4 Material.....	35
4.6.1.5 Answering.....	36
4.6.1.6 Confirmation.....	36
4.6.1.7 Completion.....	36
4.7 Lecturer Section.....	37
4.7.1 Exams.....	37
4.7.1.3 Exam Back End.....	37
4.7.1.4 Create Exam.....	40
4.7.1.5 View Exams.....	43
4.7.1.6 Edit Exam.....	44
4.7.1.7 Generate and Print Passwords.....	44
4.7.1.8 Monitor Exams.....	44
4.7.1.9 Marking Front End.....	45
4.7.2 Tasks.....	47
4.7.3 Settings.....	47
4.7.4 IP Ranges.....	48
4.7.4.1 IP Ranges Back End.....	48
4.7.4.2 IP Ranges Front End.....	48
5 Results.....	49
5.6 Security Testing.....	49

5.6.1	Accessing Resources .....	49
5.6.2	Secure Server .....	49
5.6.3	SQL Attacks.....	49
5.6.4	IP Checks .....	49
5.7	Student Testing .....	49
5.7.1	Interface Testing .....	50
5.7.2	Exam Class Testing.....	50
5.8	Other Tests .....	51
6.	Conclusion and future work .....	52
6.1	Achievements .....	52
6.2	Problems .....	52
6.3	Future Developments .....	52
6.4	Summary .....	52
	References .....	53

## Table of Figures

Figure 1: Communication over HTTPS with IEP.....	10
Figure 2: Communication between web browser and server .....	13
Figure 3: System Interaction.....	18
Figure 4: System Structure.....	18
Figure 5: Project Schedule .....	20
Figure 6: UML diagram for the exam package .....	21
Figure 7: Shows a basic UML class diagram for the marking package .....	21
Figure 8: Database Relationships .....	24
Figure 9: Lecturer Navigation .....	25
Figure 10: Student Navigation .....	26
Figure 11: System Layout .....	26
Figure 12: Calculator Applet .....	26
Figure 13: Login Page .....	28
Figure 14: System Access Interface.....	28
Figure 15: Main Page .....	29
Figure 16: Exams Page.....	31
Figure 17: Top Navigation.....	32
Figure 18: Drop-Down Navigation .....	32
Figure 19: Quick Navigation .....	32
Figure 20: Information Bar.....	32
Figure 21: View Material.....	33
Figure 22: Exam Confirmation .....	34
Figure 23: Completion Page .....	35
Figure 24: Stage One .....	38
Figure 25: Calendar Facility.....	38
Figure 26: Stage Two.....	39
Figure 27: Stage Three.....	40
Figure 28: Question Details.....	40
Figure 29: Exam Overview .....	41
Figure 30: Single Exam View .....	41
Figure 31: Edit Exam .....	42
Figure 32: Monitor Exams Overview .....	42
Figure 33: Monitor Single Exam .....	43
Figure 34: Marking Overview.....	43
Figure 35: Automatic Marking.....	44
Figure 35: Manual Marking.....	44
Figure 36: Marking Overview.....	44
Figure 37: Tasks Overview .....	45
Figure 39: Change Settings.....	45
Figure 40: IP Range Overview .....	46
Figure 41: Testing Efficiency Results.....	51

# 1. Introduction

Examinations can be a traumatic experience and feelings of pressure and angst can lead to loss of concentration and often time and marks. As education advances, so should the assessment procedures that are vital to any students' personal development. By creating a system that is friendly, comfortable and stress free, the future of exam procedures will be revolutionised.

This project aims to develop a system that provides a secure, safe environment for lecturers to create and host dynamic exams. Exams will allow the incorporation of multiple question formats. Whilst the exam is underway the lecturer will be able to monitor the progress of the participating students. The student will be given a set time in which to complete the exam and can monitor his or her progress through the user friendly interface designed to almost eliminate human error.

Upon completion of the exam, closed ended questions can be automatically marked thereby reducing lecturer administration time. Open response questions must be marked manually. Both manual and automatic marks are saved by the system.

The design of the system will enable both computer literate and illiterate students to sit a secure examination with ease and in comfort. The system will reduce the considerable paper and consumable consumption associated with traditional exams and in addition will consequently be more cost effective.

A reduction in the administration time of both lecturers and support staff will be further complemented by the fact that the system automatically inserts results into an easy to use database that can be exported to Extensible Markup Language (XML) format.

## 1.1 Tasks Involved

There are a number of tasks that require implementation of a successful project. Such tasks are outlined briefly below, and will be considered in more detail in later chapters.

### 1.1.1 Research

Research is an essential part in creating any successful project. A large part of the development of this project will require thorough research into all aspects of the system, with specific reference to exam types, question types and other relevant issues. I therefore plan to spend a large part researching all aspects of the system. I will start my research into different types of exams, the main elements required, different question types, and other issues relevant to exams.

Further research will be required into the available technologies essential for creating a successful system, from server, programming languages and topologies to algorithms and design methods. Next investigation into interfaces will be required, the type of interfaces users find easiest to relate to, the layout and the easiest interaction methods.

### 1.1.2 Design

The design stage involves, creating a specification of the whole system, designing the structure of classes, data flow, communication methods between sections of the project, how data will be transferred, and thoroughly planning the whole project.

### 1.1.3 Implementation

The implementation stage, will consists of building the system, keeping to the project plan and specification to ensure I have implemented every aspect as planned. As with



any project, the specification will change throughout the implementation, due to unforeseen circumstance, but I intend to analyse and thoroughly plan and redesign the system before continuing.

#### **1.1.4 Testing**

Testing will happen throughout development, and on the final product. I plan to test both front and back ends, improving and redesigning relevant sections as required in order to build a successful product.

## 2 Background

### 2.1 Application Research

At the present time, there exists very few online assessment applications. Two of the more widely used applications are outlined below.

#### 2.1.1 Blackboard

Blackboard is an academic content management system with an examination function that is used by many Universities and colleges nationwide.

Further investigation of Blackboard highlighted several drawbacks to the application, mainly with relation to usability and stability;

*"Students are likely to have problems if they ... double click the submit button ... this may result in error messages which ... cause the test not to be saved." [1].*

In addition, the reliability of Blackboard is questionable;

*"Quite a few participants have problems with the speed of access ... the reason for this is believed to be the number of students accessing the Blackboard server and overloading the system." [2]*

In an ideal world an online examination system would allow simultaneous access to many students and would be proficient enough to prevent failure with heavy usage.

#### 2.1.2 Richard Healy Test Builder

This server based exam program is a simple and effective design for minimal use unsecured exams.

*"It is designed to be an easy to use testing solution. The code is written in ASP using an Access database to store its data." [3]*

This type of system provides minimal security and hence would be ineffective for use as a large scale educational assessment tool.

#### 2.1.3 Other Professional Systems

A number of professional examination programs exist at present, but the cost of such systems run into thousands due to their exceptionally high levels of security and reliability.

##### 2.1.3.1 ExamWeb

ExamWeb is an example of a professional examination system. A large one off installation price, and a per use fee makes the system costly and impractical for educational use.

*"A stand-alone subscription sells for \$19.95 per semester." [4]*

In light of the afore-mentioned problems, continual user feedback will be obtained during the design and implementation of the system for this project.

## 2.2 Security

The security aspects pertaining to the development of an online examination system require some consideration before construction of the system can take place.

### 2.2.1 SSH (Secure Shell)

*"Secure Shell" is a program to log into another computer over a network, to execute commands in a remote machine, and to move files from one machine to another.* [5] Such systems could be used by students to log on to University computers, effectively fooling the system into thinking they are in University premises.

At present, no advertised procedures to detect such an intrusion exist, however on-the-day passwords would be an appropriate method of prevention. Passwords would be assigned from a specific location by approved personnel.

### 2.2.2 IP Monitoring

In order to ensure that examinations are undertaken in University approved premises, continual monitoring of approved IP addresses will occur in an aim to prevent illegal addresses accessing the system.

Unlike SSH, it is possible to detect an intrusion from an invalid IP, and if one is detected that does not belong to the allocated computer then access is denied.

### 2.2.3 On the day Passwords

Unique student passwords distributed on the day of the exam is a simple yet secure method of ensuring that only genuine candidates sit the exam.

### 2.2.4 Internet Access

The use of a networked computer during an examination poses a problem in that students have access to internet resources whilst under examination conditions.

One option for avoiding illegal internet access is port monitoring and restricting, but further investigation suggests that such a method would prove to be a project in itself. Blocking ports can lead to network problems, including issues with database connections, network connections and printing services.

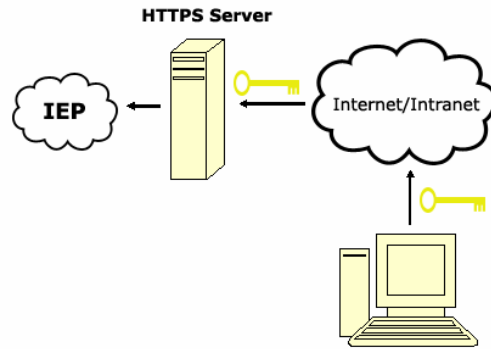
An alternative approach to this problem is the use of a shell. A shell is used to build a user environment and the creation of a custom shell would prevent access to all other programs bar IEP. Common log in information would be used at each workstation which would load the shell and prevent access to all other programs.

### 2.2.5 HyperText Transfer Protocol Secure

HyperText Transfer Protocol (HTTP) is used to transfer data throughout the World Wide Web (WWW). HyperText Transfer Protocol Secure (HTTPS) is a combination of the HTTP protocol with a secure layer known as Secure Socket Layer (SSL) which provides encryption methods to ensure secure internet communication. HTTPS is widely used throughout industry for secure communication such as payments and transactions.

Appropriate security provided by the protocol relies on accurate implementation by the administrator and encryption of the server to prevent outside interference with the service (*Figure 1*).

Figure 1: Communication over HTTPS with IEP



HTTPS allows the transmission of data securely from the server to the browser, avoiding interception during transit.

### 2.2.6 SQL Injection Attack

An SQL injection attack is one of the main security risks associated with a web application. Such an attack can occur when queries are made on the database and make the system vulnerable.

An attack can happen during any dynamic queries for example:

A simple query that checks an exam exists:

```
"select * from exams where Exam='" + examId + "'"
```

If the attacker has the option to manipulate the value in `examId`, they can potentially alter or change parts of the database. For example, if `"1'; drop table exams;"` was inserted in `examId`, then the attacker would be given free reign of the database. The IEP system will be designed with this type of attack in mind.

## 2.3 Possible Approaches

Two possible approaches to building this system exist. The first would be to build a stand alone system that communicates with a database, and the second would be to build a web based system. The standalone system will require installation of the program on every machine, whereas only internet access would be required for a web based application.

### 2.3.1 Available Technology

A review of the available technology is paramount in ensuring the most efficient system is developed. A critical analysis of several approaches therefore follows.

#### 2.3.1.1 Programming Languages

There are a number of different programming languages available for use with both standalone and web based systems.

- o C/C++  
The C programming language is widely used, *"C is distinguished for the efficiency of the code it produces, and is the most commonly used programming language for writing system software."* [6]. Whilst many programmers find the use of the software relatively simple, others find its complexity daunting.

C functions effectively with small efficient systems that do not require human interaction.

- Java/Java Server Pages  
Java is a preferred programming language for many users, because it is object orientated<sup>1</sup>. It is based on the C language syntax but personal experience proves Java to be easier to use. Java deals with memory management and handles user interaction more effectively than C.  
Java Server Pages (JSP) is Java's version of dynamic web programming. It is used for server side scripting and is extremely effective for large scale web applications. JSP is easy to maintain and is fast and efficient.
- Active Server Pages  
Active Server Pages (ASP) is Microsoft's language for creating dynamic server side scripting that incorporates the use of Visual Basic (VB).

Following investigation into these programming languages it was decided that the use of a web based application would be most effective in this system. It therefore remained to be decided whether to use ASP or JSP for the task.

ASP is a scripting language that allows scripts to be run when requested at run time, but is restricted to the Microsoft platform. Conversely, JSP is based on Java technology and allows access to the vast range of existing Java API. The page is compiled at runtime into a servlet or Java class. The benefits of Java can be exploited with JSP.

In conclusion JSP language was chosen for use in this system. As an extension of Java, complicated classes and objects can be created and in addition JSP supports Tag Libraries<sup>2</sup> and JavaBeans<sup>3</sup>.

### 2.3.1.2 Databases

An effective online examination system requires a secure and efficient database for storage of information. Various commercial databases are available and a brief overview of the most widely used follows below:

- Microsoft Access  
Microsoft Access is a commercial database application that can be used by both professional programmers and beginners alike. Access is suitable for systems with limited scalability and it performs poorly when used with a high volume of clients.

*"MS-Access databases are slow and inefficient and bog down when multiple people try to use them." [7]*

As Access is a Microsoft product it cannot be used in conjunction with non-Microsoft products such as JSP.

- MySQL  
MySQL is an open source database management product with a simplistic setup. It functions using a command line interface and is favoured by web programmers for its flexibility in terms of the number of different languages used.

---

<sup>1</sup> Object Orientated ("OO") splits data into "Objects" which can then be manipulated

<sup>2</sup> "Tag libraries define declarative, modular functionality that can be reused by any JSP page". [8]

<sup>3</sup> JavaBeans: A reusable Java component that can be accessed over a web server

*"MySQL is the world's most popular open source database, designed to be fast, reliable and easy to use." [9]*

Although MySQL is well known for being reliable, older versions do not focus heavily on security.

- o Oracle  
Oracle is the most expensive and popular commercial database available.

*"Oracle is widely regarded as one of the most popular full featured database systems on the market." [10]*

It is a complete database management system that deals with high volumes of data and combines high efficiency with high levels of security.

Oracle can be complicated to set up and manage and requires a skilled administrator to operate large scale systems. As the product designed in this project is a relatively small scale system, such issues are not cause for concern. Other factors to consider when using this system are the time it takes to run and resources required.

After much consideration, Oracle was used as the database of choice in this project due to its proficiency and high level of security. As a system developed specifically for industry, Oracle is both robust and reliable and its current use in the Computer Science department at Bristol University will enable hassle free integration.

Although chosen for use in this project, it will be possible to convert from Oracle to an alternative database format with ease.

### **2.3.2 Web Servers**

A web server is the essential aspect required to build a web application. Various web servers are available the best of which follow:

- o Microsoft Internet Information Services  
Microsoft Internet Information Service (IIS) is a Microsoft web server that can be used for programming with ASP but does not support non-Microsoft languages such as JSP and IIS better with a Microsoft Access database.
- o Apache Tomcat  
Tomcat is a Java based web server that provides extensive support for JSP and servlets. It has extensive security features and in my personal experience is a lot easier to customise than IIS.

The Tomcat web server was chosen for use in this system, as it provides the most support for Java based web projects.

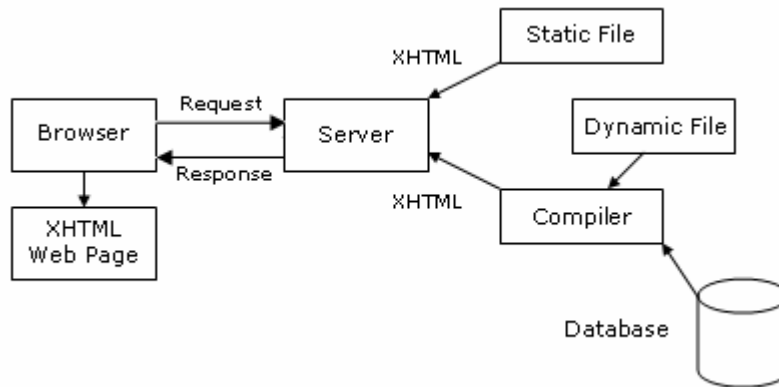
### 2.3.3 Internet Technologies

Several different “internet” technologies exist, but they all centre around a common aim - to allow the user to view, display or interact with data over large area of interconnected computers. The term “web” may also be used to refer to a collection of interconnect documents [11].

#### 2.3.3.1 Web Browsers

A web browser is a software application which aids users in displaying information from a web page. A browser displays information received from a server in HyperText Mark-up Language (HTML) format. HTML uses tags to structure and format a document. A common problem with certain browsers is the inability to interpret incomplete HTML structures which results in partial web pages being displayed. To combat this problem Extensible HyperText Mark-up Language (XHTML) was created. XHTML has similar syntax to HTML, but abides by stricter rules which create well formed documents. Figure 2 illustrates how XHTML communicates between browser and server.

Figure 2: Communication between web browser and server.



#### 2.3.2.1 Page Formatting

The best and most effective way of styling a web page is using a Cascading Style Sheet (CSS) which allows specific formatting to be added to a web page.

Not only does the style sheet offer efficient and effective formatting, it also allows separation of the actual page content from style frills and trimmings. In this manner single sheet styling makes alteration of style effects for a whole site effortless.

Browser compatibility issues are easy to repair with the use of style sheets. Simple syntax is used to reference elements in the HTML page as outlined below for a <p> element:

```
p
{
    color: red;
    size: 10px;
}
```

By adhering to CSS conventions, the online examination system will be designed to have easily altered style and be cross browser compatible.

### 2.3.3 Conclusions of Final Software

After a thorough review of all available software the following specification was chosen for use in this project

- Apache Tomcat Web Server
- Oracle Database
- Java 5
- JSP 2.0
- HTML and CSS

## 2.4 Exam Research - Question Formats

Although one of the aims of this project is to create a comfortable, user friendly system, one must remain attentive to maintaining a high standard of question formats.

The information that follows is a brief overview of the seven question formats used in the system designed. The bulk of this information was obtained from a teaching academy at the University of Wisconsin **[12]**, which is aimed at educating academics on the most effective questions in order to obtain maximum student potential.

### **Bloom's Taxonomy**

Benjamin Bloom created a system of categorising levels of student comprehension, aimed at finding the correct level of question complexity based on the students' level of knowledge.

*"Bloom's taxonomy is a system created to improve testing precision by categorising cognitive functioning into distinct levels." [13]*

Although his process was designed for the creation of correct question types based on level of complexity, his theories produced a "set" of question formats which are best used to produce maximum results.

According to Bloom's theory, the question formats that follow promote the most interaction at different levels of complexity. It should also be taken into account that test questions must fulfil four basic requirements. They must:

1. Be Achievable
2. Monitor Progress
3. Be diagnostic
4. Be Prognostic

**[14]**

### Multiple Choice

Multiple choice questions allow formulation of questions of differing complexities, from easy factual questions, to more in depth scenario questions. Multiple choice questions can be tailored to be as challenging as the lecturer wishes through the incorporation of believable "false" options.

### True/False

True or false questions are one of the more simple and common formats where a student has an even probability of getting a question correct. This question format is limited in its ability to assess knowledge as the student may have used a measured calculation to obtain an answer.

### Short Answer

Short answer question formats can be used in a variety of scenarios, from single word to multiple statement answers. Such questions require the student to enter information without prior clues or assistance as to the direction of the answer. One limitation of this



format is that it allows a limited level of complexity in that the question only requires a short answer.

Automatic marking for such a question is not possible due to variation in grammar and wording.

#### Long Answer/Essay

Long answer questions require students to submit a response without guidance and are good for testing high levels of intelligence. Students are able to communicate their opinions in more depth than a short answer question permits, however they cannot be automatically marked.

#### Matching

Matching questions are designed for exams that test vocabulary and understanding of definitions. Although not necessarily used often, and not for highly complex questions, their inclusion will complete the application.

#### Ordering

Ordering questions require the student to select the correct sequence for a series of answers and answers may be ordered in relation to importance or relevance. Questions of this type offer variation with respect to complexity.

#### Multiple Answer

A multiple answer question can be a highly useful tool for exams, requiring the student to select more than one answer to a question. It can be used for simple or complex scenarios and can test understanding by allowing choice.

## **3 Specification**

### **3.3 Requirements**

#### **3.3.2 Functional Requirements**

IEP will be a complete examination system that is easily adaptable to different educational environments. The system will be split into three sections that can be accessed by the student, lecturer and administrator respectively.

##### 3.3.2.1 Student Section

The student aspect of the system allows individuals to log in with unique passwords and be transferred to a secure server. Here, information such as IP addresses and personal credentials are validated, and the assessment procedure can begin. Any examinations to be sat in the next ten minutes are presented on screen and after further validation the student is able to proceed with the test. Upon completion of the examination, answers are saved.

At each stage of the assessment, validation checks are performed preventing users from making errors, losing work or committing plagiarism.

The system will include an integrated calculator eliminating the necessity for external calculator use.

##### 3.3.2.2 Lecturer Section

The lecturer aspect of the system can only be accessed by certified staff. Examinations can be created using standard online forms or via direct XML input. Lecturers are able to view, edit, monitor and delete their exams as well as issuing on the day passwords.

Upon completion of the exam, marking may begin. Automatic marking based on model answers inserted by the lecturer is completed before manual marking begins. All results can be viewed in a variety of different formats.

The inbuilt task system enables lecturers to create reminders and tasks for completion by a certain deadline.

##### 3.3.2.3 Administration Section

The administration side of the system allows complete customisation of the examination, from elements such as folder and image details, to database setup and display information.

Specific information such as IP ranges and individual passwords are set and stored in the administrator section of the system.

### **3.3.4 Efficiency**

The IEP system will be designed with efficiency as a priority. High volumes of users will be able to access the system without it crashing or hanging up, an area in which many existing systems fail with.

One major problem with efficiency involves database connections, and hundreds of connections at once can lead to long delays and errors. Every time a database query is entered a new connection to the database is created and over time can prove to be expensive and inefficient. The IEP system will be designed to keep database activity to a minimum by building Java structures which hold the data. In addition, to minimise the overhead caused by large numbers of queries, connection pooling will be implemented. A connection pool is a method used to minimise the load and enhance performance of a database. In web based applications, connections are constantly created then left hanging. A connection pool consists of a set number of connections, and each one is continually recycled.

A number of methods for creating connection pools exist. Hand coding in a programming language such as Java can be used, however is not guaranteed to work and may prove inefficient with excessive use.

By far the easiest and most efficient method of pooling makes uses of the Jakarta-Commons database connection pool [15]. This is built in Java a reusable component package that is configured within Tomcat.

### **3.3.5 Design Constraints**

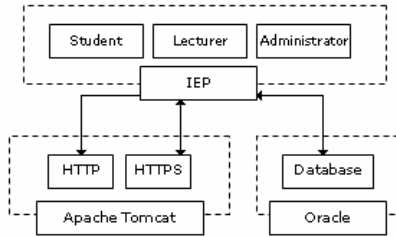
As with most web applications, browser independence is of great importance. The IEP system will be built using browser independent HTML and JavaScript, adhering to the strict standards set out World Wide Web Consortium (W3C) [16].

W3C is a governing body for standards on the web that creates open standards allowing the internet to grow in a single direction. The standards are strict, and W3C provides validation software to ensure that IEP will conform to these standards.

### 3.4 System Environment

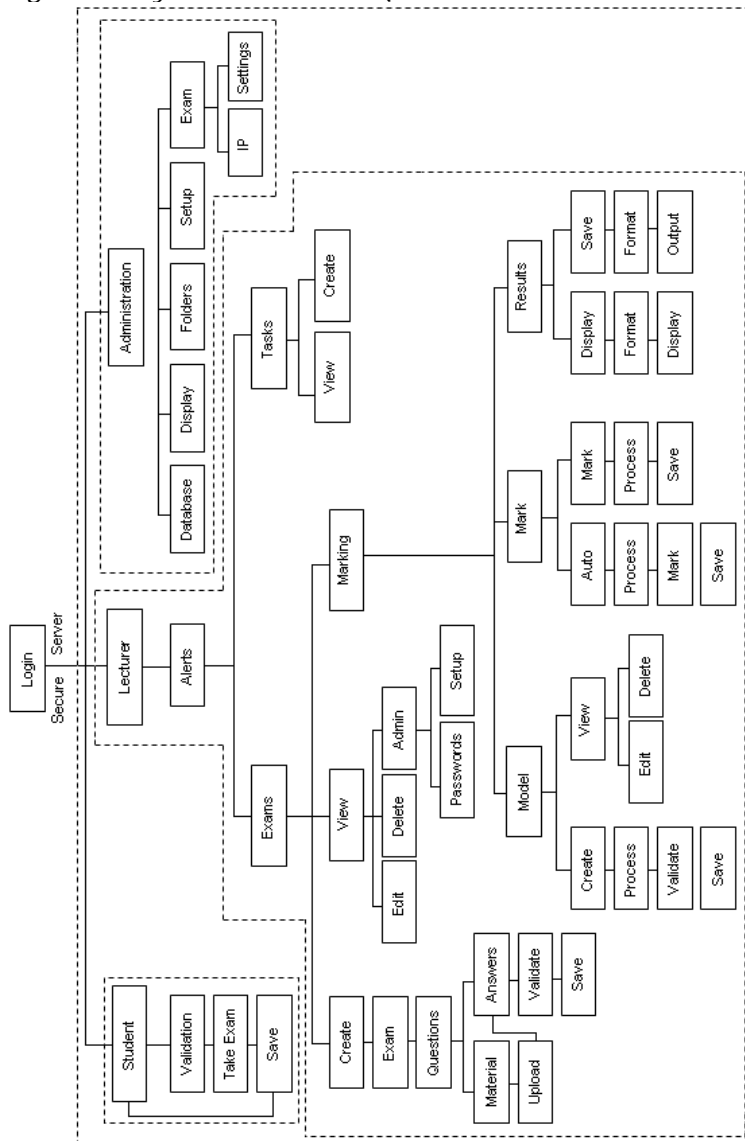
The system itself interacts with other software as shown in *Figure 3*.

*Figure 3: System Interaction.*



The system works in three levels as shown in *Figure 4*, which outlines the layers of security and how the system interacts with itself. Each layer (separated by dotted lines) can only be accessed according to the correct user roles and privileges.

*Figure 4: System Structure (dotted lines indicate the four layers of security)*



### **3.5 Interface**

A simple to use yet effective interface is crucial in the development of a successful system. Many people find that a complicated looking interface would be enough to confuse them before even interacting with the system. The IEP system will have a straightforward interface suitable for any category of user. The lecturer interface on the other hand contains several complex elements and a trade off between efficiency and simplicity must be made.

### **3.6 Security**

High levels of security will be required for this system to prevent students and others accessing the system without authority.

As a website application, several aspects of security require consideration. HTTPS acts as the first security layer and requires the user to validate themselves before access is granted. A second validation is implemented when a resource is accessed, ensuring the user possesses sufficient privileges for the action.

#### SQL Injection

An SQL injection attack is the most common attack on a web based database. Whenever a query is made, all characters that could lead to a potential security breach will be removed beforehand.

#### Bombardment

When an attacker bombards a database with more queries than it can handle, the database may overload. The solution to this problem is to use connection pooling, as previously described. Limiting the number of connections allowed reduces bombardment considerably.

#### Security Procedures

Although several methods of security breach have been considered in the design of this system it is not possible to identify every existing risk and hence three standard "best practice" procedures will be incorporated to keep vulnerability low.

##### Shared Computers

Caching of pages on shared computers could allow unauthorised viewing of stored files. Methods that do not save details on individual computers have been employed to combat this problem.

##### User Validation

Implementation of strict validation routines that not only check the format of the inputted data but also verify that the content is not malicious will prevent misuse of the system. Queries that are submitted will be checked to see if characters that can be used in a scripting attack are included. If this is the case, such characters will be converted to a safe format.

##### Session Management

Session management is often overlooked in web applications and not setting session timeouts can mean that redundant sessions are available for attack. A session can be hijacked by stealing a users session token over a Local Area

Network (LAN). As the system is designed for use in Universities, this risk must be evaluated.

Continual assessment of security risks must be carried out in order to keep abreast of any new developments.

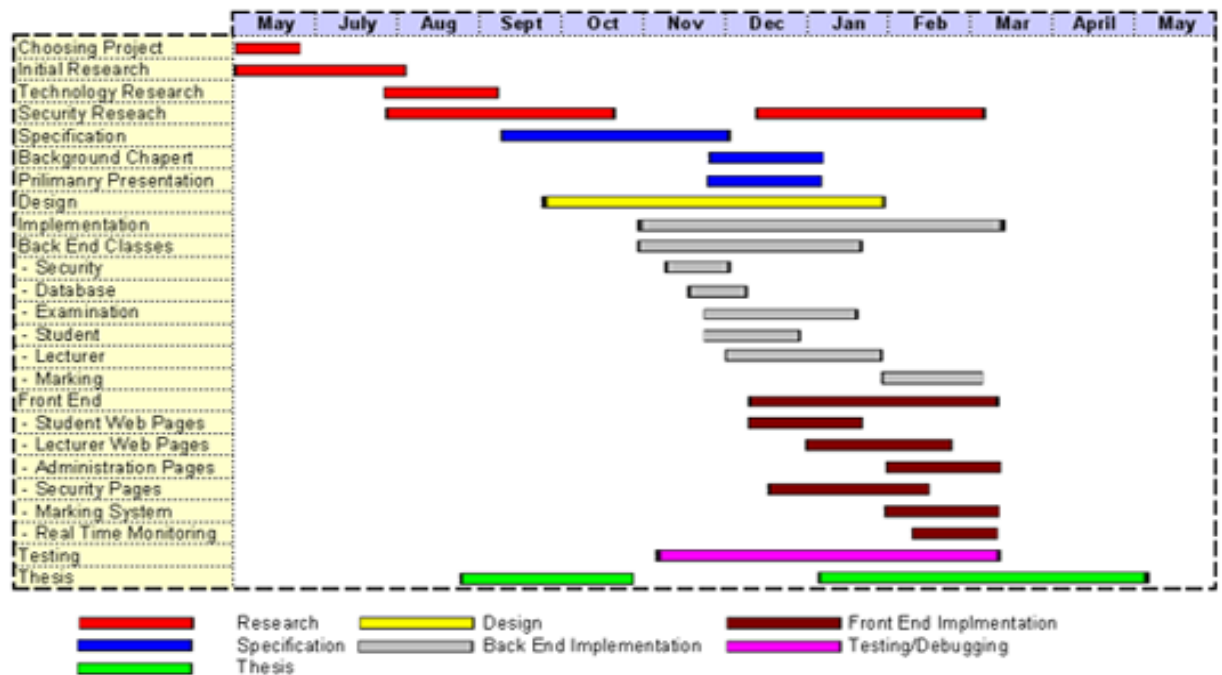
### 3.7 Data Transfer

Improvements in the efficiency and security of a system containing large amounts of secure data can be achieved through the use of Java structures. Java structures and JavaBeans will be used in the system to hold, transfer and access data.

Although data transfer in the structures is preferred, it is still necessary to have Java inside the JSP page for functions such as getting parameters and if statements. Although most data transfer will occur in the structures, small queries are still required in the web page for validation purposes.

### 3.8 Project Schedule

Figure 5: Project Schedule



### 3.9 Class Design

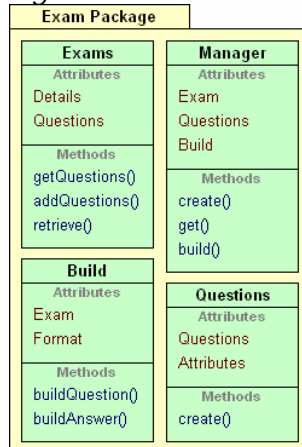
#### 3.9.4 Exam Package

The exam package will contain all material relevant to exams, from creating the exam to handling exam information and validation. The exam package is split into two sections; student and lecturer.

The student sections will handle the exam data, retrieving questions from the database, handling data input from the exam and building question structure. When an

input is entered, the package will build a class to hold data until the exam is complete, at which point it will be exported to the database.

Figure 6: Basic UML diagram for the exam package



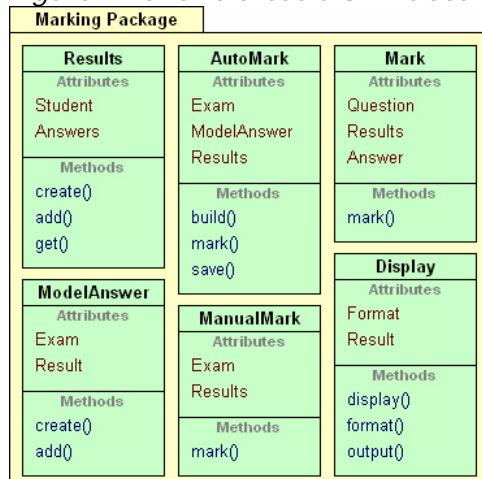
The lecturer section is more complex and will contain the classes that allow the lecturer to build the exam, retrieve results and alter data.

The structure of the package is important as the data needs to be secure, reliably handled and transferred. To prevent illegal access of data all variables will be private, and data will only be accessed using `get` and `set`<sup>4</sup> methods on specific data which is allowed to be viewed.

### 3.9.5 Marking Package

The marking package will contain all functions for marking and display results. The package will handle the process of automatic marking, which requires a model answer to be submitted by the lecturer. This model answer is then used to mark specific types of questions. Marking classes will also handle the data during the manual marking process. The answers are pulled from the database and displayed in the requested format for the lecturer to mark. Results can be displayed graphically in terms of percentages, through text, in the form of averages or may be outputted to an XML file which can then be inserted into a database or other application.

Figure 7: Shows a basic UML class diagram for the marking package



<sup>4</sup> Get and set methods allow classes to alter or retrieve data

### **3.9.6 Tasks Package**

This package will handle the creation, viewing and displaying of tasks as well as date calculations.

### **3.9.7 Security Package**

The security package is split into three levels allowing different levels of access according to user type. The security package will hold:

- functions which validate user information
- IP settings
- Course information
- Connection to the secure server
- Session and user validation
- Role checking
- Other security features that may be required.

### **3.9.8 Monitoring Package**

The monitoring package will allow lecturers to monitor exams currently underway. Listeners will be used to monitor activity on the server. When activity is detected, the package will locate the lecturer associated with the exam, and allow access to the session information (such as student information and exam status). A listener class is built on the JavaBean topology, and interacts with `ServletContext` and `HttpSession` objects. Use of a listener allows monitoring of lifecycle events.

### **3.9.9 Utilities Package**

Other aspects of the exam, such as database and results handling, user tracking and application settings will be encompassed within the utilities package.



## 4 Implementation

### 4.1 Database

The database is the backbone of the system, and it is essential that it is efficient and cost effective. Poorly designed databases complicate the implementation process.

*"A well-designed database is easier to implement than a poorly designed database."*  
**[17]**

#### 4.1.1 Structure

Currently, the University of Bristol have an existing database implemented for student management which has been incorporated into the IEP system. Below is an overview of existing table structures:

##### People

The People table contains personal data regarding University personnel with elements such as name and email.

##### Sessions

University specific information such as degree course, examination number and tutor are stored within this table.

##### Teachers

Lecturer details contain information about course units such as unit title and lecturer identification number.

##### Courses

The courses table states details for each degree type; it holds the degree code, faculty, department, number of years and degree classification.

##### Unit Sessions

This session contains a list of all students undertaking a particular unit.

Additional tables required for the successful implementation of the system are briefly outlined below:

##### Exams

All data relating to exams including course and unit information, start date and time are stored within the table.

##### Questions

Questions are held as individual entities within this table. The table contains information on exam origin, question format and other question details.

##### Model Answers

Information pertaining to model answers is held within this table.

##### Results

Student answers to exams are collected here, and reference is made to the student, exam, and question.

##### Marks

The individual mark obtain for each question is stored at this juncture.

##### Question Formats

Available question formats are stored in this table, along with a unique identifier, name and html constructor example.

##### Settings

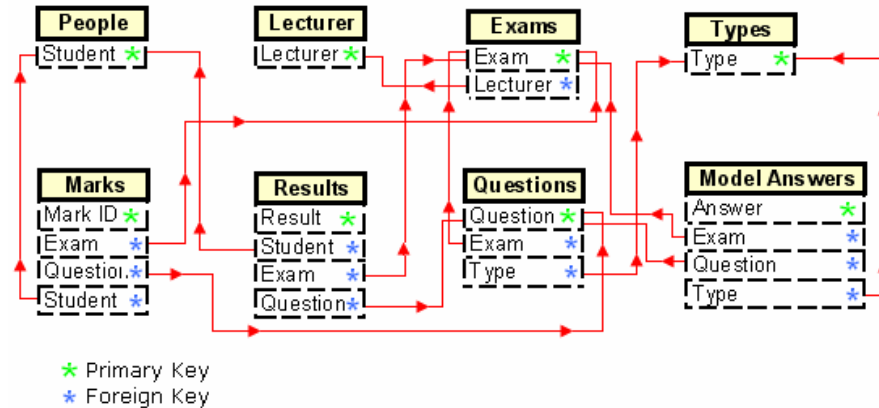
To allow complete customisation of the system, an additional settings table will be required to store all application attributes used within the system.

### 4.1.2 Table Relationships

A primary key is a unique identifier for each record within a table. Some tables contain foreign keys which provide references to primary keys within another table.

If constructed incorrectly, foreign keys can destroy referential integrity, and in order for this to be upheld, references to primary keys from foreign keys must exist. Figure 8 graphically emphasises the relationships between table elements within this system.

Figure 8: Database Relationships



### 4.1.3 Connection Pooling

During the research phase, the most effective method of implementing connection pooling was found to be through the use of Tomcat.

For successful connection pooling Java Naming and Directory Interface (JNDI) requires configuration within Tomcat allowing Java components to locate each other at runtime on the system.

In addition, to improve the throughput and efficiency provided by connection pooling, Tomcat's connector entity should contain an `acceptCount` attribute. This element is established to inform the server of the connection limits required during heavy traffic periods. A reference to the IEP database is also required to inform the service with which database to implement connection pooling, as seen below.

```
<resource-ref>
  <description>Connection Pooling</description>
  <res-ref-name>jdbc/IEP</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
</resource-ref>
```

### 4.1.4 Database Class

It is essential to structure and manage the database functions accurately. By designing a class to manage the database it provides more security and separates the database activity from the JSP code. Database essential methods include `connect()`, `disconnect()`, `create()` and `query()`.

The database is managed using the Java `DriverManager` initialised using the following code which connects to the database using an available pooled connection:

```
Class.forName(dbDriver);
Connection = DriverManager.getConnection(address, name, password);
```

The `query()` function performs an SQL query on the associated database. Handling queries inside the database manager allows stricter monitoring and regulation to prevent security threats.

#### 4.1.5 Database Results

Once a query has been made, the database returns a `ResultSet` containing requested data. Java's `ResultSet` has existing pitfalls (3.8) the solution requires a custom result class.

```
while (rs.next()) // while there are results left in the set
{
    ArrayList<String> row = new ArrayList<String>(); // row list
    for (int i=0; i<columns.size(); i++) // loop through columns
    {
        String data = rs.getString(i+1); // get data
        row.add(data); // add data to row
    }
    table.add(row); // add the row to the table
}
```

When a query is executed, the `ResultSet` is passed as a parameter. The `ResultSet` is then used to build a custom result table. Once initialised, the table can be accessed using `getCell(row, col)`. Convenience functions not provided with the `ResultSet` are available, including getting row and column counts.

## 4.2 Interface

The system requires two interfaces, student and lecturer. Although separate, the styles for both interfaces are consistent.

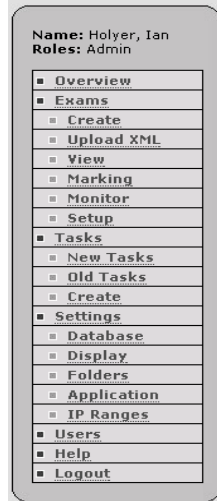
### 4.2.1 Navigation

The system has a complicated hierarchy, as shown in figure 4. It is crucial that the navigation design is fluid.

#### 1. Lecturer Navigation

As a more complex navigational design, the lecturer section requires a hierarchal structure. Certain elements are displayed based on user privileges. The navigation must be fluent and well formed as shown in figure 9, which indicated the hierarchal design and multiple levels.

Figure 9: Lecturer Navigation



## 2. Student Navigation

Simplicity is a key element in implementing the student navigation. To minimise human error, access to links are restricted to the task currently in operation, as shown in figure 10.

Figure 10: Student Navigation



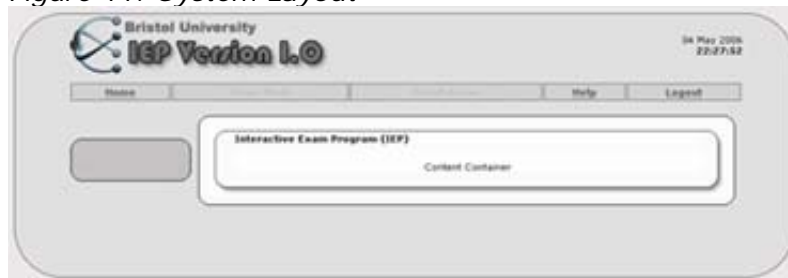
### 4.2.2 Style

With the ability of formatting pages using CSS (2.3.2.1), the style can be consistent throughout the system. The system is designed as an education package, whereby the use of bright colours and flashy imagery are impractical and unnecessary. The system will follow a simple fluent style with variants of grey, black and white.

### 4.2.3 Layout

Due to the nature of the system, the layout should complement the style. The layout as shown in figure 11 follows a consistent style, with fluid lines and round corners.

Figure 11: System Layout



## 4.3 Calculator

An integrated calculator has been incorporated within the system, written as an applet allowing interaction within a web page. To include the applet into a web page an Object tag is required which specifies the location and source.

```
<object codebase="iepcalculator" code="iepcalculator.class"></object>
```

The implementation of the calculator monitors for user input, when a button is pressed, the action is processed. The result is calculated and outputted to the screen.

The calculator interface follows the site style and contains shortcut keys for usability as shown in figure 12.

Figure 12: Calculator Applet



## 4.4 Security

Implementations of the outer layers of security are necessary before the system can function correctly.

### 4.4.1 Security Realms

A security realm protects web application resources with defined security constraints, allowing only specified users access to the resource. Tomcat has an integrated Realm system that can be used to manage security. The Tomcat roles permitted for use with this system are shown below:

```
<role rolename="lecturer"/>
<role rolename="admin"/>
<role rolename="student"/>
```

Resources can then be protected allowing only specified roles to access them. Protecting a resource should be configured manually, shown below:

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>IEP</web-resource-name>
    <url-pattern>/lecturer/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>lecturer</role-name>
  </auth-constraint>
</security-constraint>
```

### 4.4.2 HTTPS

HTTPS (2.2.5) provides the initial security layer. The protocol requires detailed configuration using Tomcat. The following are steps required to configure HTTPS:

#### Server Certificate

Server certificates are used to authenticate clients; a certificate is required to operate HTTPS and can be generated using Tomcat's `keytool` facility. Once created, it must then be imported to be associated with Tomcat.

## Configuration

An SSL connector must be configured in Tomcat, detailing the port number that the server will listen to for incoming connections. Once configured, all connections to the system will be directed to the secure port.

### 4.4.3 Security Validation

On connection to the system, resources accessed require security processing. Once accessed, Tomcat security checks that the user privileges provide have the correct authorisation. Checks to verify user logon and connection over a secure server are performed as shown below.

```
if ((request.getRemoteUser() == null) || (request.isSecure() == false))  
    Throw new Error("Validation Error")
```

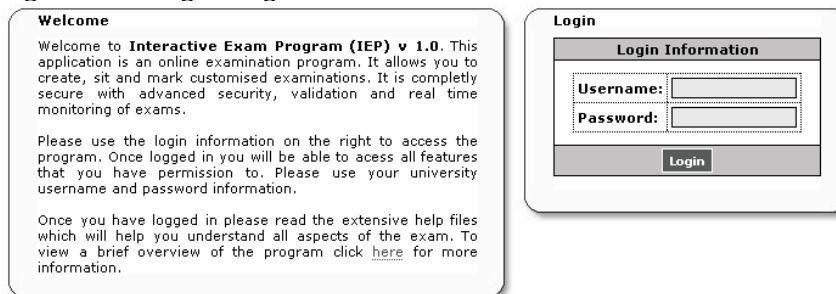
## 4.5 System Access

### 4.5.1 Index

(*index.jsp*)

If resources are accessed before user credentials have been validated, the system automatically redirects to this page. Figure 13 shows the interface presented on connection.

Figure 13: Login Page

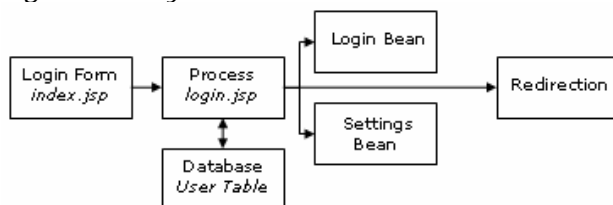


### 4.5.2 Login

(*login.jsp*)

Once credentials have been entered, redirection to the secure server and login procedures are initialised. Credential parameters are accepted with security risk characters removed. The database is queried to validate the user parameters. Once verified, user (4.5.3) and settings (4.5.3) JavaBeans are created. Finally, upon successful login, the system retrieves the associated privileges and redirects the user to the correct section of the system. Figure 14 shows the communication during the access process.

Figure 14: System Access Interface



### 4.5.3 User JavaBean

(IEP/Users.java)

Once credentials are validated, database information constructs a user bean, which is used throughout the site to retrieve user specific information, and provides a secure way of tracking users.

### 4.5.4 Settings Bean

To ensure a dynamic and completely adaptable system, a settings bean is created to hold folder, image and CSS locations, page names and database settings. When a resource is accessed, the setting is retrieved using get methods shown below.

```
public String getCss(){ return cssFile; }
public String getTitle(){ return pageTitle; }
public String getImgPath(){ return imgPath; }
```

## 4.6 Student Section

### 4.6.1 Welcome

(student/main.jsp)

Once securely logged in, examinations that are preparing to commence are displayed. This section performs the following:

- Time and date checks for exam start times.
- Student credential checks to ensure they are permitted to take an exam.
- Disabling the exam if already taken.

#### Time Checks

(IEP/Util/ExamTimer.java)

This java class, accepts an exam time and date as parameters, and then retrieves the current date and time using the java.util.Date functions. The times, such as time left, time until the end of exam are then calculated and checks to see if the exam is allowed to be accessed, as shown below.

```
public boolean isAllowed()
{
// check time is within limit
if(start.getTime() <= now.getTime() && end.getTime() >= now.getTime())
{ return true; }// allowed
return false; // not allowed
}
```

Once validated, the available exams are displayed, as shown in figure 15. The page is refreshed every 15 seconds to update the exam start times.

Figure 15: Main Page

**Welcome to IEP**

Welcome to the *Interactive Exam Program (IEP)*. You are logged in as a student. Please logout correctly using the **Logout** link. If you have any exams which are to be taken today they will appear below, when they do please click the **[ Start Exam ]** option which will take you to the exam.

Please read the **Help** files before you take the exam. The help files will be available throughout the exam from the menu bar at the top of the page. If you have any other problems whilst taking the exam please call a member of staff.

When you start your exam you will be asked to enter your on the day password, if you do not have a password please contact a member of staff immediatly.

Title	Start Time	
COMS12100 - Integration	16.00	[ Start Exam ]

**Note:** This page will be refreshed every 15 seconds, when the exam is ready to start, the **[ Start Exam ]** option will appear.

## 4.6.2 Start Exam

(*student/startExam.jsp*)

When an exam is selected, student users are redirected to the exam processing page. Further validation checks such as those to ensure that the student taking the exam is on the course are performed before the student may proceed (see below).

```
public String validateExam()
{
    ExamResults isOnCourse = db.query(...); // check user in on course
    ExamResults isOnUnit = db.query(...); // check user is on unit
    ExamTimer et = new ExamTimer(startHour, startMin, len);

    if(isOncourse==false || isOnUnit==false || et.isAllowed()==false)
        throw new Error("Invalid Exam");
}
```

Finally, the student is then requested to enter their on-the-day password. Invalid passwords or credentials will redirect the student to the main page displaying the error that occurred.

## 4.6.3 Exams

(*student/exams.jsp*)

### 4.6.3.1 Exam Back End

#### Question Class

(*IEP/Questions.java*)

The question class contains question information such as exam ID, title and type. It provides get and set methods for accessing and altering the contained data.

#### ExamDetails Class

(*IEP/Student/ExamDetails.java*)

Once selected, the resulting exam is built into this class. It contains a list of question structures (*Question.java*), plus all other associated information, such as title, date and start time. The class is a container for holding all exam details.

#### Answers Class

(*IEP/Student/Answers.java*)

Once a student has answered a question, the result is built into the Answers class which saves the question ID, answer and title.

#### Exam Class

(*IEP/Student/Exam.java*)

The exam class is built as a JavaBean container. It is referenced within the Exams.jsp page using a JSP tag library:

```
<jsp:useBean class="IEP.Student.Exam" id="exam" scope="session" />
```

The class stores the exam details (*ExamDetails.java*), and a list of student answers along with other exam related information. The class is a manager for the exam, used to combine questions, answers and data together. Attributes can be created and modified through this container. When displaying information in the exam, details are referenced from this class using get and set methods (see below). The exam class contains all validation, checking and saving functions.



```
public String getTitle(){ return exam.getTitle(); }
public String getCourse(){ return exam.getCourse(); }
public String getUnit(){ return exam.getUnit(); }
public String getNoQuestions(){ return exam.getNoQuestions(); }
public String getId(){ return exam.getId(); }
...
public Questions getQuestion(int i){ return exam.getQuestion(i); }
```

### Build Class

(IEP/Build.java)

To ensure a dynamic system, questions are required to be built at run time. The build class accepts a question and an optional answer as parameters.

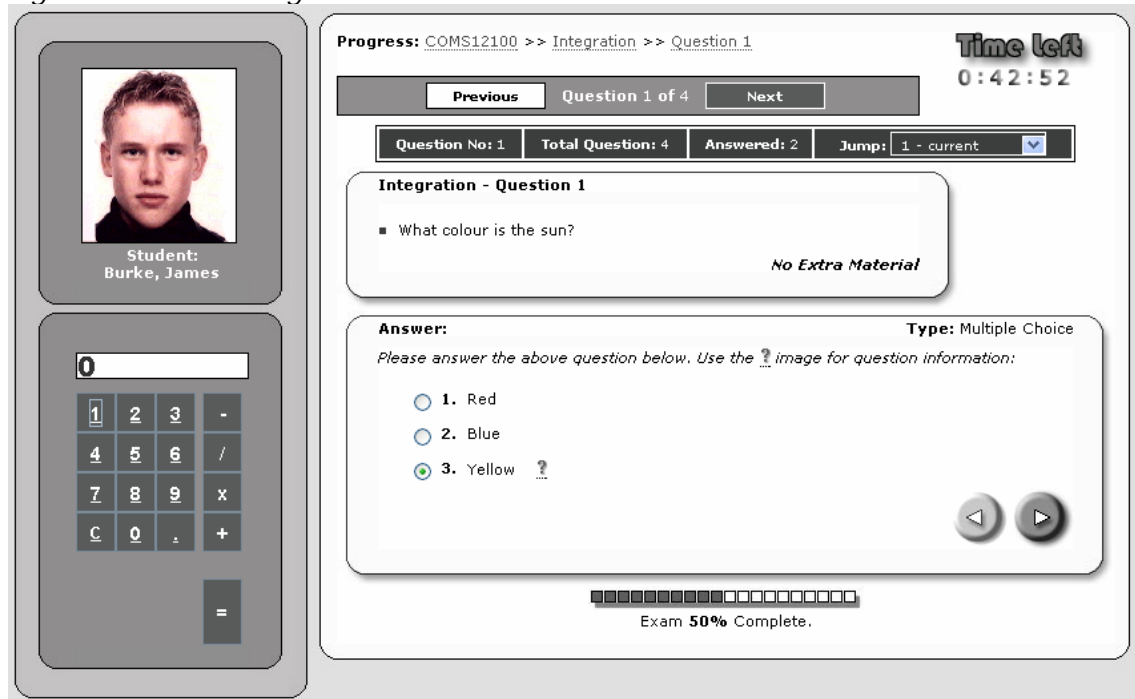
When initialised, the class determines the question format and automatically builds the output (see below).

```
private String buildMultipleChoice()
{
    String answers = question.getAnswers();
    // loop through each attribute
    for(int i=0; i < answers.length; i++)
    {
        // build the input tag based on parameters
        build += buildInput("radio", "answer", answers[i]);
        build += "<strong>" + (i + 1) + ".</strong>&nbsp;";
        build += answers[i]; // display attribute value
    }
    return build;
}
```

#### **4.6.1.2 Layout**

Figure 16 shows the layout of the exam interface. A photograph of the student is displayed to allow moderators to ensure that the person sitting at the computer is the one specified. The calculator (4.3) is located below the photograph, for easy access and usability.

Figure 16: Exams Page



### Progress

A graphical progress bar detailing the exam completion rate is displayed below the answer. As questions are answered, the exam class calculates the progress;

```
public double getPercent()  
public String getPercentImg()
```

The result is calculated and a graphical image displayed.

### Time Left

A count down for the time left in the exam has been included, using JavaScript `setTimeout("startTime()", 1000)` which calls the function every one second to update the time.

### Navigation

To prevent human error, the interaction for the user is limited. During an exam, the user may only move between questions and finish the exam. There are three elements that allow the student to progress between the questions.

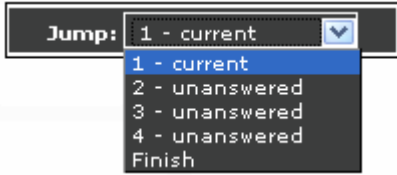
Figure 17 shows the first navigation method, allowing the user to navigate between previous and next questions.

Figure 17: Top Navigation



Questions can be selected from a drop down menu, informing the user of the question status, as shown in figure 18.

Figure 18: Drop-Down Navigation



The final method uses straightforward arrow buttons allowing navigation between questions (figure 19).

Figure 19: Quick Navigation



### Information Bar

To keep the student informed, a graphical representation of the current exam status showing the question number, total questions and number of answered questions is given and is shown in figure 20.

Figure 20: Information Bar



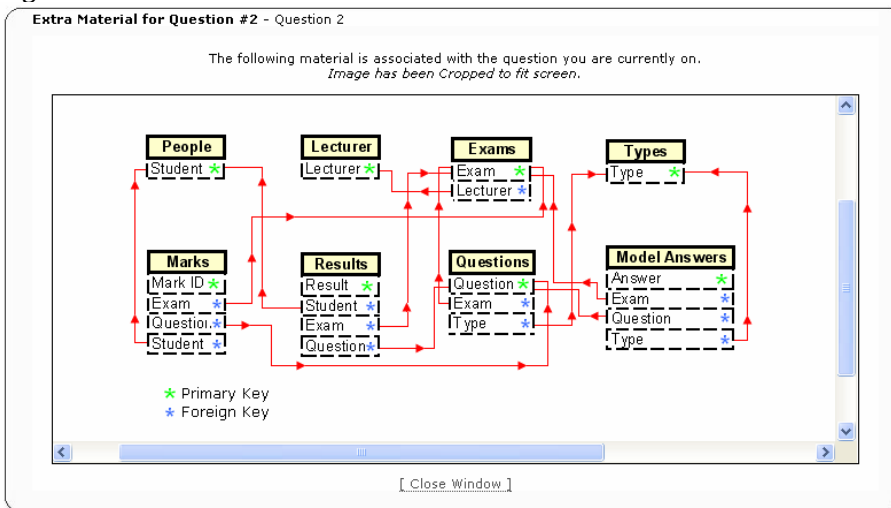
### 4.6.1.3 Setup

When initialised, the system creates a new instance of the Exam Bean (*Exam.java*). A parameter `questionNumber` is passed between pages to inform the system the current question the student attempting. The question is retrieved from the Exam bean and, the results passed as parameters to the build class (*Build.java*), which generate and displays the question and answer inputs.

### 4.6.1.4 Material

Questions which have additional material associated with them may be selected by the user via "View Material" link, which then displays the material resource (figure 21).

Figure 21: View Material



#### 4.6.1.5 Answering

On completion of a question, the system builds a new `Answer` class (`Answer.java`) inserting the students answer. Once produced the `Answer` is added to the Exam bean.

```
Answer answer = new Answer(exam, question, answer, id);
exam.addAnswer(answer);
```

If more than one answer to a question is detected, the answers are separated by a semi-colon. To retrieve each answer, a `String.split()` function separates the answer at the semi-colon.

```
String[] answers = answer.split(";");
```

#### 4.6.1.6 Confirmation

Upon completion of the exam, the user is presented with a confirmation page where they may view or alter all questions answered and unanswered (figure 22)

Figure 22: Exam Confirmation

**Exam Completed**

**Congratulations!** You have finished the exam. Below are the questions you have and have not answered. If you are within the time limit you can go back and change/answer the questions. Simply select the question title below.

**Completed Questions**

No	Title	Answer(s)
#1	Question 1	Yellow
#2	Question 2	63
#3	Question 3	1. 1 2. 3 3. 4 4. 2
#4	Question 4	Computer Science is great fun!

**Incomplete Questions**

No	Title	Type
No Incomplete Questions		

[Complete, Save and Exit](#)

*Note: By clicking save and complete it will save you questions and you will not be able to return to the exam.*

#### 4.6.1.7 Completion

At any point during the exam, if time runs out or completion has been selected, redirection to the completion page occurs. All answers are committed to the database and the class is removed, preventing the student from returning to the exam. Once saved, exam statistics are displayed offering the option to return to the main page as show in figure 23.

Figure 23: Completion Page

**Exam Completed**

**Thank you** Burke, James you have successfully completed the exam. You can no longer return to the exam. Please click [here](#) to return to the exam main page.

**Exam Information**

---

Total Questions: 4  
Questions Answered: 4  
Questions Unanswered: 0  
Start Time: 14:00  
End Time: 14:46:56

---

[\[ Home \]](#)

## 4.7 Lecturer Section

### 4.7.1 Exams

#### 4.7.1.3 Exam Back End

##### Create Exam

(IEP/Lecturer/createExam.java)

This Java bean has been designed to handle all aspects and stages required to create an examination. Once the bean has been initialised, the different stages required to create an exam can be managed. On creation, the bean requires exam details such as course, title, date and start time to be inputted. Through each stage of the process, questions can be added, deleted or altered. Get and set methods for each element are also provided (*shown below*). Once all data has been validated within the class, the exam and all related material are committed to the database.

```
public void addQuestion(Questions q){ question.add(q); }
public Questions getQuestion(int no){ return question.get(no); }
public void save()
{
    if (isValid() == false) throw new Error("Invalid Exam");
    database.query("insert into exams values" ...);
    for(int i=0; i < questions.size(); i++)
        database.query("insert into questions values" ...);
}
```

##### Upload

(IEP/Utils/Upload/)

For questions associated with additional material, the option to upload a file is given to the lecturer. The Upload package handles all file uploading. When submitted the "multi-part/form data" which contains the upload file is passed to `FileUpload`, and is then parsed by a `HttpMultiPartParser` [18]. The data is split and a hash table created containing the file data. The file details are removed from the hash table and, the file is then constructed in a directory passed as a parameter.

```
public string process()
{
    HttpMultiPartParser parser = new HttpMultiPartParser();
    Hashtable hashTable = parser.processData(request.getInputStream());
    FileInfo fileInfo = (FileInfo) hashTable.get("myFile");
    File fileName = fileInfo.file; // construct file
}
```

##### XML Parsing

(IEP/XML/), (IEP/Lecturer/createXmlExam.java)

An XML file can be used to create an exam. The file is first uploaded using `UploadFile.java`, and then passed to `XMLParse.java`. `XMLParse.java` uses a `DocumentBuilderFactory` to validate and construct the Document Object Model (DOM) tree and passes it to the `createXmlExam.java`, which extracts the exam information. This information is then validated, constructed into a `createExam` class and saved to the database. An example code snippet is shown below.

```

(XMLParse.java)
public Document parse()
{
    DocumentBuilder builder = factory.newDocumentBuilder();
    if (validate) builder.setErrorHandler(new XMLReporter());
    Document document = builder.parse(new File(file)); ...
}
(createXmlExam.java)
public void processExam(Document doc)
{
    Node examNode = xml.getNode("entity");
    for (int j=0; j<examNode.getLength(); j++)
    {
        extractDate(examNode.item(i));
    } ...
}

```

### Calendar

(*lecturer/calendar.jsp*)

To aid in date selection, a calendar has been provided. It retrieves the current date, displaying all days within the current month. Access to days prior to the current date are disabled. Figure 25 shows the interface for the calendar. Arrows are provided to scroll through months and years. When selected, the month and year are passed as parameters, where the dates are refreshed.

### Generate Passwords

(*lecturer/PasswordGen.java*)

The password class generates on the day passwords, creating a string of 10 characters for each student taking the exam. Once all passwords have been generated, the class writes the passwords to the database with the associated exam.

```

StringBuffer temp = new StringBuffer(length);
for ( int i = 0; i < length; i++ )
{
    char c = (char) ( random.nextInt( 'Z' - 'A' + 1 ) + 'A' );
    temp.append( c );
}

```

### Monitor Exams

(*listeners/SessionCounter.java*)

(*Monitor/ExamMonitor.java*)

As students are participating in an exam, the lecturer has the option to monitor progress. During the student exam process, when an action is processed, the ExamMonitor class is built with a snapshot of the students current statistics. These are then added as a session variable using `session.setAttribute("exam", ExamMonitor)`. The first step in implementing the monitor section was to create a listener class. A session listener sits on the web server and listens for sessions. If a session change is detected the listener is notified, allowing the session to be processed.

The session listener needs to be configured in Tomcat using the following code:

```

<listener>
    <listener-class>IEP.listeners.SessionCounter</listener-class>
</listener>

```

The `SessionCounter` class, waits until a session alteration has been detected, and then fires the `sessionCreated(...)` or the `attributeAdded(...)` function. When these methods are called, the session is retrieved; if the attribute added is of an "exam" type then the `ExamMonitor` class is taken from the session. The `ExamMonitor` class is then added or altered into a `Vector` holding all exam sessions, as shown below.

```
public int addValues(Object val, String aname)
{
    for(int i=0; i < values.size(); i++) // loop through sessions
    {
        ExamMonitor thisObject = (ExamMonitor)values.get(i); // get existing
        ExamMonitor value = (ExamMonitor)val; // get new session object
        if(thisObject.getStudent().equals(value.getStudent())) remove(i);
    }
    values.add(i, val); // add to vector
}
```

## Marking

(*IEP/Marking*)

The four sections for marking are as follows; model answers (`ModelAnswer.java`), automatic marking (`AutoMarking.java`), manual marking (`ManualMarking.java`) and the mark results (`Results.java`).

### Model Answers

The `ModelAnswer` structure holds the model answer to a single question. It contains the exam name, question, format and the correct answer. `Model.java` is a `JavaBean` container that holds all `ModelAnswers` for an exam. The `Model` class, takes an exam ID, and retrieves all the related questions, checking each question to confirm it can be automatically marked. If the question *can* be automatically marked a new `ModelAnswer` structure is created holding the question, and waits for the lecturer to insert the answer.

### Automatic Marking

The `AutoMarking` class accepts an exam ID and, retrieves the model answers for the exam and the students answers (`IEP/Student/Answers.java`). The class then loops through each of the model answers comparing them against student answers. Each answer is then assigned a mark, as shown below. The assigned mark is then used to create a `Results` structure, holding the exam and student ID, question and mark. Once all student answers have been marked, the `Results` are then saved to the database.

```
public int mark (String mAnswer String sAnswer, Scheme markScheme)
{
    int mark = 0;
    String[] mAnswers = mAnswer.split(";"); // split model answers up
    String[] sAnswers = sAnswer.split(";"); // split student answers

    for(int i=0; I < mAnswers.length; i++) // loop through answers
    {
        ...
        mark += compare(mAnswer[i], sAnswer, markScheme);
        ...
    }
    return mark;
}
```

## Manual Marking

The manual marking class loads all questions, student answers and results into its structure. The lecturer can then sift through each question assigning it a mark. Automatically marked questions will appear with each students mark, and will allow the lecturer to change the mark if required. Once the marking process has been completed, all Results are saved to the database.

### 4.7.1.4 Create Exam

There are two methods that can be employed to create an exam: using the system forms or using an XML file.

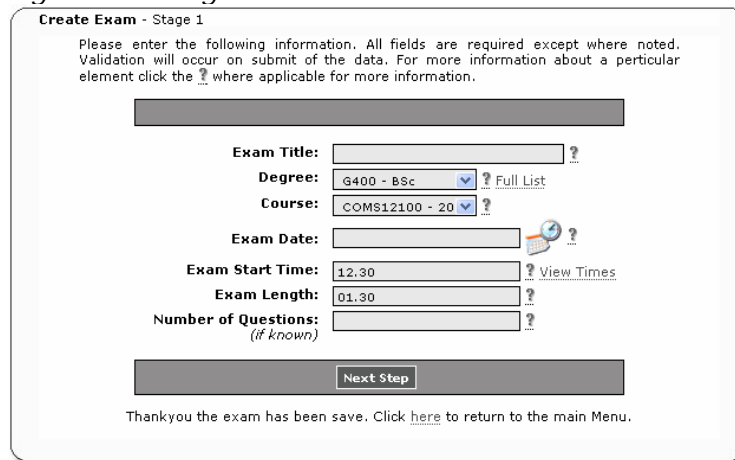
#### Form Method

(*lecturer/exams.jsp*)

#### Stage One

The first stage requires essential examination data to be entered including elements such as title, unit, date and start time. To prevent errors, the unit and course attributes have been provided in a drop-down box with a list of values that the lecturer is associated with. Figure 24 shows the stage one interface.

Figure 24: Stage One



**Create Exam - Stage 1**

Please enter the following information. All fields are required except where noted. Validation will occur on submit of the data. For more information about a particular element click the ? where applicable for more information.

**Exam Title:**

**Degree:** G400 - BSc

**Course:** COMS12100 - 20

**Exam Date:**

**Exam Start Time:** 12.30

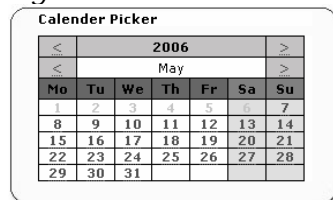
**Exam Length:** 01.30

**Number of Questions:**  (if known)

Thankyou the exam has been save. Click [here](#) to return to the main Menu.

The date requires a strict format (dd/Month/yyyy) and for this reason, a calendar page that allows the user to scroll through months and days (*figure 25*) has been created. Once a value has been selected it is automatically inserted into the "Exam Date" attribute.

Figure 25: Calendar Facility



**Calendar Picker**

2006						
May						
Mo	Tu	We	Th	Fr	Sa	Su
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

When the "Next Step" button is selected, the system validates that the required element values have been inserted, and creates a new `createExam()` JavaBean containing the exam data.



## Stage Two

Stage two requires the questions related to the exam to be entered. Multiple choice, answer and ordering questions require extra input. When the format is selected; the extra attributes required are built at runtime for user input. Figure 26 shows the stage two interface, with the multiple choice format selected.

Figure 26: Stage Two

**Create Exam - Stage 2**  
Please enter the following information. All fields are required except where noted. Validation will occur on submit of the data. For more information about a particular element click the ? where applicable for more information.

**Question 1**  
Please enter the following information for the questions.

**Question Title:**  ?

**Question Type:**  ? [View List](#)

**Question:**  ?  
Characters Left: 159

**Reference Material URL:**  ?  
[Upload Material](#)

Please enter the multiple choice/answer options below. There is a limit of 5 options.

**Answer Option 1:**  ?

**Answer Option 2:**  ?

**Answer Option 3:**  ?

**Answer Option 4:**  ?

**Answer Option 5:**  ?

Thank you the exam has been save. Click [here](#) to return to the main Menu.

In order to minimise database size, question length has been restricted to 200 characters. Below the question box (*figure 26*), "Characters Left" informs the lecturer of the number of characters remaining for the question. Using simple JavaScript on key press, the length of the question is retrieved, and a calculation of the number of remaining characters is performed.

### Material

To allow extra material to be used, the lecturer can either insert a URL to the resource or upload an image or file. If upload is selected the lecturer enters the file. When the form is submitted, the `Upload` class handles the file and uploads to the Material directory.

Once all elements have been entered, the lecturer can select to add other questions or to finish and save the exam. When either option is selected, the parameters build a new `Question` class, inserting the class into the `createExam` bean.

```
Questions q = new Questions(id, title, ... , answers, type);  
examCreate.addQuestion(q);
```

If "Add another Question" is selected, stage two is then repeated until all required questions have been inserted. When "Finish and Save" is selected, all questions are validated and the user is redirected to stage three.

## Stage Three

Once the marking process is complete, the confirmation interface is displayed, showing all exam information and available questions. Questions can be altered or deleted, shown in figure 27.

By clicking on a question title, a brief description of the question is displayed (figure 28). By selecting edit or delete, the user is returned to stage two, with the required question details selected.

Figure 27: Stage Three

**Create Exam - Stage Confirm**

Please enter the following information. All fields are required except where noted. Validation will occur on submit of the data. For more information about a particular element click the ? where applicable for more information.

**Exam Confirmation**

Please confirm that the following information is correct, click **Save Exam** to save the exam making it active or **Cancel** to delete exam.

**Exam Title:** Final Exam  
**Course:** ENG  
**Unit:** COMS12100  
**Exam Date:** 13 May 2006  
**Exam Length:** 01.30  
**Exam Start Time:** 12.30  
**No of Questions:** 4

The following are questions you have added.

No.	Title	Type	
1	Question 1	Multiple Choice	[ Edit ] [ Delete ]
2	Question 2	True/False	[ Edit ] [ Delete ]
3	Question 3	Long Answer	[ Edit ] [ Delete ]
4	Question 4	Multiple Answer	[ Edit ] [ Delete ]

Thank you the exam has been save. Click [here](#) to return to the main Menu.

Figure 28: Question Details

**CreateQuestion**

<b>Title:</b>	Question 4
<b>Type:</b>	Multiple Answer
<b>Question:</b>	Please select one or more values that apply to computer science.
<b>Answers:</b>	0: Can be fun 1: Its complicated 2: Boring 3: Excellent
<b>Material:</b>	http://www.

[ Back ]

[ Close Window ]

Once "Save Exam" has been selected, the createExam bean validates and inserts the data into the database.

## XML Creation

(lecturer/parseXml.jsp)

To create an exam from an XML file, the user first uploads the XML file to the system. The upload is then converted into a DOM tree, broken up and created into a createExam class, and saved to the database. The XML format of the file is strict and must follow a Document Type Definition (DTD), which is a schema that the syntax of the XML must follow. Below is an example part of the XML file used to create an exam.

```

<entity name="exam">
  <attribute name="Exam" value="10020021" />
  <attribute name="Title" value="Exam Title" />
  <attribute name="Date" value="07 May 2006" />
  <entity name="question">
    <attribute name="Title" value="Question 1" />
    <attribute name="Type" value="2" />
    <attribute name="Question" value="Select all that Apply" />
    <entity name="answers">
      <attribute name="answer_1" value="Option 1" />
      <attribute name="answer_2" value="Option 2" />
    </entity>
  </entity>
</entity> ...

```

#### 4.7.1.5 View Exams

##### Exams Overview

The lecturer may view all exams they are associated with. Figure 29 shows the overview screen which displays brief details about each exam. The exam title can be selected to view full details about the exam.

Figure 29: Exam Overview

**Active Exams**

The following are exams allocated to you which have not yet taken place. Click an exam name to find more details about the exam.

Title	Unit	Degree	Date	Time	Questions
<a href="#">Integration</a>	COMS12100	G400	07 May 2006	13.00	4
<a href="#">Final Exam</a>	COMS12100	G400	01 May 2006	21.30	5

[ Previous ] [ Next ]

Page 1 of 1

##### Single Exam View

All information available for a selected exam will be displayed, as shown in figure 30. From the main view page all related features can be accessed, and features such as edit exam, generate or print passwords and marking are allowed. By clicking the question title, the user can view the question, as shown in Figure 28.

Figure 30: Single Exam View

**View Exams**

Please view the following details on your exams. Below the exam details are question details. Click the question title for more information regarding the selected question.

[ Edit ] [ Print Passwords ] [ Marking ]

---

**Title:** Final Exam  
**Unit:** COMS12100  
**Degree:** G400  
**Date:** 01 May 2006  
**Start Time:** 21.30  
**Length:** 01.30  
**No Questions:** 5

The following are questions related to this exam, select a question title for more information.

Title	Type
<a href="#">A Multiple Choice</a>	Multiple Choice
<a href="#">Question 2</a>	True/False
<a href="#">Question 3</a>	Multiple Answer
<a href="#">Question 4</a>	Ordering Question
<a href="#">Question 5</a>	Short Answer
<a href="#">A Multiple Choice</a>	Multiple Choice

#### 4.7.1.6 Edit Exam

The user has the ability to edit all details related to an exam. They are able to edit, delete or add questions, and change any detail such as course or title, as indicated in figure 31.

Figure 31: Edit Exam

**Edit Exams**

The following allows you to edit all parts of an exam. Edit the exam information and click save, or select the [ Edit ] or [ Delete ] options to alter the actual exam questions. To view full details regarding a question please select the question title. Where available select the ? for more information regarding the options and format of element.

Exam Title: Final Exam ?

Degree: G400 - BSc ? Full List

Course: COMS12100 - 20 ?

Exam Date: 01 May 2006 ?

Exam Start Time: 21:30 ? View Times

Exam Length: 01:30 ?

Number of Questions: 5 ?  
(# known)

Title	Type	
A Multiple Choice	Multiple Choice	[ Edit ] [ Delete ]
Question 2	True/False	[ Edit ] [ Delete ]
Question 3	Multiple Answer	[ Edit ] [ Delete ]
Question 4	Ordering Question	[ Edit ] [ Delete ]
Question 5	Short Answer	[ Edit ] [ Delete ]
A Multiple Choice	Multiple Choice	[ Edit ] [ Delete ]

Add Question Save Changes

To prevent multiple calls to the database, the exam is built into a `createExam` class, which allows the user to add or change details using the classes' `get` and `set` methods. When all changes have been made, calling the `update()` function writes the exams changes to the database.

#### Edit Question

By clicking the "[ Edit ]" option (figure 31), the user can edit any question. The question is retrieved from `createExam` using `getQuestion(questionNo)`. When saved the system calls `editQuestion()`, which updates the question in the `createExam` class.

#### 4.7.1.7 Generate and Print Passwords

Before the examination day, the lecturer is required to generate a number of on the day passwords for each student. By selecting "[ Create Passwords ]" the `PasswordGen` class creates a set number of passwords and stores them in the database. Once passwords have been generated, the "[ Print Passwords ]" option is available. When selected, the page prints all passwords related to the exam in blocks, using the JavaScript `window.print()` function.

#### 4.7.1.8 Monitor Exams

(*lecturer/monitor.jsp*)

#### Monitor Overview

The overview allows the lecturer to briefly view any students currently sitting an associated exam. From the overview, the system calls the `SessionCounter` listener, getting all sessions. The sessions are then separated, and if the session exam is related to the lecturer, the data is extracted from the `ExamMonitor` class and displayed (Figure 32).

Figure 32: Monitor Exams Overview

Monitor Exams [\[ Create Sessions \]](#)

Please look at the following students taking the exam.

Student	Exam	Question	Answer	IP	
#0332135	#1000000000	3	3	82.32.44.61	<a href="#">[ View ]</a>
#0332136	#1000000000	2	1	82.32.44.64	<a href="#">[ View ]</a>


### Monitor Single Exam

When a single session has been selected, the student details are retrieved from the ExamMonitor class. Information such as the student ID is then used to query the database to access and display student details. Other attributes from the ExamMonitor class such as exam ID, current question number and the number of questions answered are also displayed, as shown in figure 33.

Figure 33: Monitor Single Exam

Monitor Exams

**Student Information**

Name:	Burke, James	
Email:	jb3135@bristol.ac.uk	
Current Question:	3	
Answered:	3	
IP Address:	82.32.44.61	
Login Time:	08 May 2006 03:34:00	

**Exam Information**

Exam:	#1000000000
Title:	Integration
Course:	G400
Unit:	COMS12100
Date:	07 May 2006
Start Time:	13.00

[\[ Back to Monitor \]](#)

### 4.7.1.9 Marking Front End

(*lecturer/markings.jsp*)

The marking process is split into three sections, creating a model answer, automatic marking and manual marking.

### Marking Overview

When the marking process is selected, the lecturer will see an overview of the marking process. If no model answers exist, the lecturer has the option to create a model answer otherwise they may alter a model answer. If automatic marking has not yet been performed, then the lecturer may proceed, with the manual marking. The interface is shown in figure 34.

Figure 34: Marking Overview

Exam Marking

Welcome to the marking main page. You are about to mark the following exam. If the details are correct you can then choose to auto mark the exam using the model answers, which cuts marking time by more than half, or click mark to manually go through the exam. **Note:** The auto mark only marks certain types of questions, so you will be still required to manually check the questions that cannot be automarked.

Title:	Final Exam
Date:	01 May 2006
Course:	G400
Unit:	COMS12100
Questions:	5

[\[ Create Model \]](#) [\[ Marking Complete \]](#)

*After generating the auto mark answers you will then be able to answer the unmarked questions straight after.*

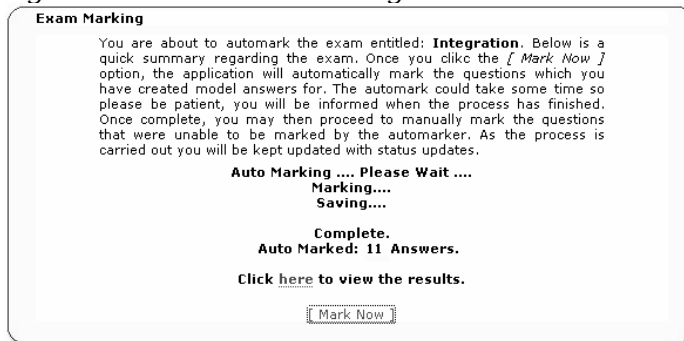
## Model Answers

For the automatic marking process to take place, model answers are required. The Model container class builds model answers automatically. The marking page provides the lecturer with each question where the correct answer is entered. Once the lecturer has submitted all model answers for each of the questions, the Model class then saves the data into the database.

## Automatic Marking

Once the model answer has been generated, the automatic marking process can begin. The automatic marking process requires no user interaction as the AutoMark class handles all aspects mechanically. Once the marking has been completed, the user is informed of the number of automatically marked questions (Figure 35).

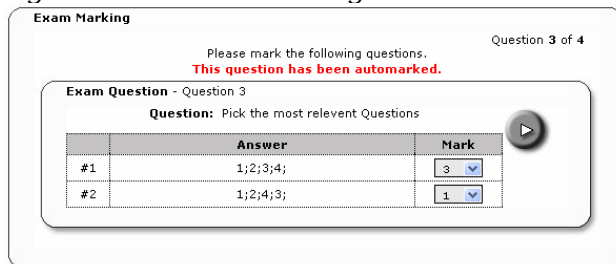
Figure 35: Automatic Marking



## Manual Marking

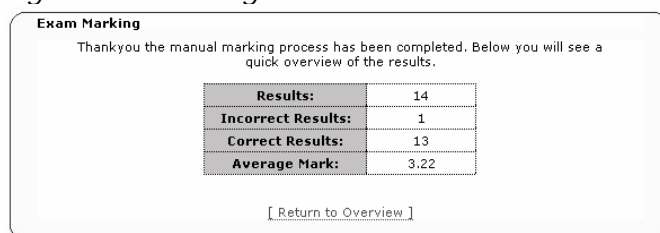
The manual marking process allows the lecturer to manually assign marks. The ManualMarking class retrieves all questions, student answers and automatic marking results building structures to hold the values. The lecturer then works through each question assigning a mark. The automatically marked questions have each student's mark inserted, allowing the lecturer to alter them if necessary (Figure 36).

Figure 36: Manual Marking



Once all answers have been marked, a brief overview of the results are displayed such as average mark and the number of correct and incorrect answers (figure 37)

Figure 37: Marking Overview



## 4.7.2 Tasks

The task system is a basic reminder process. Lecturers can create a task that has to be completed by a certain date. They are able to create, edit and delete tasks. When the lecturer first logs in, the system checks to see if any tasks have expired. Expired tasks are set to inactive and will no longer be displayed. Figure 38 shows the main task overview, showing all activate and inactive tasks. When a task is created, the lecturer inputs the title, task content and a date to be completed.

Figure 38: Tasks Overview

**Current Tasks**

The following are recent tasks that you havent tagged as complete, they are ordered by date. To change the status or content of a task click the title below.

Title	Created	Date	
Task Title	09 May 2006	19 May 2006	[ Delete ]

[ Previous ] [ Next ]

Page 1 of 1

**Task Guide**

The tasks pages allow you to add, edit, delete and view tasks. The aim of the tasks function is allowing you to make notes regarding your exams.

To add a new exam select the "Create" option from the navigation. If you no longer require a task, select the [ Delete ] option next to the task on the left.

On login to the Administration area, you will be able to see any tasks you currently have active.

*Note: It displays the most recent 5 tasks by default, this can be changed in Settings.*

**Archived Tasks**

The following tasks are archived tasks which you have marked as completed.

Title	Created	Date	
You have no Archived Tasks.			

## 4.7.3 Settings

The settings section can only be accessed by users with administration privileges. It allows the system settings to be altered. The Settings bean retrieves settings using get methods and the bean also provides set methods for changing values.

The settings process has four areas; database, display, folder and application settings. Figure 39 shows the folder settings area. When an attribute is changed, the Settings bean's set method is called with the new attribute as a parameter. When a change occurs the new value is inserted into the database.

Figure 39: Change Settings

**Exam Settings**

The following are folder locations used by this application. Changing them incorrectly will point images and options to the wrong place.  
*All paths are relevant to the server. ie /images/ will be http://localhost/images/*

<b>CSS File:</b>	lep_css.css
<b>Image Path:</b>	images/
<b>Number Path:</b>	images/numbers/black/
<b>Material Path:</b>	/student/Material/
<b>Header Image:</b>	images/lep_header.jpg
<b>Student Image Path:</b>	/images/students/

**Note:** If you change the database settings incorrectly, the program may no longer work.

#### 4.7.4 IP Ranges

The IP ranges section, monitors connected IP address for illegal activity.

##### 4.7.4.1 IP Ranges Back End

###### IP Class

(*IEP/Utils/IP.java*)

The IP class holds a structure for an IP; it contains an IP address and the associated computers' location. The class also provides get methods for accessing the data.

###### IP Monitor

(*IEP/Utils/IPMonitor.java*)

IPMonitor loads all allowed IP addresses into memory. When a new computer connects to the system, the IP is passed to IPMonitor and compared with the values in memory. If the address is found, then it is allowed, otherwise it is classed as an illegal IP and the user cannot proceed any further.

```
public boolean check(String ip)
{
    for(int i=0; i < rangeCache.size(); i++)
    {
        IP checkIp = rangeCache.get(i);
        if(checkIp.getIP().equalsIgnoreCase(ip)) return true;
    }
    return false;
}
```

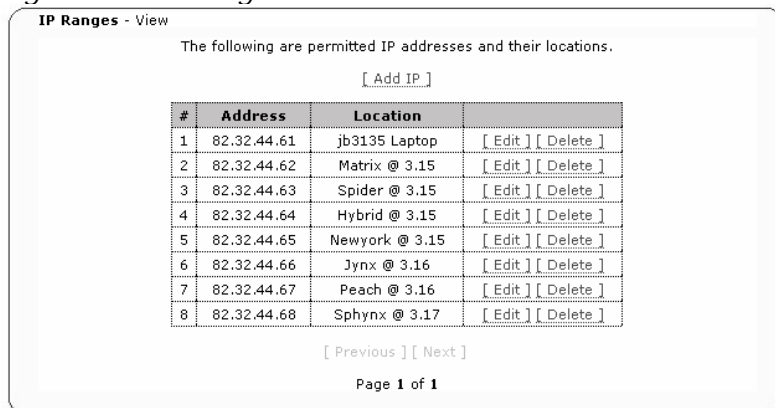
##### 4.7.4.2 IP Ranges Front End

###### Managing Ranges

(*lecturer/ranges.jsp*)

Only users with administration privileges can alter IP ranges. The manager allows you to view, alter or delete IP ranges within the system. Figure 40 shows IP the overview screen.

Figure 40: IP Range Overview



The screenshot shows a web page titled "IP Ranges - View". It contains the text "The following are permitted IP addresses and their locations." and a link "[ Add IP ]". Below this is a table with 8 rows and 4 columns: "#", "Address", "Location", and a column with "[ Edit ]" and "[ Delete ]" links. The table lists IP addresses from 82.32.44.61 to 82.32.44.68 with various locations like "jb3135 Laptop", "Matrix @ 3.15", "Spider @ 3.15", "Hybrid @ 3.15", "Newyork @ 3.15", "Jynx @ 3.16", "Peach @ 3.16", and "Sphynx @ 3.17". At the bottom of the table are links "[ Previous ]" and "[ Next ]", and "Page 1 of 1".

#	Address	Location	[ Edit ] [ Delete ]
1	82.32.44.61	jb3135 Laptop	[ Edit ] [ Delete ]
2	82.32.44.62	Matrix @ 3.15	[ Edit ] [ Delete ]
3	82.32.44.63	Spider @ 3.15	[ Edit ] [ Delete ]
4	82.32.44.64	Hybrid @ 3.15	[ Edit ] [ Delete ]
5	82.32.44.65	Newyork @ 3.15	[ Edit ] [ Delete ]
6	82.32.44.66	Jynx @ 3.16	[ Edit ] [ Delete ]
7	82.32.44.67	Peach @ 3.16	[ Edit ] [ Delete ]
8	82.32.44.68	Sphynx @ 3.17	[ Edit ] [ Delete ]

###### IP Check

(*login.jsp*)

When the user logs into the system, their IP is compared with that of the IP addresses stored in IPMonitor. If the IP is not allowed they will be removed from the system.



## 5 Results

Efficiency and reliability are paramount in this system. Throughout the project every area of the system was thoroughly tested and debugged. With the final product, extensive testing was required.

### 5.6 Security Testing

Many security vulnerabilities have been researched and a solution implemented in the system. However, the only way of ensuring complete security, would be to test each part thoroughly.

#### 5.6.1 Accessing Resources

The system has been clearly spit into three sections, student, lecturer and administration. Only members with the correct privileges can access the resources within. To ensure the resources were protected the following test were carried out:

- When logging in as a student, the only resources available to you are within the student folder. By attempting to access the lecturer and administrator resources the server should deny permission. The result proved that the student was unable to access resources outside of their domain.
- Finally, by logging in as a lecturer and attempting to access the administration resources. The result again, showed that the lecturers' privileges did not give permission to the administration resources.

#### 5.6.2 Secure Server

The only available resource outside of secure server allows the user to login therefore all other resources can only be accessed from inside. To test that both the server and security procedures in place are implemented correctly, resources were accessed from outside the secure environment. The result was correct, in that the resource was denied access and system was redirected to the login page.

#### 5.6.3 SQL Attacks

To ensure the SQL injection attack prevention methods were working correctly, an SQL parameter was altered to contain "1'; drop database iep;" this potentially would delete the entire database. When the query was executed, the correct parameters were removed, but an unplanned side effect occurred. As the string literal was removed, the parameter then consisted of "1; drop database iep". Although this would do no harm to security, it would potentially add the parameter to the database when simply "1" was required. The procedure was therefore altered to simply terminate the query when a threat detected.

#### 5.6.4 IP Checks

To ensure the IP monitor was implemented correctly, tests were performed from IP addresses that were not permitted. The result was as expected with the system detecting the error and redirecting to the login page.

### 5.7 Student Testing

The main testing required for the student section, involved testing the interface for usability flaws and the Exam class for efficiency problems.

### 5.7.1 Interface Testing

The student interface was required to be efficient and durable, making the system easy to use and prevent human error. When the first interface prototype was created, batch testing was carried out by a number of students. Their feedback provided further restrictions to the interface making the interaction relaxed and safer. Once the batch testing had been carried out a well-organised, easy to use interface was implemented. "Random" tests were then carried out on the system. By simply clicking all available buttons, making every mistake possible, further flaws were found. One such defect was that the `Build` class would not implement correctly if called twice by moving between questions before the last instance had finished loading. The solution consisted of disabling all buttons once an action had been selected until it was complete.

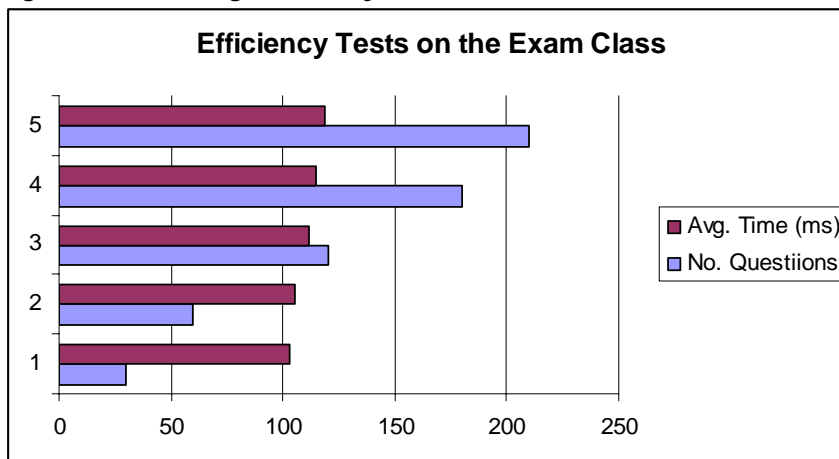
### 5.7.2 Exam Class Testing

The `Exam` class holds all exam questions and data. If the exam is large, the great number of questions could prove an efficiency risk or the class simply could not handle the work load and crash. To test the efficiency, a Java class was created, which generated a new exam, then in a loop added 30 questions consecutively, followed by the retrieval of each of the questions. A timer was used, to calculate the time taken to process the data.

```
Time.start()
for(int i=0; i < 30; i++)
{
    Question q = new Question();
    Exam.addQuestion(q);
}
System.out.print(Time.finish());
```

It was possible, that the data from a few tests might not be conclusive, but after running the program repeatedly, incrementing the number of questions added and retrieved. Evaluation of the outcome was irrefutable, showing that the class always executed within an efficient time period proving that the class was always reliable, shown in figure 41. The values used to create the result has a sample size of 20, the number of questions is incremented by 30 after each set.

Figure 41: Testing Efficiency Results



As the graph shows that when the questions grow large, the average process time does not increase exponentially, but stays within an acceptable range proving the class was built in an efficient manner.

## **5.8 Other Tests**

The system tests outlined above were only a handful of the tests carried out. Each aspect of the system was thoroughly tested ensuring all data was correctly validated, database entries and queries were successful and the system was as reliable as possible. Each aspect of creating an exam using forms of XML was tested exhaustively to ensure security vulnerabilities or validation problem did not occur.

## **6. Conclusion and future work**

### **6.1 Achievements**

The system produced, covered all aims and objectives set out for this project. The final system is an efficient and secure Online Examination Program. All features discussed in the design stages have been successfully implemented. After exhaustive testing the forms and structures are extremely efficient and reliable. The security methods implemented are fully operational, achieving their requirements, after testing all security elements are implemented correctly providing a completely secure system.

### **6.2 Problems**

Although the system provided met all the required objectives, this did not happen without set-backs. Some of the main problems are outlined below:

- Java Null Pointer Exception. When working with web applications, if you get a null pointer, it is extremely hard to track down the root, taking large amounts of time to debug and remedy.
- Sessions Monitor. The session monitor was initially hard to implement, problems occurred when a session was detected that was not of the ExamMonitor type. This caused exceptions in the listener which intern cause Tomcat to error. To overcome this problem, the system checks the session object at runtime, if it is an ExamMonitor the session is then processed.
- File Upload. The file upload process was initially straight forward, by parsing the file then retrieving the data. Problems occurred when attempting to rebuild the file. Eventually I created a FileInfo structure that contained the file content which will then be used to construct the file.

### **6.3 Future Developments**

There are many possible future developments for this system. The secure shell aspect researched initially could be implemented freezing all other applications except IEP. A more advance automatic marking system could be produced, which handles part string matching and key word finding. More question formats could be included, for example mathematical calculations or more complex formats could be incorporated. The displaying of results could be enhanced, providing more graphical interfaces or format styles. A more complex future development work would be to possibly enhance the monitoring process. Rather than getting a snapshot of the student exam, the actual Exam class could be retrieved allowing the lecturer to physically view student answers at real time, providing a more in-depth monitoring process.

### **6.4 Summary**

The whole lifecycle of this project has proved a success, from the research and design stage through to implementation and testing. I have stuck to my goals, aims and schedule successfully. I have gained valuable experience with the whole development process, and am very proud of the final system. The project has highlighted areas of website development that cannot be seen when creating a small application. Due to the system scale, tracking and security proved a lot harder than with a simpler system. The resulting project provides both theoretical and practical experience which will prove beneficial in the future.

## References

1. Blackboard & WebCT. [No Date] Online CMS System. [Online] [Accessed 2006 April] Available from URL <http://www.blackboard.com/>
2. Northumbria University. [3<sup>rd</sup> January 2006] Blackboard Known Issues. [Online] [Accessed 2006 April] Available from URL <http://northumbria.ac.uk/sd/elearning/issues/?view=Standard>
3. Richard Healy.com [No Date] Richard Healy Test Builder. [Online] [Accessed 2005 September] Available from URL <http://richardhealey.com/testbuilder/>
4. ExamWeb Assessment Technologies [No Date] Online Assessment Software. [Online] [Accessed 2006 May] Available from URL <http://www.examweb.com/>
5. SAOL.com. [No Date] A glossary of commonly used Internet terminology. [Online] [Accessed 2006 March] Available from URL <http://www.saol.com/glossary.asp>
6. Wikipedia [8<sup>th</sup> April 2006] C Programming Language. [Online] [Accessed 2006 April] Available from URL [http://en.wikipedia.org/wiki/C\\_programming](http://en.wikipedia.org/wiki/C_programming)
7. About.com [26/5/2004] ASP Programming. [Online] [Accessed 2006 February] Available from URL <http://experts.about.com/q/Active-Server-Pages-1452/asp-virtual-path.htm>
8. Java Sun [No Date] JSP Tag Libraries. [Online] [Accessed 2006 March] Available from URL <http://java.sun.com/products/jsp/taglibraries/>
9. domainurus.com [No Date] MySQL Database. [Online] [Accessed 2006 April] Available from URL [http://www.domainurus.com/faqmasterflex/faq.php?answer=57&cat\\_name=MySQL%20Database&category\\_id=8#57](http://www.domainurus.com/faqmasterflex/faq.php?answer=57&cat_name=MySQL%20Database&category_id=8#57)
10. About.com [No Date] Oracle Definition. [Online] [Accessed 2006 January] Available from URL <http://databases.about.com/cs/oracle/g/Oracle.htm>
11. Wikipedia [April 2006] Internet. [Online] [Accessed 2006 April] Available from URL <http://en.wikipedia.org/wiki/Internet>
12. University Wisconsin Teaching Academy [No Date] Exam Question Types & Student Competencies [Online] [Accessed 2006 February] Available from URL <http://wiscweb3.wisc.edu/teaching-academy/Assistance/course/questions.htm>
13. Wikipedia [No Date] Taxonomy of Education Objectives. [Online] [Accessed 2006 February] Available from URL [http://en.wikipedia.org/wiki/Bloom%27s\\_Taxonomy](http://en.wikipedia.org/wiki/Bloom%27s_Taxonomy)
14. Ted Power [No Date] Language Testing and Methods of Assessment. [Online] [Accessed 2006 February] Available from URL <http://www.btinternet.com/~ted.power/esl0704.html>
15. Apache Jakarta Project [No Date] Jakarta Commons Proper. [Online] [Accessed 2005 December] Available from URL <http://jakarta.apache.org/commons/>
16. World Wide Web Consortium [12<sup>th</sup> April 2006] W3C XHTML and CSS standards. [Online] [Accessed 2006 January] Available from URL <http://www.w3.org/>
17. Web Developers Virtual Library [No Date] Web Databases. [Online] [Accessed 2005 December] Available at URL <http://www.wdvl.com/Authoring/DB/>
18. Wrox Press Inc [February 1<sup>st</sup> 2003] Professional JSP 2<sup>nd</sup> Edition [Book] Brown, Burdick, Faulkner, et al. [ISBN: 1861004958]