School of Computer Science and Information Systems
Birkbeck College
University of London

Web-based coursework submission system

Sokratis Karkalas

December 2000

TABLE OF CONTENTS

GLOSSARY **4**

ABSTRACT **5**

INTRODUCTION **5**

DESCRIPTION OF THE SYSTEM AT ITS CURRENT STATE 6

1. DESIGN OF THE SYSTEM **11**

1.1 ARCHITECTURE 11
1.2 IMPORTANT DESIGN ISSUES 12
1.2.1 Efficiency *12*
1.2.2 Security *14*
1.2.3 Access *14*
1.2.4 Organisation and management of the data *15*

2. IMPLEMENTATION **15**

2. IMPLEMENTATION **16**

2.1 BACK END 16
2.1.1 Advantages of using Oracle Server as the back end component *16*
2.1.2 Disadvantages of using Oracle Server as the back end component *16*
2.2 MIDDLE TIER 17
JavaServer™ Web Development Kit (JSWDK) *17*
2.2.1 Advantages of using JSWDK as the middle tier component *17*
2.2.2 Disadvantages of using JSWDK as the middle tier component *17*
2.2.3 Advantages of using Java Servlets as part of the middle tier component *18*
2.2.4 Disadvantages of using Java Servlets as part of the middle tier component *19*
2.2.5 JDBC – OCI driver *20*
2.3 FRONT END 20
2.3.1 Advantages of using JavaScript for development at the client end *20*
2.3.2 Disadvantages of using JavaScript for development at the client end *21*

3. CONCLUSIONS – SUGGESTIONS FOR FURTHER DEVELOPMENT **22**

REFERENCES **24**

APPENDIX 1: IMPLEMENTATION – THE BACK END COMPONENT **25**

SQL DDL STATEMENTS USED FOR THE CREATION OF THE DATABASE 25
SQL DML STATEMENT USED FOR UPDATING THE DATABASE WITH STUDENTS' INFORMATION 26

APPENDIX 2: IMPLEMENTATION – THE MIDDLE TIER COMPONENT **27**

JSWDK CONFIGURATION 27
contents of webserver.xml *27*
contents of mappings.properties *27*
contents of servlets.properties *28*
JAVA CODE – REPRESENTATION OF THE DATABASE (DATABASE.JAVA) 29
JAVA CODE – REPRESENTATION OF THE FORM SUBMITTED BY THE USER (HTMLFORM.JAVA) 31
JAVA CODE – SERVLET THAT RETURNS, ON REQUEST, STUDENT DETAILS AS AN HTML TABLE (STUDENTDETAILS.JAVA) 33
JAVA CODE – SERVLET THAT FORWARDS, ON REQUEST, ASSIGNMENT DETAILS TO THE DATABASE (STUDENTDETAILS.JAVA) 37

APPENDIX 3: IMPLEMENTATION – THE FRONT END COMPONENT 47

HTML CODE USED FOR LOADING THE INTERFACE (HOME.HTM) 47
HTML CODE USED FOR THE ONE AND ONLY FRONT WEB PAGE (ASSIGNMENTS.JSP) 48
JAVASCRIPT CODE USED FOR BROWSER CHECKING AND LOADING THE FRONT PAGE (BROWSER.JS) 50
JAVASCRIPT CODE USED FOR INPUT VALIDATION AND DYNAMIC SCREEN ALTERATION (UTILS.JS) 51
JAVASCRIPT CODE USED FOR DATABASE IMPLEMENTATION (DATABASE.JS) 55
CSS CODE USED FOR THE APPEARANCE OF THE INTERFACE (STYLE.CSS) 58

APPENDIX 4: USAGE INSTRUCTIONS GIVEN TO STUDENTS 59

GLOSSARY

C++ (C plus plus):	A general purpose (compiled) programming language.
JavaScript:	A general purpose (scripting) programming language.
Java:	A general purpose (semi-compiled) programming language.
Web-based system:	A system that uses the World Wide Web as its platform.
SQL (structured query language):	A declarative special purpose programming language that specialises in database definition and manipulation.
API (application programming interface):	A software component that can facilitate the programmatic interaction between the application that it belongs to and an application that requires its service.
URL (uniform resource locator):	A fully qualified reference to a resource that is available on a networked computer system. This reference specifies not only the name of the resource and its path on the local system but also the name of the machine it resides on.
HTML (hypertext mark-up language):	A declarative special purpose (scripting) programming language that specialises in document definition for the World Wide Web.
XHTML (extended hypertext mark-up language):	A constrained specialisation of HTML.
Servlet:	A software component written in Java that is used either as part of a web-based application or as an enhancement of a web server.
JSP (Java server pages):	A specialised version of a servlet that can incorporate HTML and be compiled dynamically by the web server on demand.
DBMS (database management system):	A computer system (software and/or hardware) that specialises in database management.
JDBC (Java database connectivity):	A DBMS-neutral programming interface through which client applications can access and manipulate a relational database.
ORACLE OCI driver:	A JDBC implementation by Oracle.
ORACLE THIN driver:	A JDBC implementation by Oracle.
CSS (cascading style sheets):	A declarative special purpose (scripting) programming language that specialises in style definition for documents of the World Wide Web.
LAN (local area network):	A group of networked computers and sharable devices that cover a small geographic area (e.g. an office)
WAN (wide area network):	A group of networked computers and sharable devices that cover a big geographic area (e.g. a city, country, continent).
HTTP (hypertext transfer protocol):	A connectionless protocol that is used at the application layer for exchanging resources (like text or images) between different sites on the Internet.
TCP-IP (transmission control/ Internet protocol):	A set of rules that is used to form the basic language (protocol) that facilitates communication on the Internet. This language cannot be used directly by humans, but it is the language used by the computer components that interact using the Internet as a platform.
GUI (graphical user interface):	A graphical (logical) representation of a computer system's interface to its user.
DOM (document object model):	The DOM is a platform and language independent interface through which programs can dynamically manipulate the content, structure and appearance of a web document. Currently, there are DOM extensions implemented by most popular web browsers.

Abstract

This report describes the design and implementation of a web-based coursework submission system. This system was developed for the School of Computer Science and Information Systems of Birkbeck College at the University of London. The specification for the system was set by the author of this paper in co-operation with the course director of the MSc in Computing Science Dr. Roger Mitton. The final product was intended to be used to assist the coursework evaluation of the course 'Programming in C++' which is one of the main courses of the MSc. The number of students that take this course each year is approximately 150 and the number of assignments they have to submit is 6.

The specification of the system can be summarised in the following paragraph:

This is intended to be an electronic coursework submission system that will gradually replace the old-fashioned paper-based submission. The purpose of the system is to enable students to upload electronically the assignments to a central repository using a simple web-based interface. The system has to be efficient, transparent and simple to use. The students should be able, having given their username to the system, to retrieve their personal details, including full name and mode of study, and to submit their assignment as a text file along with their username and the number of the assignment to the system. The retrieval of the user's details prior to the submission of the assignment serves as a confirmation that the user has used the correct username to communicate with the system. After a successful submission, the system should respond with a confirmation message, which the students ought to save for future reference.

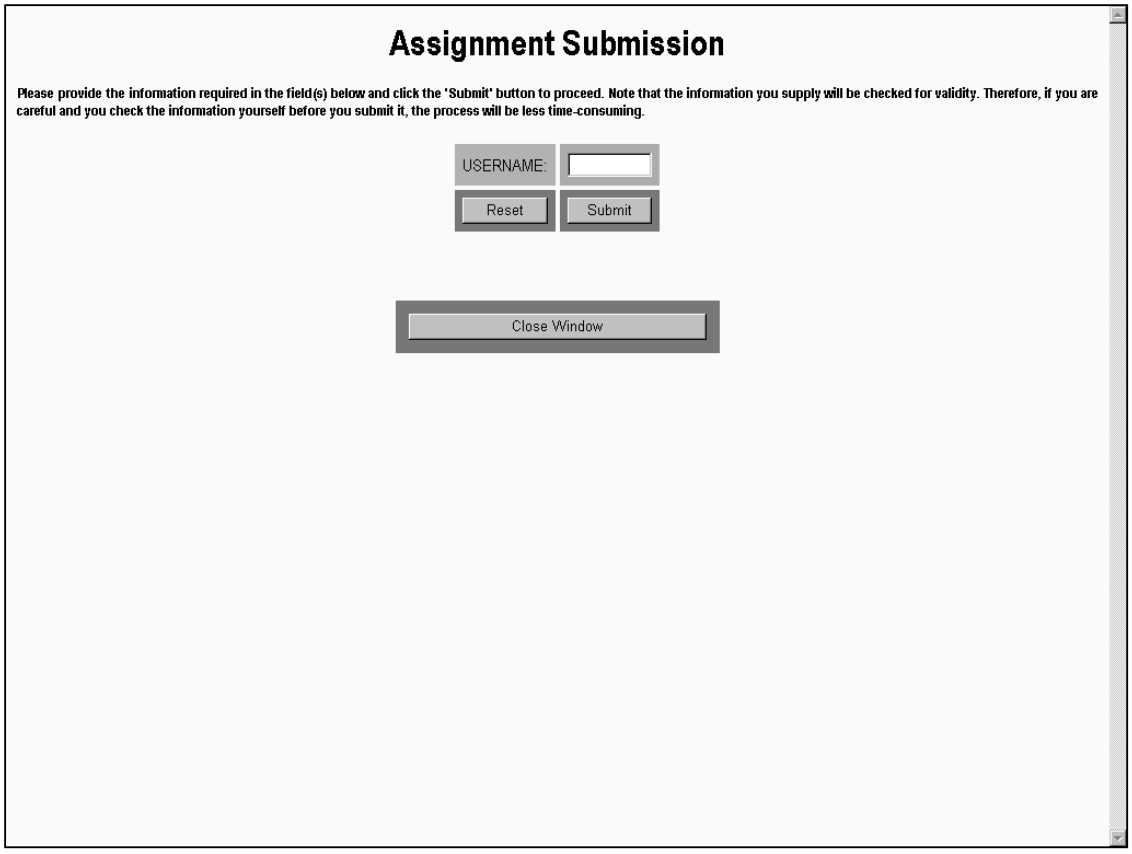
Introduction

When I designed this system my intention was not to provide a specialised application to facilitate only the submission of C++ assignments but a framework within which software components could be easily integrated with each other and existing functionality could be reused and extended. This project is therefore not the final objective, but the first step towards a component based system to deal with coursework evaluation. The functional orientation that I gave to the system is not constraining its potential. Since, the primary design goals were extensibility, scalability and reusability, the system can be extended in any direction and existing components can be reused for different purposes and by different systems and groups within the school (e.g. administration).

A fully-fledged coursework evaluation system is expected to provide mechanisms to facilitate submission, processing, retrieval and assessment of coursework. The only part of the system that has been implemented so far is the coursework submission component, which can be used independently and enhanced with new components as soon as these components become available. Due to the modular nature of the system, the dynamic enhancement of its functionality will not affect its availability. As new components are added to the system more functionality will become available and that will happen without compromising the availability of existing components.

Description of the system at its current state

One of the big advantages of the system is the simplicity of its interface. The user can communicate with the system and request the service from any terminal that is connected to the Internet. After, the submission of the URL associated with the service the user is presented the following screen:



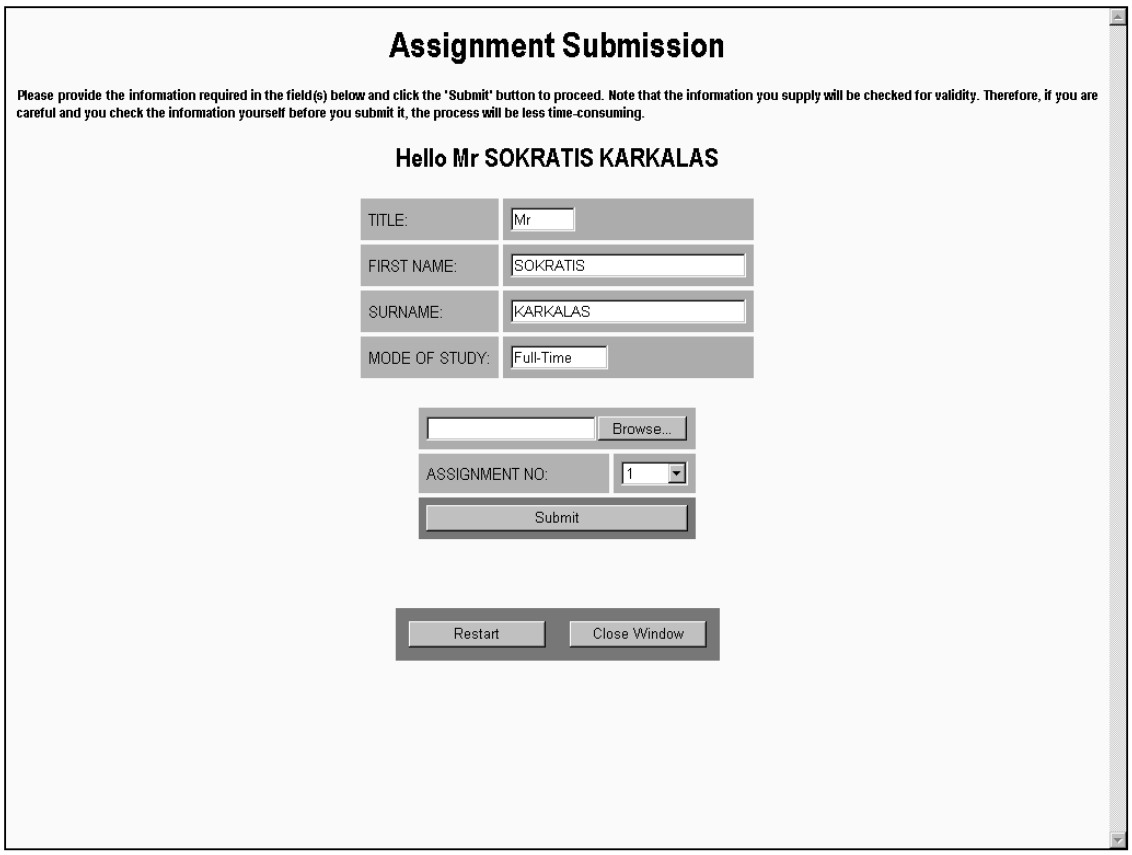
Note that this is a borderless window that occupies the entire screen and there is no menu bar and tool bar available. If the user attempts to click the right mouse button in the client area of the window the following dialog window appears:

The reason for all this strictness is security. All these things are measures taken to prevent the user from getting access to the underlying code and investigating the mechanics of the system.

The next thing the user is required to do is to type the username in the corresponding field. If the user has made a typing mistake, the 'Reset' button can be used to remove the wrongly typed username. The user can then replace it with the correct one. If the correct username has been inserted, then the user can press the 'Submit' button to proceed to the next stage.

If the user is not registered with the system then the system responds with the following dialog window.

If the user is registered with the system, then the following screen is presented.

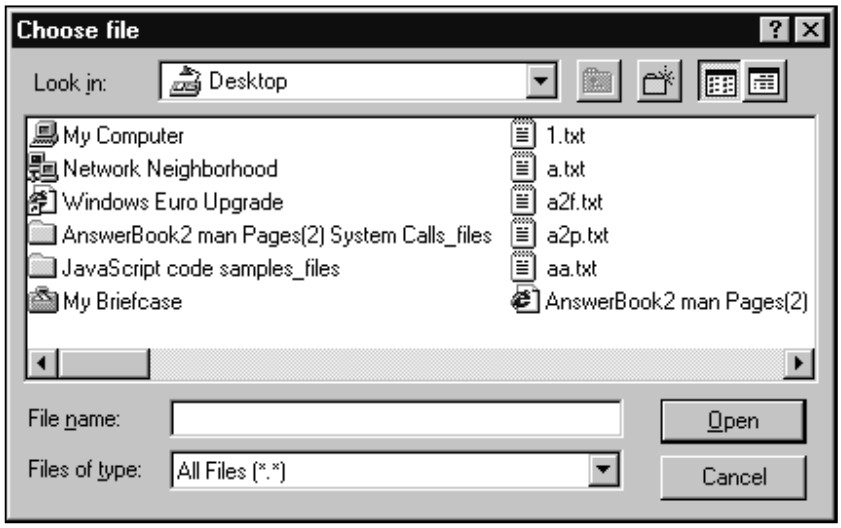


The first four fields serve as a confirmation that the user has submitted the correct username. If the information in these fields does not correspond to the personal details of the student, then obviously the student has submitted a 'legal' username, which corresponds to a different student. In this case the student is advised to press the 'Restart' button to go back to the first screen and submit the correct username.

If the information is correct, then the user should press the 'Browse…' button to browse the file containing the assignment on the local file system.

If the user decides to submit the assignment without having selected a file using the above button, the following dialog window appears.

If the user follows the instructions given and presses the 'Browse…' button, the following window is presented.



Using this window the user can visit the logical location on the local file system at which the file containing the assignment is stored and select the corresponding filename. Immediately after that the filename along with its full path is displayed in the field next to the 'Browse…' button.

The next thing the user is required to do is to select the number of the assignment from the drop down menu located below the 'Browse…' button and press the 'Submit' button to send the information to the server.

The system at that stage may object to provide the service, in case the filename selected is not a permissible one. The system expects the filenames to have extensions .cpp or .C. If that is not the case, then the following message appears.

The purpose of this check was to prevent users from uploading the wrong files to the system. If the extension is a permissible one then the upper portion of the window is replaced by spinning hourglasses and a message informing the user that the uploading process is in progress.

Finally, when the uploading process is completed the upper portion of the window is replaced with the confirmation page sent back by the server. This confirmation window is shown in the following screenshot.

The user, at that stage is required to press the button 'Print Confirmation Page' to print out the submission details of the assignment. The hard copy of the confirmation needs to be retained by the student for future reference. The student then can either close the window or press the 'Restart' button in order to repeat the process again for another assignment.

1. Design of the System

1.1 Architecture

Since this is a web-based application intended for a relatively low volume site, a simple three-tier architecture has been used. N-tier architectures have become the most popular choice for Internet-centric applications because of the advantages they offer against old-fashioned two-tier architectures in terms of performance, flexibility, maintainability, reusability, and scalability. N-tier architecture implies the adoption of the thin-client model, which means the separation of interface and process management where the business logic and rules are executed. Furthermore, the effective use of sharable resources at the middle tier(s) means higher performance and increased availability when at the same time the complexity of the distributed processing is hidden from the user. The three main tiers of this system are presented in the following figure.

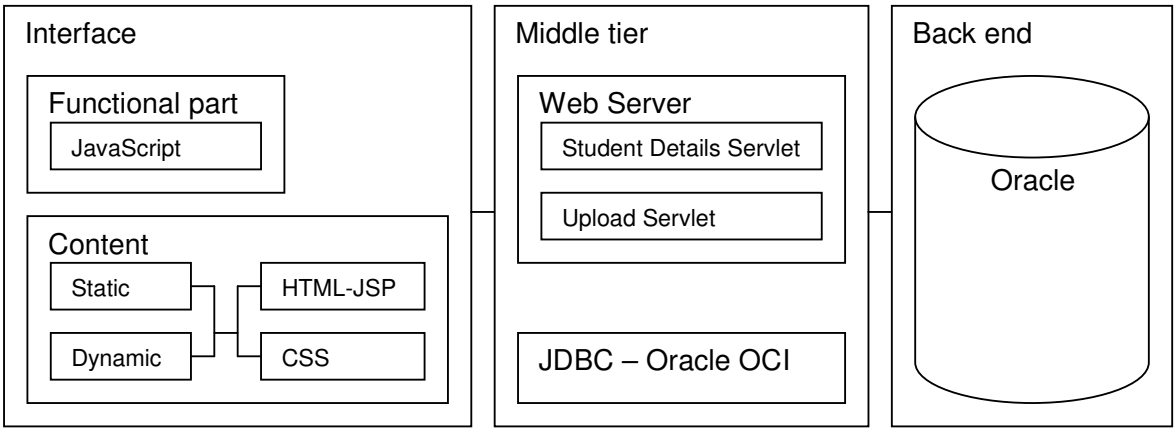


Figure 1 – Architecture of the system

The interface (front end tier) is the component through which the user interacts with the system. It comprises a functional and a non-functional part. For the functional part JavaScript is used. The non-functional part comprises HTML and CSS code which is sent by the middle tier to the browser. Part of that code is hard-coded HTML that resides on the middle tier and serves as a template whereas the rest of it is dynamically formed HTML sent by the servlets. A servlet is a software component written in Java that resides on a web server and implements the application logic or enhances the server's functionality. The content of the interface, in most of the cases, using the functional part, can dynamically change depending on the user's instructions on the client end, without any interaction with the middle tier.

The middle tier comprises the web server, the application integrated with it and the Oracle JDBC component needed for interfacing the applets with the back end tier.

The back end tier comprises the Oracle Database Management System and the database created for the application.

1.2 Important design issues

1.2.1 Efficiency

One of the main considerations that were taken into account during the design process was the efficiency of the system in terms of response time. This is usually the main and most important problem that every web-based system suffers from. There are two parameters that affect performance most. The first is the distribution of workload and the second is the network traffic. If, for instance, the middle tier is very busy and has reached the point where it cannot handle more requests, its performance degrades because of the resources taken up for answering back these requests. Things become worse when the users that are denied the service, frustrated, try hitting their buttons multiple times requesting the same thing more times than necessary causing the server to reserve even more valuable resources for handling them. An obvious solution to this problem is to transfer part of the workload from the middle tier to the other two tiers. For example, if the application is database-processing intensive, the back end tier should perform, as much processing as possible and only results should be sent to the middle tier. This was not a problem in this system since the database processing that is required is not very heavy. Another optimisation is to transfer part of the workload to the front end. That means that the server needs to know in advance what data the client is going to need to send them all as a single packet in response to the first request. After receiving the data the client can do all the processing locally without using the network and any resources on the server. When the client is ready to finish the job it can send the final request along with the data and receive back a receipt from the server. This approach reduces network traffic significantly since it requires only two TCP-IP connections for each session. This is the model I used for this system. In this system the application of this model was particularly easy since there is no need to keep a user profile in order to know what he/she is going to need for local processing. All that is needed at the client end is a database table with the students' details. This table is sent to the client in the form of HTML and gets transformed locally to a JavaScript object. This JavaScript object is then available for queries/searches that can be performed locally. The only time that the user needs to communicate again with the server is when the request to upload the data to the database is sent. After that, if the operation is successful, the server sends back a confirmation and the session terminates. If the user wants to send more than one assignment in succession, then the system becomes even more efficient, because the database table that was sent after the initial request can be reused and there is no need for it to be transferred again for every session. Finally, the different screens that the user is presented with at every different stage of the process are all part of the same HTML document. The user has the illusion that a request is sent to the server and a response is sent back but the reality is that all the processing again is carried out locally and the transition from screen to screen is done by hiding and displaying different parts of the same page to the user. The result of that is another plus in terms of efficiency.

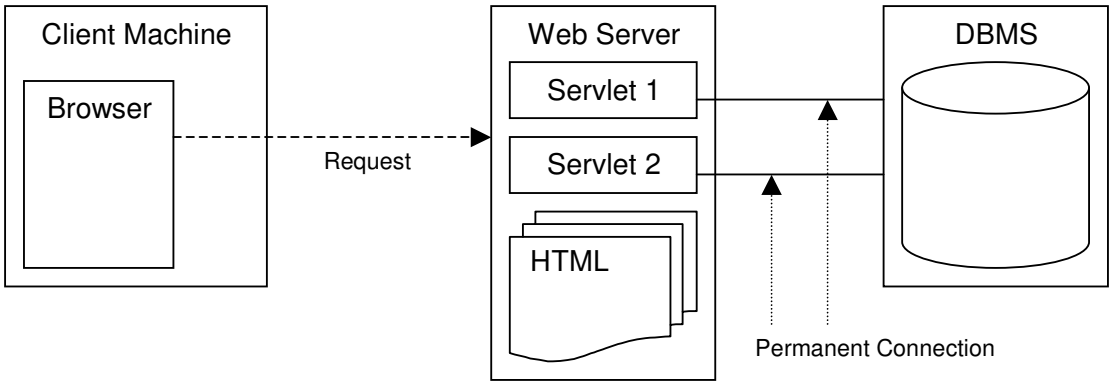


Figure 2 – Phase 1: Request for the service

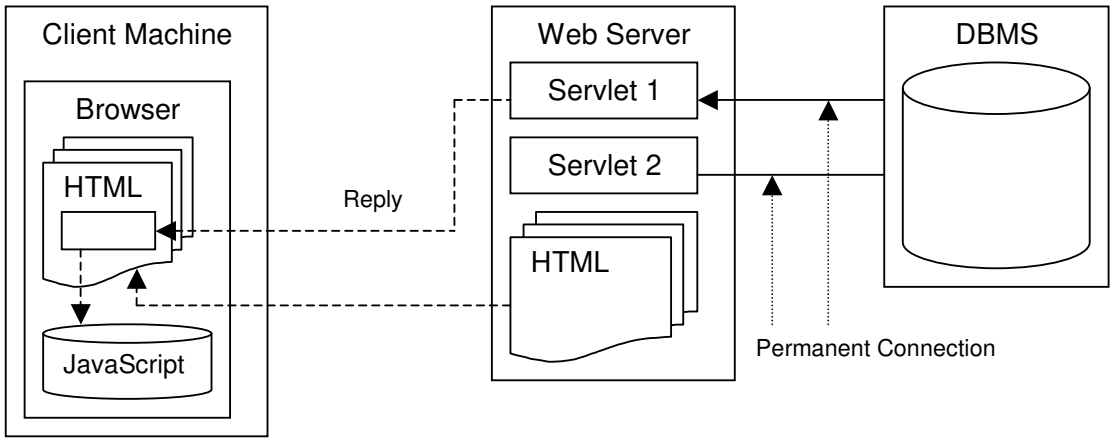


Figure 3 – Phase 2: Formation of the interface.

- Static HTML is sent from the web server to the client.
- A database table is extracted from the DBMS and sent by the servlet to the client in the form of an HTML table.
- The JavaScript program extracts the data from the HTML table and instantiates a Database object (local database).

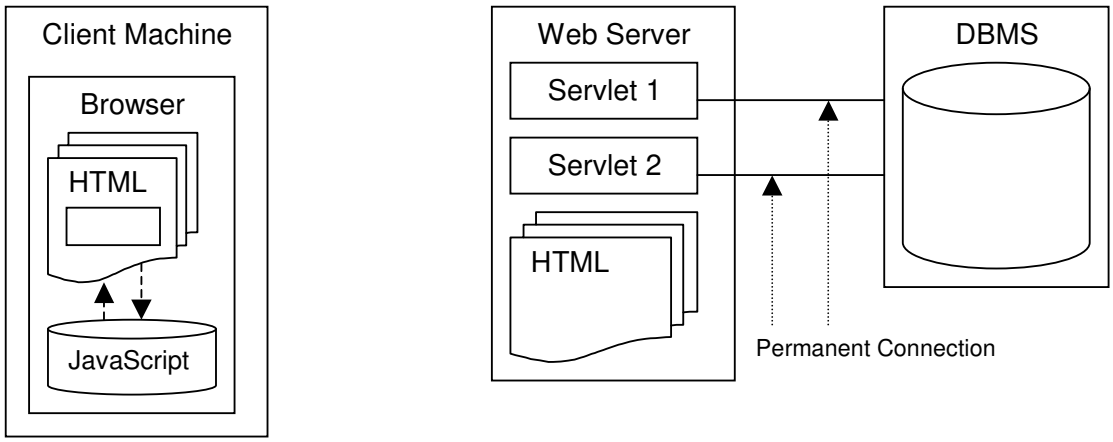


Figure 4 – Phase 3: Client (local) processing.

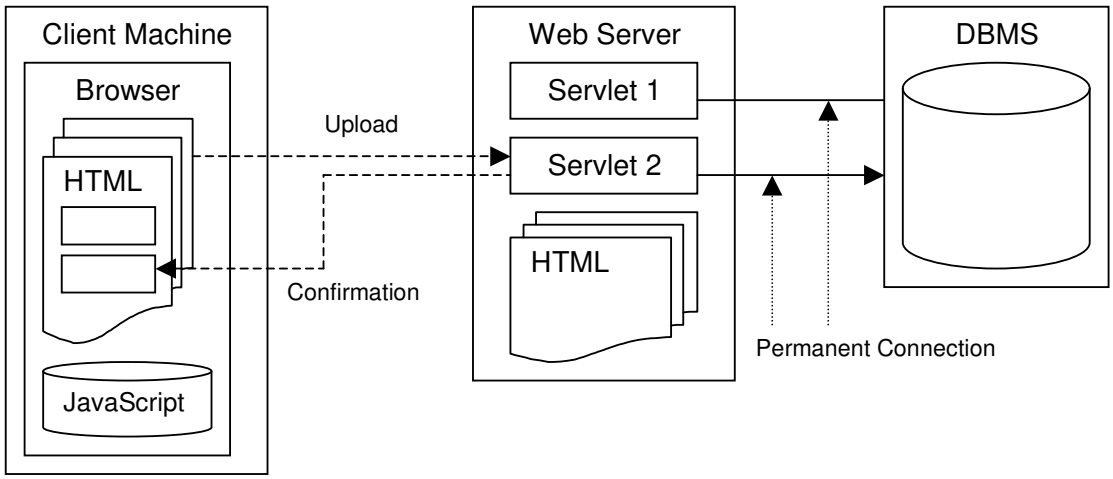


Figure 5 – Phase 4: Completion

- Request to upload along with the data is sent by the client to the servlet.
- The servlet forwards the data to the DBMS and sends confirmation in HTML form to the client.

1.2.2 Security

Another important consideration was the security of students' personal information. The system should be able to prevent unauthorised access to the database at the back end. If a malicious user could gain access to the database, then he would be able to query the database and find out who has submitted what and when and possibly alter the state of the database. Under the Data Protection Act, the data user (the department) has an obligation to protect such events. Since Java was used at the middle tier component for exchanging information with the database server, this problem became even more serious (see 2.2.4). The solution was two fold. First, I decided the username and password needed for establishing a connection with the database should not be hard coded in Java but rather should be retrieved dynamically from the web server during the servlet initialisation process. The second decision was to block all the ways through which the user can have access to the underlying HTML, JavaScript and possibly Java code. JavaScript itself was used for this. The result was to load the front page in a frameless window without menu bar, toolbar and status bar and make that window occupy the entire screen. Furthermore, the right mouse key is disabled and therefore there is no way the user can have access to the source code from the context menu.

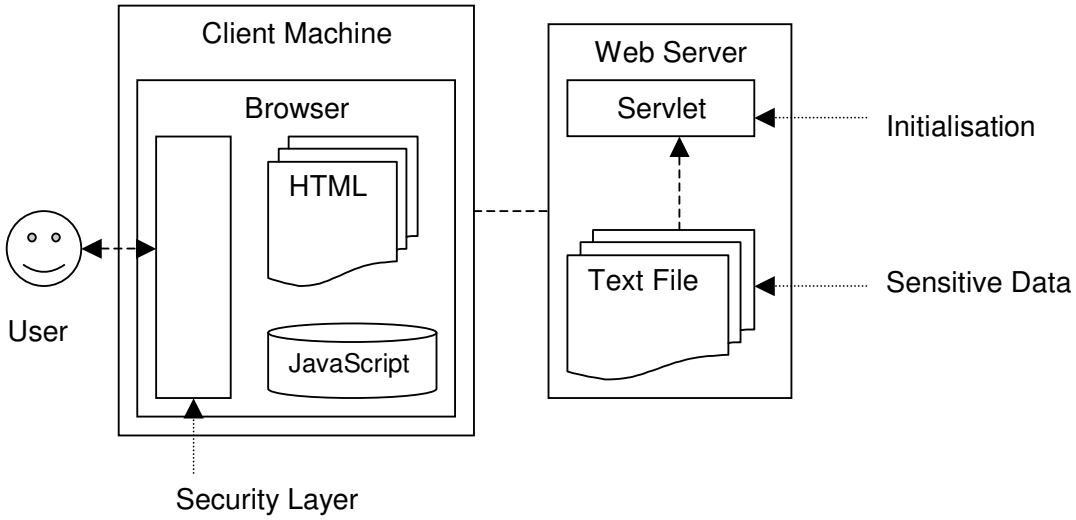


Figure 6 – Security mechanism

- The very first time a request is sent, the servlets on the web server get initialized. On initialization, they retrieve all the sensitive data dynamically from the server.
- After a request for the service is sent, a combination of HTML, CSS and JavaScript is sent back by the server. During the formation of the web page, the browser gets instructed to disable and hide the features of the window that contains the page through which the user can access the underlying code. This is the security layer.

1.2.3 Access

The third and equally important consideration was the simplicity of the interface. Special attention was given to this aspect of the system in order for the users to have a fast and intuitive service. It is important for the user to feel that there is control over what the system does and to know what the progress of the job is at every step. Also, it is important for the user to receive, automatically and synchronously with the submission of

the information to the server, a confirmation that the job has been completed successfully. All these things were taken into consideration during the design phase of the system.

1.2.4 Organisation and management of the data

There are many alternative methods to organise the data needed for such an application at the back end component of the system. The decision to use one of them is not related only to organisational considerations but it may have implications in other aspects of the design like security. If, for instance, there is a database management system (DBMS) used at the back end for storing the data, a security policy can be easily enforced using the native security mechanisms of the DBMS itself. Transaction management is another important issue, especially when according to the specification of the system it is possible for the database to be concurrently accessed and used by more than one client. In this system it is possible for the database to be concurrently used by one or more Java servlets and for different purposes. Furthermore, it may be necessary for the same database tables to be used by administrators or other external programs for different purposes and through different interfaces. It would have been a very difficult and time-consuming task to develop a transaction management system to manage efficiently and securely all these tasks if I had decided to use a semi-structured or a non-structured form to model the data store. Since all this functionality is available in existing systems and has been thoroughly tested for many years in real world implementations, it seems to be sensible and economical to reuse it wherever possible. In addition to the above, easy insertion, deletion and retrieval of information from a DBMS are a few more reasons that support the above view.

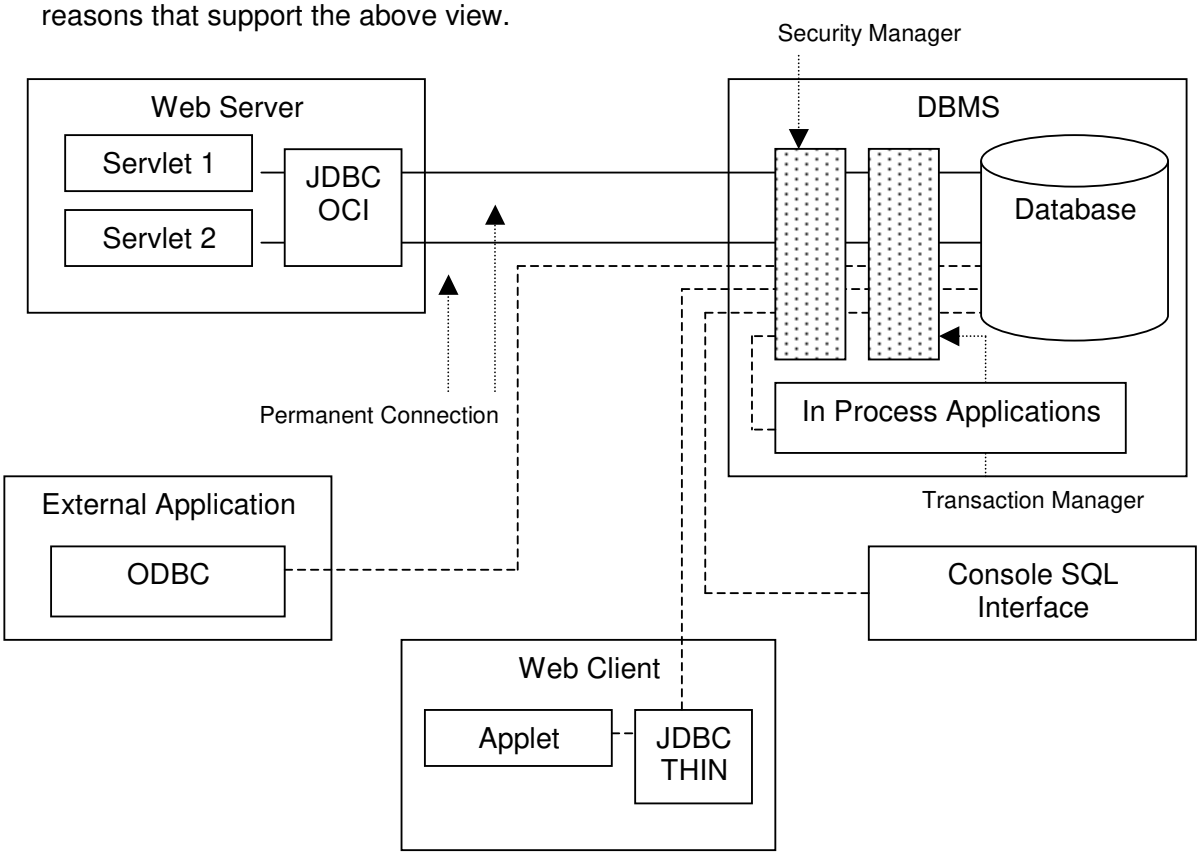


Figure 7 – Advantages for using a DBMS at the back end

2. Implementation

2.1 Back end

At the back end of this system there is an Oracle v.8 database management system, which is used to store into relational tables students' personal details and information about assignments. The assignments themselves are also stored as large binary objects in the database.

Integrity constraints have been used as part of the definition process where appropriate. For flexibility, I used *named* constraints, which can be easily removed or modified without removing the whole table from the system. Special attention has been paid to the USERNAME attribute. A trigger is provided along with the tables to ensure that the username inserted into the system is compliant with the College's username policy.

2.1.1 Advantages of using Oracle Server as the back end component

Flexibility: There is a JDBC interface provided with the server, which means that the system can be easily used by any Java program (either stand-alone or applet) from any computer node that is part of the same Local Area Network (LAN) or a Wide Area Network (WAN).

Availability: The Oracle system was the only relational DBMS with a JDBC interface available in the Department at the time of the development.

Reliability-Efficiency: Oracle DBMS's are known to be very reliable and efficient.

2.1.2 Disadvantages of using Oracle Server as the back end component

Development: Unfortunately, Oracle decided - not wisely - to extend the standard JDBC libraries provided by SUN Microsystems to adapt them to the functionality their own DBMS offers. As a consequence, JDBC programming in Oracle became more time-consuming, since the programmer needs to learn how to use Oracle's adaptation of JDBC to access and use a database. I am not against inheritance and reusability of code, but in this case part of the base classes cannot be used at all to use the 'standard' features of the database. It should be up to the programmer to decide whether he/she wants to use the extra functionality or not, but that is not the case with Oracle. Furthermore, Oracle's extension can only be used with certain versions of the Java Development Kit, since SUN Microsystems – again not wisely – keep changing the base classes from release to release. This is a classic example of abuse of a standard and abuse of object orientation as a software engineering approach.

Portability: Another consequence of Oracle's above mentioned decision is that the code produced for accessing and manipulating the database is not portable. That means that it cannot be used for accessing other database systems, which provide JDBC interface of the same level of functionality. Another problem, which has to do with portability, is that the SQL used by Oracle is not fully compliant with the standard. Again, Oracle has extended the SQL implementation used by its systems to differentiate them from other

systems available in the market that provide similar functionality for reasons that apparently are not related in any way with computing. Unfortunately, this is not an Oracle specific problem.

Documentation: This is one of the biggest disadvantages of Oracle. The documentation is not complete and not properly organised. It is, in general, difficult and therefore time-consuming to find the information that you are looking for. If you are lucky and the information is there, you have to guess how to use it to give a solution to your problem. Often the information you can obtain from the documentation is inaccurate, incomplete, not up to date and/or misleading.

2.2 Middle tier

JavaServer™ Web Development Kit (JSWDK)

The web server that is chosen for the development and deployment for this application is JSWDK. JSWDK is the reference implementation for JSP and the Java™ Servlet API. The release of JSWDK I have used contains a simple servlet engine for developing and testing servlets, the JSP and Servlet APIs, API documentation, a simple HTTP web server and a JSP-enabled engine.

2.2.1 Advantages of using JSWDK as the middle tier component

Cost: This server is downloadable free on the Internet.

Development: Developing servlets with this server is relatively painless since you can use standard output to display information on the servers' window and debugging is easy.

Interface: It is a relatively trivial task to create an application and configure the server to incorporate it.

Size: One of the very good features of the server is its size. The whole server can fit in a couple of floppy disks and that makes life a lot easier.

Compatibility: Since this is a server that comes straight from where the servlet technology was born, it seems to be the best candidate for servlet development.

2.2.2 Disadvantages of using JSWDK as the middle tier component

Reliability: It seems that this server is good for development because of its simplicity but not good enough for real world application deployment. In practice, the server seems not to be able always to cope with the workload. So, there are times where it crashes and the only way to restore the system's functionality is to kill the process and run it again. Also, I noticed that from time to time the server throws an exception even when there is no noticeable problem with the service.

Versatility: This is not a fully-fledged web server and therefore it doesn't provide too much functionality. For example, you cannot have a pool of servlet instances to serve a relatively demanding environment.

2.2.3 Advantages of using Java Servlets as part of the middle tier component

Development: The framework provided by Sun Microsystems for servlet development is very nicely designed and therefore it is easy to understand. The interface is neat, simple and well documented. The servlet libraries can be easily extended to incorporate more functionality when needed.

Portability: Java is platform-independent and therefore portable. It is relatively easy to migrate the middle-tier Java components to another machine or web server.

Efficiency: The servlet engine that is provided by Sun Microsystems is known to be particularly efficient. Furthermore, the fact that servlets are multi-threaded components that run in the server's address space means that there is no context switching between them and the server when they get executed. Also, the servlet, after the first invocation, can be instantiated and remain attached to the server for as long as needed. That means that there is no need to load and unload the component for every request. Another very important feature is its multi-threaded capability. There are many different configurations that can be used as far as the threading model is concerned. The selection of threading model is also related to the functionality provided by the web server on which the servlet is intended to run. Since this particular application is designed for a relatively low volume site, I chose the approach where a single multi-threaded servlet gets instantiated, remains attached to the server and generates a single thread for every request. For every instance of a servlet there is a permanent database connection that is sharable by all its threads. In this way multiple requests can be served by the same database connection, which means that this single resource is being utilised as effectively as possible and there is no need to open and close or maintain at the same time multiple expensive database connections in the system unnecessarily.

Synchronisation: Since multiple threads of the same servlet get concurrent access to the same (sharable) resources, it seems to be necessary for a thread-safety mechanism to be used. Java servlets provide a very straightforward mechanism for synchronisation, which can significantly simplify development with respect to this very delicate aspect of it.

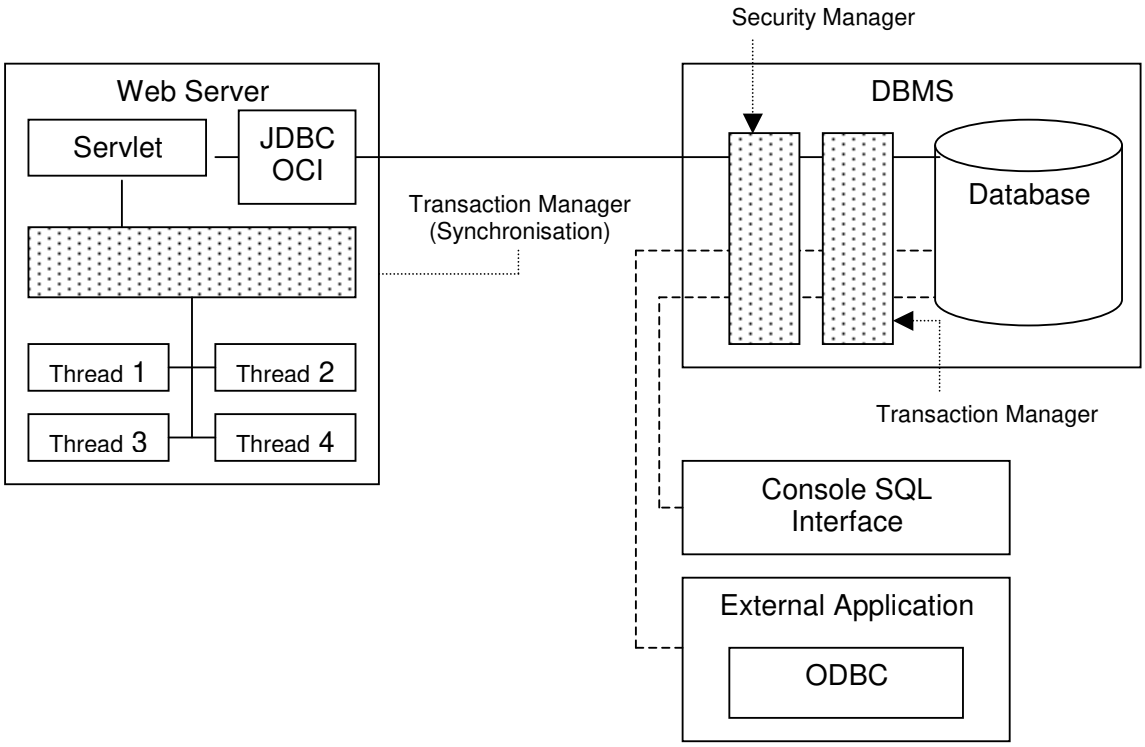


Figure 8 – Concurrent access of the same database connection requires synchronisation

2.2.4 Disadvantages of using Java Servlets as part of the middle tier component

Security: There are many ways to categorise programming languages depending on the classification criterion used. If we consider as a classification criterion the time at which the code gets translated into machine code, we will come up with three main categories of programming languages: Compiled, semi-compiled and scripting. Java is a semi-compiled language, which means that the source code gets translated into an intermediary representation, named bytecode, which in turn needs further processing in order to be scheduled for execution. The consequence of this is that the initial translation process can be reversed and a piece of software known as de-compiler could be used to recover the source code from the bytecode file. That makes Java code particularly vulnerable to malicious users especially when it is used in web-based applications. If the user knows the exact location of the java bytecode file on the Internet and is able to download it, then it is possible to recover the source code and to extract every piece of sensitive information that is hard coded into it. If happen that usernames and passwords are present, then the user can gain unauthorised access to the resources this information is related with. This is obviously a big disadvantage and the only way to reduce the security risk is not to hard code sensitive information in Java code where possible. In this implementation, all the sensitive information resides on the web server and gets retrieved dynamically during the servlet initialisation process.

2.2.5 JDBC – OCI driver

In this system, part of the middle tier component is Oracle's JDBC OCI driver, which is used for interfacing the Java servlets with the back end component. The OCI driver is not written entirely in Java and that means that it is not portable. The advantage in using this driver instead of a pure Java one is efficiency. The disadvantage is that it requires a separate installation on the local machine, but since this is not the front-end tier this is not an issue.

2.3 Front end

For the front end, since this is a web-based system, a combination of HTML, Java Server Pages (JSP's), JavaScript and Cascading Style Sheets have been used.

The HTML part of the system is absolutely essential and there is no alternative technology that could be used instead. XHTML is not an alternative but a constrained refinement of HTML, which hasn't been adopted yet as the new standard. The transition from HTML to XHTML in future releases is not considered a big problem since the HTML part is not very substantial and complicated.

JSP's are the best partners for Servlets from every point of view. One of the best arguments to justify that is that a JSP is a specialisation of a Servlet and therefore it requires the same support, in terms of functionality at the server side. JSP's are used in conjunction with HTML to provide template pages for the interface, which can dynamically include data (query results) sent by servlets at the middle tier.

CSS provides a good mechanism to maintain a consistent look and feel in the system.

JavaScript is the only scripting language, useable by web browsers, that offers true object-oriented programming capabilities. Furthermore, it is the most widely used and versatile language for web development.

2.3.1 Advantages of using JavaScript for development at the client end

Development: JavaScript is the only fully-fledged object oriented scripting language that can be used for client end web development. That means that you can design and implement your own classes, inherit functionality from already existing classes using containment and so on. For this reason, I believe that JavaScript is the only candidate for serious database development at the client end and this is why it was chosen for this system. Also, there is a very helpful tool that is provided by Microsoft that can reduce significantly the development time and difficulty. Microsoft's scripting debugger supports JavaScript and has been proved to be an excellent tool for development.

Versatility: JavaScript provides a very nice and easy interface to the Document Object Model (DOM) used by browsers. That means that it can easily be used to dynamically manipulate contents and appearance in web pages when there is a DOM extension implemented by browsers.

Efficiency: In general, JavaScript interpreters seem to be very efficient. I don't have any information about comparisons between different interpreters of JavaScript or

interpreters of different scripting languages but my general impression is that JavaScript scripts execute fast. The difference in speed between a JavaScript script and a Java applet that does the same job on the browser (which is not scripting but semi-compiled code) is noticeable.

2.3.2 Disadvantages of using JavaScript for development at the client end

Compatibility: Unfortunately, all the software vendors, with no exceptions (as far as I know), implement their own version of JavaScript. Therefore, if you decide to use JavaScript for client web-based development you have two options. The first is to check with if conditions which browser is being used and duplicate your code with mutually exclusive statements that apply to one or the other browser or different versions of the same browser. The second option is to constrain the use of your system to users that use a particular web browser, which you provide an implementation for. Due to lack of time, I chose the second option and I implemented JavaScript that can only be used with Internet Explorer v.5 or above.

3. Conclusions – Suggestions for further development

The system is currently being used on an experimental basis and the results seem to be very promising. The time needed for evaluation and monitoring of students' performance has already been reduced significantly. The only problem encountered so far is that the web server seems not to be very stable. The problem is, in practical terms, serious, since if the middle tier fails then the whole system fails to provide the service. But that has nothing to do with the design and it is only a matter of choosing a suitable server that can be used reliably for deployment.

The development of the system is an ongoing process, which will carry on until we have a stable and complete system that meets the requirements of the course. Our plans for the near future include optimisations and refinement in various parts of the system.

Currently, the middle tier uses the network to communicate with the back end tier because the web server and the DBMS reside on different machines. If a single machine could be used to accommodate both, then there would be no overhead by using the network. A further future step towards this target could be to use a DBMS that provides an HTTP listener and could be used as a web server at the same time. In this case I could eliminate the context switching between them as well.

Another part of the system that needs improvement is the administration of the database at the back end tier. The system, in its current state, does not provide any interface through which the database can be administered. The only way to administer the database or to use the data stored in it is through the native SQL interface of the DBMS itself. The SQL interface is very flexible and versatile, since the user can submit any query, regardless of the complexity of it, but it is not very friendly and it takes time to form statements even for very simple operations. Also, the SQL interface cannot be used with certain types of attributes, such as attributes representing large binary objects. Currently, I am in the process of developing and testing a text-based (console) command line interface to the database. The final version of this interface will provide a GUI and will be available on the Internet as well.

Another limitation of the system is that the interface can only be used with a certain type of browser, which may not always be available. There are various methods to overcome this limitation. One way would be to use a different language that is machine and browser independent for the functional part of the interface. The only alternative that could be used for this part is Java. Unfortunately, there is a noticeable performance penalty when Java is used instead of JavaScript. Performance degrades both in downloading time and in processing time. The other way to do it is to duplicate JavaScript code and try to anticipate all the different behaviours of browsers in the market and add the necessary code to deal with them. But again with this approach there are practical problems that need to be considered. The most obvious one is that the size of the downloadable code and the complexity of it will increase substantially. Furthermore, the developer will have keep updating the code as new browsers or new versions of browsers are released. Clearly, this is not an acceptable solution either. The only thing that can be done, as far as this specific problem is concerned, is to hope that future versions of Java will be more efficient and I will have the opportunity to migrate the functional part to Java.

One of the methods I used to hide the implementation from the end user for security purposes was to present the interface in a window that occupies the entire screen and does not provide any means to access the underlying code. Unfortunately, the only way to load a web page within a frame with only the desired visual features was to load it dynamically using JavaScript through another HTML page. This method is not, in general, desirable because it involves two TCP-IP connections for loading a single page, but unfortunately there was no alternative. There aren't currently any HTML constructs that can be used to specify the properties of the browser and the frame that encloses the web page. So if for example you don't want the toolbar to be present when a specific web page is displayed you have to load it through another web page using JavaScript or another language. I consider that a limitation of HTML and I don't expect this problem to be solved in future releases of the system.

According to the specification for this system, a strict security mechanism involving passwords and secure transmission of data over the network was not a requirement, since the students wouldn’t be able to access personal data. In the future, since this system is fully scaleable and extensible, it is likely to incorporate a component through which the students can monitor their performance throughout the course themselves. In this case, an authentication system that conforms to the Data Protection Act will definitely be a requirement and therefore will be provided with the system.

References

1. Callaway, D. 1999. Inside Servlets. Addison Wesley Longman. ISBN 0-201-37963-5.
2. Hall, M. 2000. Core Servlets and JavaServer Pages. Sun Microsystems Press, Prentice-Hall. ISBN 0-13-089340-4.
3. Nahimovsky, A. Myers, T. 1998. JavaScript Objects. Wrox Press. ISBN 1-861001-89-4.
4. Van Der Lans, R. Introduction to SQL, Addison Wesley Longman. ISBN 0-201-59618-0.
5. Feuerstein, S. Pribyl, B. 1997. Oracle PL/SQL Programming. O' Reilly & Associates. ISBN 1-56592-335-9.
6. Kreines, D. Laskey, B. 1999. Oracle Database Administration. O' Reilly & Associates. ISBN 1-56592-516-5.
7. Topley, K. 1998. Core Java Foundation Classes. Prentice-Hall. ISBN 0-13-080301-4.
8. Van Hoff, A. 1997. Mastering Java 1.1. SYBEX. ISBN 0-7821-2070-9.
9. White, S. Fisher, M. Cattell, R. Hamilton, G. Hapner, M. 1999. JDBC API Tutorial and Reference. Sun Microsystems Press, Addison Wesley Longman. ISBN 0-201-43328-1.

Appendix 1: Implementation – the back end component

SQL DDL statements used for the creation of the database

```
CREATE TABLE STUDENTS
(
        USERNAME        CHAR(7)
                NOT NULL,
        TITLE           VARCHAR(4)
                NOT NULL
                CONSTRAINT      STUDENTS_TITLE_VALUES
                CHECK(TITLE IN('Dr','Mr','Mrs','Ms','Miss')),
        NAME            VARCHAR(30)
                NOT NULL,
        SURNAME         VARCHAR(30)
                NOT NULL,
        MODE_OF_STUDY   CHAR(1)
                NOT NULL
                CONSTRAINT      STUDENTS_MODE_OF_STUDY_VALUES
                CHECK(MODE_OF_STUDY IN('F','P')),
        CONSTRAINT      STUDENTS_PK     PRIMARY KEY(USERNAME)
);

CREATE TABLE ASSIGNMENT_SUBMISSION
(
        USERNAME        CHAR(7)
                NOT NULL,
        SUBMISSION_DATE         DATE
                DEFAULT SYSDATE
                NOT NULL,
        ASSIGNMENT_NO   NUMBER(1)
                NOT NULL
                CONSTRAINT      ASSIGNMENT_SUBMISSION_ASSIGNNO
                CHECK(ASSIGNMENT_NO>0 AND ASSIGNMENT_NO<7),
        ASSIGNMENT_CODE         CLOB
                NOT NULL,
        CONSTRAINT              ASSIGNMENT_SUBMISSION_PK
        PRIMARY KEY(USERNAME,SUBMISSION_DATE),
        CONSTRAINT              ASSIGNMENT_SUBMISSION_FK_USERN
        FOREIGN KEY(USERNAME) REFERENCES STUDENTS(USERNAME)
                ON DELETE CASCADE
);

CREATE OR REPLACE TRIGGER STUDENTS_USERNAME_TRIGGER
        BEFORE INSERT ON STUDENTS
        FOR EACH ROW
        DECLARE
                username CHAR(7):=TRANSLATE(LOWER(:NEW.USERNAME),
                'abcdefghijklmnopqrstuvwxyz0123456789',
                'AAAAAAAAAAAAAAAAAAAAAAAAAANNNNNNNNN');
        BEGIN
                IF username!='AAAAANN'
                THEN
                Raise_application_error(-20201,:NEW.USERNAME||' is not valid username');
                END IF;
        END STUDENTS_USERNAME_TRIGGER;
.
run;
```

SQL DML statement used for updating the database with students' information

```
INSERT INTO STUDENTS(USERNAME,TITLE,NAME,SURNAME,MODE_OF_STUDY)
VALUES('bkark01','Mr','Sokratis','Karkalas','P');
```

Appendix 2: Implementation – the middle tier component

JSWDK configuration

contents of webserver.xml

```
<!DOCTYPE WebServer [

<!ELEMENT WebServer (Service+)>
<!ATTLIST WebServer
    id ID #REQUIRED
    adminPort NMTOKEN "">

<!ELEMENT Service (WebApplication*)>
<!ATTLIST Service
    id ID #REQUIRED
    port NMTOKEN "8043"
    hostName NMTOKEN ""
    inet NMTOKEN ""
    docBase CDATA "webpages"
    workDir CDATA "work"
    workDirIsPersistent (false | true) "false">

<!ELEMENT WebApplication EMPTY>
<!ATTLIST WebApplication
    id ID #REQUIRED
    mapping CDATA #REQUIRED
    docBase CDATA #REQUIRED
    maxInactiveInterval NMTOKEN "30">
]>

<WebServer id="webServer">
    <Service id="service0">
      <WebApplication id="examples" mapping="/examples" docBase="examples"/>
      <WebApplication id="UploadSystem" mapping="/UploadSystem" docBase="UploadSystem"/>
    </Service>
</WebServer>
```

contents of mappings.properties

```
# $Id: mappings.properties,v 1.3.2.1 1999/07/22 23:55:58 mandar Exp $

# Map servlets to paths
# Properties beginning with a . are extension properties, all other
# properties are path properties

#/snoop=snoop
#.snp=snoop
.jsp=jsp
/servletToJSP=jts
/Details.htm=Details
```

contents of servlets.properties

```
# $Id: servlets.properties,v 1.3.2.2 1999/07/27 00:52:13 mandar Exp $

# Define servlets here

# <servletname>.code=<servletclass>
# <servletname>.initparams=<name=value>,<name=value>

#snoop.code=SnoopServlet
#snoop.initparams=initarg1=foo,initarg2=bar

cookie.code=CookieExample
cookie.initparams=foo
jsp.code=com.sun.jsp.runtime.JspServlet
jsp.initparams=keepgenerated=true
jts.code=servletToJsp
jts.initparams=foo
Details.code=StudentDetails
Details.initparams=driver=oracle.jdbc.driver.OracleDriver,
url=jdbc:oracle:oci8:@,
username=ops$bkark01,
password=bkark10
UploadData.code=Upload
UploadData.initparams=driver=oracle.jdbc.driver.OracleDriver,
url=jdbc:oracle:oci8:@,
username=ops$bkark01,
password=bkark10
```

Java code – representation of the database (Database.java)

```
import java.lang.Class;
import java.lang.ClassNotFoundException;
import java.lang.ArrayIndexOutOfBoundsException;
import java.util.Vector;
import java.util.Enumeration;
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

public class Database
{
        //public data members
        public boolean state;
        public static final boolean GOOD=true;
        public static final boolean BAD=false;

        //private data members
        private Connection connection;
        private Vector statements;
        private String driver;
        private String url;
        private String username;
        private String password;

        //constructor
        public Database(String driver,String url,String username,
        String password,Vector statements) throws SQLException
        {
                this.driver=driver;
                this.url=url;
                this.username=username;
                this.password=password;
                this.state=GOOD;              //default
                this.statements=new Vector();

                if(setConnection()==false || setStatements(statements)==false)
                {
                        this.state=BAD;
                        System.out.println("Database cannot be created..");
                        throw new SQLException("Database cannot be created..");
                }
                else System.out.println("Database created successfuly..");
        }

        public void lock() throws SQLException
        {
                connection.setAutoCommit(false);
        }

        public void unlock() throws SQLException
        {
                connection.setAutoCommit(true);
        }

        public void commit() throws SQLException
        {
                connection.commit();
        }

        public PreparedStatement getStatement(int index)
        throws ArrayIndexOutOfBoundsException
        {
                return (PreparedStatement)statements.elementAt(index);
        }
```

```
	//private member methods
	private boolean setConnection()
	{
		connection=null;

		try
		{
			Class.forName(driver);
		}
		catch(ClassNotFoundException e)
		{
			System.out.println("Unable to load driver..");
			System.out.println(e.getMessage());
			return false;
		}

		try
		{
			connection=DriverManager.getConnection(url,username,password);
		}
		catch(SQLException e)
		{
			System.out.println("Unable to open a connection..");
			System.out.println(e.getMessage());
			return false;
		}

		return true;
	}

	private boolean setStatements(Vector statements)
	{
		try
		{
			for(Enumeration e=statements.elements();e.hasMoreElements();)
			{
				String statement=(String)e.nextElement();
				this.statements.addElement
				(connection.prepareStatement(statement));
			}
		}
		catch(SQLException e)
		{
			System.out.println("Unable to create statements..");
			System.out.println(e.getMessage());
			return false;
		}

		return true;
	}
}
```

Java code – representation of the form submitted by the user (HtmlForm.java)

```
import java.io.UnsupportedEncodingException;

class HtmlForm
{
	//private data members
	private String data;
	private String boundary;

	//constructors
	public HtmlForm(byte[] data,String boundary) throws UnsupportedEncodingException
	{
		this.data=new String(data,"ISO8859_1");
		this.boundary=new String(boundary);
	}

	public HtmlForm(String data)
	{
		this.data=new String(data);
		this.boundary=new String(boundary);
	}

	//public member methods
	public String getVariableValue(String variableName)
	{
		//check whether the variable exists in the data
		if(data.indexOf("name=\""+variableName+"\"")==-1)
		{
			System.out.println("Variable "+variableName+" cannot be found.");
			return "";
		}

		String variableData=
		data.substring(data.indexOf("name=\""+variableName+"\""));

		//remove unnecessary lines
		variableData=variableData.substring(variableData.indexOf("\n")+1);
		variableData=variableData.substring(variableData.indexOf("\n")+1);

		//get variable value
		variableData=variableData.substring(0,variableData.indexOf("\n")-1);
		return variableData;
	}


	//public member methods
	public String getFileValue(String variableName)
	{
		String variableData=new String(data);

		//check whether the variable exists in the data
		if(variableData.indexOf("name=\""+variableName+"\"")==-1)
		{
			System.out.println("Variable "+variableName+" cannot be found.");
			return "";
		}

		//exctract variable from the data
		variableData=variableData.substring
		(variableData.indexOf("name=\""+variableName+"\""));

		variableData=variableData.substring(0,variableData.indexOf(boundary));
		variableData=variableData.substring(0,variableData.lastIndexOf("\n")-1);
```

```
                //check whether the variable is of type FILE
                if(variableData.indexOf("filename=\"")==-1)
                {
                        System.out.println("Variable FILE cannot be found.");
                        return "";
                }

                //remove unnecessary lines
                variableData=variableData.substring(variableData.indexOf("\n")+1);
                variableData=variableData.substring(variableData.indexOf("\n")+1);
                variableData=variableData.substring(variableData.indexOf("\n")+1);

                return variableData;
        }
}
```

Java code – servlet that returns, on request, student details as an HTML table (StudentDetails.java)

```
import java.lang.ArrayIndexOutOfBoundsException;
import java.util.Vector;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Driver;
import java.sql.PreparedStatement;
import javax.servlet.ServletException;
import javax.servlet.ServletConfig;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

//Oracle specific JDBC classes
import oracle.jdbc.driver.OracleDriver;

public class StudentDetails extends HttpServlet
{
        //static private data members
        static private Database database;
        static private String driver;
        static private String url;
        static private String username;
        static private String password;
        static private Vector statements;

        //servlet initialisation
        public void init(ServletConfig config) throws ServletException
        {
                System.out.println("================================================");
                System.out.println("BEGIN: Initialisation phase - should be executed only
                once");

                System.out.println("================================================");

                //pass ServletConfig to parent which implements getServletConfig()
                //-----------------------------------------------------------------
                System.out.println("Send initialisation parameter to parent class.");
                super.init(config);

                //get initialisation parameters from server <- for security
                //-----------------------------------------------------------------
                System.out.println("Get initialisation parameters from server.");
                driver=config.getInitParameter("driver");
                url=config.getInitParameter("url");
                username=config.getInitParameter("username");
                password=config.getInitParameter("password");

                if(driver==null||url==null||username==null||password==null)
                {
                        System.out.println("Problem initialising paramenters.");
                        throw new ServletException("Problem initialising paramenters.");
                }
                else System.out.println("Initialisation parameters have been read
                successfully.");

                //register oracle driver
                //-----------------------------------------------------------------
                if(registerDriver(new OracleDriver())==false)
                {
                        System.out.println("Problem registering Oracle driver.");
                        throw new ServletException("Problem registering Oracle driver.");
                }
```

```
        else System.out.println("Database driver has been registered
        successfully.");

        //prepare the sql statements
        //-------------------------------------------------------------------
        statements=new Vector();
        statements.addElement(new String("SELECT * FROM STUDENTS"));

        //instantiate the database
        //-------------------------------------------------------------------
        if(getDBServer()==false)
        {
                System.out.println("Problem instantiating database server.");
                throw new ServletException("Problem instantiating database
                server.");
        }
        else System.out.println("Database has been initialised successfully.");


        System.out.println("================================================");
        System.out.println("END: Initialisation phase");

        System.out.println("================================================");
}

//private member methods
private boolean registerDriver(Driver driver)
{
        try
        {
                DriverManager.registerDriver(driver);
        }
        catch(SQLException e)
        {
                System.out.println("Unable to register driver..");
                System.out.println(e.getMessage());
                return false;
        }

        return true;
}

private boolean getDBServer()
{
        try
        {
                database=new Database(driver,url,username,password,statements);
        }
        catch(SQLException e)
        {
                System.out.println("Connection with the DBMS cannot be
                established..");
                System.out.println(e.getMessage());
        }

        return database.state;
}

private boolean getDataFromDB(PrintWriter out)
{
        System.out.println("================================================");
        System.out.println("BEGIN: Get student data from database.");
        System.out.println("================================================");

        //step 1: check state of database
        //-------------------------------------------------------------------
        if(database.state==Database.BAD)
        {
                System.out.println("Database is in a BAD state..");
                return false;
        }
```

```
else System.out.println("Database is in a good state.");

//step 2: declare local variables needed
//-------------------------------------------------------------------
ResultSet resultSet=null;
PreparedStatement preparedStatement=null;

//step 3: get the prepared statement from the database
//-------------------------------------------------------------------
try
{
        preparedStatement=database.getStatement(0);
}
catch(ArrayIndexOutOfBoundsException e)
{
        System.out.println("Unable to get the statement from the
        database..");
        System.out.println(e.getMessage());
        return false;
}
System.out.println("Statement has been retrieved successfully.");

//lock the shared variables <- thread safety
synchronized(this)
{
        //step 4: execute the query
        //-------------------------------------------------------------------
        try
        {
                resultSet=preparedStatement.executeQuery();
        }
        catch(SQLException e)
        {
                System.out.println("Unable to get the data from the
                database..");
                System.out.println(e.getMessage());
                return false;
        }

        System.out.println("Statement has been executed successfully.");
}

try
{
        int iterator;
        for(iterator=0;resultSet.next();iterator++)
        {
                out.print("<TR>");
                out.print("<TD>"+resultSet.getString("USERNAME")+"</TD>");
                out.print("<TD>"+resultSet.getString("TITLE")+"</TD>");
                out.print("<TD>"+resultSet.getString("NAME")+"</TD>");
                out.print("<TD>"+resultSet.getString("SURNAME")+"</TD>");

                String mode=resultSet.getString("MODE_OF_STUDY");
                mode=mode.equals("F")?"Full-Time":"Part-Time";

                out.print("<TD>"+mode+"</TD>");
                out.println("</TR>");
        }

        System.out.print("No of students retrieved is:");
        System.out.println(iterator);
}
catch(SQLException e)
{
        System.out.println("Unable to read the data from the result
        set..");
        System.out.println(e.getMessage());
        return false;
}
```

```
                System.out.println("=================================================");
                System.out.println("END: Get student data from database.");
                System.out.println("=================================================");
                return true;
        }

        //public member methods
        public void doGet(HttpServletRequest request, HttpServletResponse response) throws
        IOException, ServletException
        {
                response.setContentType("text/plain");
                PrintWriter out=response.getWriter();

                //send the student details to the caller
                //-----------------------------------------------------------------------
                if(getDataFromDB(out)==false)
                {
                        System.out.println("Student details cannot be retrieved.");
                        throw new ServletException("Student details cannot be retrieved.");
                }

                try
                {
                        //close output stream
                        out.close();
                }
                catch(Exception e){}
        }
}
```

Java code – servlet that forwards, on request, assignment details to the database (StudentDetails.java)

```
import java.lang.ArrayIndexOutOfBoundsException;
import java.lang.IllegalStateException;
import java.lang.NumberFormatException;

import java.util.Vector;

import java.io.ByteArrayInputStream;
import java.io.DataInputStream;
import java.io.UnsupportedEncodingException;
import java.io.OutputStream;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.EOFException;

import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Driver;
import java.sql.PreparedStatement;

import java.text.DateFormat;

import javax.servlet.ServletException;
import javax.servlet.ServletConfig;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

//Oracle specific JDBC classes
import oracle.sql.CLOB;
import oracle.sql.DATE;
import oracle.jdbc.driver.OracleDriver;
import oracle.jdbc.driver.OracleResultSet;

public class Upload extends HttpServlet
{
        //static private data members
        static private Database database;
        static private String driver;
        static private String url;
        static private String username;
        static private String password;
        static private Vector statements;

        static private final int MAX_SIZE=102400;

        //servlet initialisation
        public void init(ServletConfig config) throws ServletException
        {
                System.out.println("=================================================");
                System.out.println("BEGIN: Initialisation phase - should be executed only
                once");

                System.out.println("=================================================");

                //pass ServletConfig to parent which implements getServletConfig()
                //-----------------------------------------------------------------
                System.out.println("Send initialisation parameter to parent class.");
                super.init(config);

                //get initialisation parameters from server <- for security
                //-----------------------------------------------------------------
                System.out.println("Get initialisation parameters from server.");
                driver=config.getInitParameter("driver");
                url=config.getInitParameter("url");
```

```
        username=config.getInitParameter("username");
        password=config.getInitParameter("password");

        if(driver==null||url==null||username==null||password==null)
        {
                System.out.println("Problem initialising paramenters.");
                throw new ServletException("Problem initialising paramenters.");
        }
        else System.out.println("Initialisation parameters have been read
        successfully.");

        //register oracle driver
        //---------------------------------------------------------------------
        if(registerDriver(new OracleDriver())==false)
        {
                System.out.println("Problem registering Oracle driver.");
                throw new ServletException("Problem registering Oracle driver.");
        }
        else System.out.println("Database driver has been registered
        successfully.");

        //prepare the sql statements
        //---------------------------------------------------------------------
        statements=new Vector();
        statements.addElement(new String(
        "INSERT INTO ASSIGNMENT_SUBMISSION(USERNAME,ASSIGNMENT_NO,ASSIGNMENT_CODE)
        VALUES(?,?,empty_clob())"));

        statements.addElement(new String("SELECT * "+
                                        "FROM ASSIGNMENT_SUBMISSION "+
                                        "WHERE USERNAME=? "+
                                        "AND ASSIGNMENT_NO=? "+
                                        "AND SUBMISSION_DATE IN "+
                                        "("+
                                                "SELECT MAX(SUBMISSION_DATE) "+
                                                "FROM ASSIGNMENT_SUBMISSION "+
                                                "WHERE USERNAME=? "+
                                                "AND ASSIGNMENT_NO=? "+
                                        ")"
                                        )
                        );

        //instantiate the database
        //---------------------------------------------------------------------
        if(getDBServer()==false)
        {
                System.out.println("Problem instantiating database server.");
                throw new ServletException("Problem instantiating database
                server.");
        }
        else System.out.println("Database has been initialised successfully.");


        System.out.println("================================================");
        System.out.println("END: Initialisation phase");
        System.out.println("================================================");
}

//private member methods
private boolean registerDriver(Driver driver)
{
        try
        {
                DriverManager.registerDriver(driver);
        }
        catch(SQLException e)
        {
                System.out.println("Unable to register driver..");
                System.out.println(e.getMessage());
                return false;
        }
```

```
        return true;
}

private boolean getDBServer()
{
        try
        {
                database=new Database(driver,url,username,password,statements);
        }
        catch(SQLException e)
        {
                System.out.println("Connection with the DBMS cannot be
                established..");
                System.out.println(e.getMessage());
        }

        return database.state;
}

private HtmlForm getClientRequestContents(HttpServletRequest request)
throws IOException, UnsupportedEncodingException, IllegalStateException,
EOFException, Exception
{
        System.out.println("================================================");
        System.out.println("BEGIN: Get client request contents into an HtmlForm
        object.");

        System.out.println("================================================");

        DataInputStream input=new DataInputStream(request.getInputStream());
        String contentType=request.getContentType();

        //make sure content type is multipart/form-data
        if(contentType==null)
        {
                throw new Exception("No content type information available.");
        }

        if(contentType.indexOf("multipart/form-data")==-1)
        {
                throw new Exception("Content type is not multipart/form-data.");
        }
        else System.out.println("Content type is multipart/form-data");

        //get the boundary
        String boundary=contentType.substring(contentType.lastIndexOf("=")+1,
        contentType.length());
        System.out.println("Boundary:"+boundary);

        //get length of content data
        int formDataLength=request.getContentLength();
        System.out.println("Content length:"+formDataLength);

        //check for maximum file size violation
        if(formDataLength>MAX_SIZE)
        {
                throw new Exception("File is too large to upload.");
        }
        else System.out.println("File size is OK.");

        //allocate a byte array to store content data
        byte dataBytes[]=new byte[formDataLength];

        //read file into byte array
        input.readFully(dataBytes);


        System.out.println("================================================");
        System.out.println("END: Get client request contents into an HtmlForm
        object.");
```

```
        System.out.println("==================================================");

        //create and return HtmlForm from byte array for easy manipulation
        return new HtmlForm(dataBytes,boundary);
}

private synchronized boolean sendDataToDB(String USERNAME,String ASSIGNMENT_NO,
String FILE,PrintWriter out)
{
        System.out.println("==================================================");
        System.out.println("BEGIN: Send data to database.");
        System.out.println("==================================================");

        //step 1: declare local variables needed
        //-----------------------------------------------------------------
        ResultSet resultSet=null;
        PreparedStatement preparedStatement=null;

        //step 2: lock the database
        //-----------------------------------------------------------------
        try
        {
                database.lock();
        }
        catch(SQLException e)
        {
                System.out.println("Unable to lock database..");
                System.out.println(e.getMessage());
                return false;
        }
        System.out.println("Database is locked.");

        //step 3: insert a row with an empty clob
        //-----------------------------------------------------------------
        //get the 1st prepared statement from the database
        try
        {
                preparedStatement=database.getStatement(0);
        }
        catch(ArrayIndexOutOfBoundsException e)
        {
                System.out.println("Unable to get the 1st statement from the
                database..");
                System.out.println(e.getMessage());
                return false;
        }
        System.out.println("Statement 0 has been read successfully.");

        //set parameter values
        try
        {
                preparedStatement.setString(1,USERNAME);
                preparedStatement.setInt(2,
                Integer.valueOf(ASSIGNMENT_NO).intValue());
        }
        catch(SQLException e)
        {
                System.out.println("Unable to set the parameter values of the
                statement..");
                System.out.println(e.getMessage());
                return false;
        }
        catch(NumberFormatException e)
        {
                System.out.println("Unable to convert the value of ASSIGNMENT_NO to
                an int.");
                System.out.println(e.getMessage());
                return false;
        }
        System.out.println("Parameters have been set successfully.");
```

```
//execute the query
try
{
        resultSet=preparedStatement.executeQuery();
        System.out.println("Query has been executed successfully.");
}
catch(SQLException e)
{
        System.out.println("Unable to insert row..");
        System.out.println(e.getMessage());
        return false;
}

//clear the parameters
finally
{
        try
        {
                preparedStatement.clearParameters();
        }
        catch(SQLException e)
        {
                System.out.println("Unable to clear the parameters of the
                statement..");
                System.out.println(e.getMessage());
        }
        System.out.println("Statement parameters have been cleared
        successfully.");
}

//step 4: get the clob locator from the table
//-----------------------------------------------------------------
//get the 2nd prepared statement from the database
try
{
        preparedStatement=database.getStatement(1);
}
catch(ArrayIndexOutOfBoundsException e)
{
        System.out.println("Unable to get the 2nd statement from the
        database..");
        System.out.println(e.getMessage());
        return false;
}
System.out.println("Statement 1 has been read successfully.");

//set parameter values
try
{
        preparedStatement.setString(1,USERNAME);
        preparedStatement.setInt(2,
        Integer.valueOf(ASSIGNMENT_NO).intValue());
        preparedStatement.setString(3,USERNAME);
        preparedStatement.setInt(4,
        Integer.valueOf(ASSIGNMENT_NO).intValue());
}
catch(SQLException e)
{
        System.out.println("Unable to set the parameter values of the
        statement..");
        System.out.println(e.getMessage());
        return false;
}
catch(NumberFormatException e)
{
        System.out.println("Unable to convert the value of ASSIGNMENT_NO to
        an int.");
        System.out.println(e.getMessage());
        return false;
}
```

```
System.out.println("Parameters have been set successfully.");

//execute the query
try
{
        resultSet=preparedStatement.executeQuery();
        System.out.println("Query has been executed successfully.");
}
catch(SQLException e)
{
        System.out.println("Unable to get the row containing the clob..");
        System.out.println(e.getMessage());
        return false;
}

//clear the parameters
finally
{
        try
        {
                preparedStatement.clearParameters();
        }
        catch(SQLException e)
        {
                System.out.println("Unable to clear the parameters of the
                statement..");
                System.out.println(e.getMessage());
        }
        System.out.println("Statement parameters have been cleared
        successfully.");
}

CLOB clob=null;
DATE date=null;

try
{
        if(resultSet.next())
        {
                clob=((OracleResultSet)resultSet).
                getCLOB("ASSIGNMENT_CODE");
                date=(DATE)(((OracleResultSet)resultSet).
                getDATE("SUBMISSION_DATE"));
        }
}
catch(SQLException e)
{
        System.out.println("Unable to get the clob locator or date..");
        System.out.println(e.getMessage());
        return false;
}
System.out.println("Clob locator and date have been retrieved
successfully.");

//step 5: open an output stream to the CLOB
//-------------------------------------------------------------------
OutputStream outStream=null;

try
{
        outStream=clob.getAsciiOutputStream();
}
catch(SQLException e)
{
        System.out.println("Unable to open output stream to CLOB..");
        System.out.println(e.getMessage());
        return false;
}
System.out.println("Output stream to the clob has been read
successfully.");
```

```
//step 6: create a buffer
//-----------------------------------------------------------------------
int size;
try
{
        size=clob.getBufferSize();
}
catch(SQLException e)
{
        System.out.println("Unable to get the size of the CLOB..");
        System.out.println(e.getMessage());
        return false;
}
System.out.println("CLOB's buffer size: "+size);

byte[] buffer=new byte[size];

//step 7: convert file data from String to InputStream
//-----------------------------------------------------------------------
byte[] data=null;
ByteArrayInputStream inStream=null;

try
{
        data=FILE.getBytes("ISO8859_1");
        inStream=new ByteArrayInputStream(data);
}
catch(UnsupportedEncodingException e)
{
        System.out.println("Unable to convert file data with required
        encoding..");
        System.out.println(e.getMessage());
        return false;
}
System.out.println("File data have been converted to ByteArray
successfully.");

//step 8: transfer the data from the file to the CLOB using the buffer
//-----------------------------------------------------------------------
int length=-1;
try
{
        while((length=inStream.read(buffer))!=-1)
                outStream.write(buffer,0,length);
        System.out.println("Data have been transfered successfully to
        CLOB.");
}
catch(IOException e)
{
        System.out.println("Unable to transfer data to CLOB..");
        System.out.println(e.getMessage());
        return false;
}
finally
{
        //step 9: close the streams
        //-------------------------------------------------------------------
        try
        {
                inStream.close();
                outStream.close();
                System.out.println("Streams have been closed
                successfully.");
        }
        catch(IOException e)
        {
                System.out.println("Unable to close the streams..");
                System.out.println(e.getMessage());
                return false;
        }
}
```

```
        //step 10: commit the changes
        //-------------------------------------------------------------------
        try
        {
                database.commit();
        }
        catch(SQLException e)
        {
                System.out.println("Unable to commit changes..");
                System.out.println(e.getMessage());
                return false;
        }
        System.out.println("Changes have been committed successfully.");

        //step 11: unlock the database
        //-------------------------------------------------------------------
        try
        {
                database.unlock();
        }
        catch(SQLException e)
        {
                System.out.println("Unable to unlock database..");
                System.out.println(e.getMessage());
                return false;
        }
        System.out.println("Database has been unlocked successfully.");

        //send information to the user
        out.println("<H2>");
        out.println("<P ALIGN=CENTER>Assignment No "+ASSIGNMENT_NO+
        " for "+USERNAME+" has been saved </P>");
        out.println("<P ALIGN=CENTER> Date: ");
        out.println(DateFormat.getDateInstance().format(date.dateValue()));
        out.println(" ");
        out.println(date.timeValue());
        out.println("</P>");
        out.println("</H2>");

        System.out.println("HTML - submission info - has been sent
        successfully.");


        System.out.println("================================================");
        System.out.println("END: Send data to database.");

        System.out.println("================================================");

        return true;
}

//public member methods
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException
{
        System.out.println("\n\n   ***   BEGIN SESSION   ***\n");

        //set up responce
        //-----------------------------------------------------------------------
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();

        //send the 1st part of the page
        out.println("<HTML>");
        out.println("<HEAD>");
        out.println("<TITLE></TITLE>");
        out.println("</HEAD>");
        out.println("<BODY BGCOLOR=\"#FFFFD2\">");
        out.println("<P ALIGN=CENTER>");
```

```
System.out.println("HTML - header - has been sent successfully.");

//get content of request
//-------------------------------------------------------------------------
HtmlForm form=null;
boolean errorFlag=false;

try
{
        form=getClientRequestContents(request);
}
catch(UnsupportedEncodingException e)
{
        System.out.println("Unable to create HtmlForm object with required
        encoding..");
        System.out.println(e.getMessage());
        out.println("Internal Error: "+e.getMessage());
        errorFlag=true;
}
catch(EOFException e)
{
        System.out.println(e.getMessage());
        out.println("Internal Error: "+e.getMessage());
        errorFlag=true;
}
catch(IOException e)
{
        System.out.println("Unable to transfer form data to byte array..");
        System.out.println(e.getMessage());
        out.println("Internal Error: "+e.getMessage());
        errorFlag=true;
}
catch(IllegalStateException e)
{
        System.out.println(e.getMessage());
        out.println("Internal Error: "+e.getMessage());
        errorFlag=true;
}
catch(Exception e)
{
        System.out.println(e.getMessage());
        out.println("Internal Error: "+e.getMessage());
        errorFlag=true;
}
System.out.println("Client request has been retrieved successfully.");

//get ASSIGNMENT_NO variable from request
//-------------------------------------------------------------------------
String ASSIGNMENT_NO=form.getVariableValue("ASSIGNMENT_NO");
System.out.println("ASSIGNMENT_NO:"+ASSIGNMENT_NO);

if(ASSIGNMENT_NO.equals(""))
{
        out.println("Internal Error: ASSIGNMENT_NO cannot be found");
        errorFlag=true;
}

//get USERNAME variable from request
//-------------------------------------------------------------------------
String USERNAME=form.getVariableValue("USERNAME");
System.out.println("USERNAME:"+USERNAME);

if(USERNAME.equals(""))
{
        out.println("Internal Error: USERNAME cannot be found");
        errorFlag=true;
}

//get FILE variable from request
//-------------------------------------------------------------------------
```

```
                String FILEDATA=form.getFileValue("FILEDATA");
                System.out.print("FILEDATA length:");
                System.out.println(FILEDATA.length());

                if(FILEDATA.equals(""))
                {
                        out.println("Internal Error: FILEDATA cannot be found");
                        errorFlag=true;
                }

                //send assignment to the DB
                //-------------------------------------------------------------------------
                if(errorFlag==false)
                {
                        if(sendDataToDB(USERNAME,ASSIGNMENT_NO,FILEDATA,out)==false)
                        {
                                System.out.println("The system cannot store the assignment
                                in the database.");
                                out.println("Internal Error: The system cannot store the
                                assignment in the database.");
                                errorFlag=true;
                        }
                        else System.out.println("Data have been sent successfully to the
                        database.");
                }

                out.println("</P>");

                if(errorFlag==false)
                {
                        out.println("<FORM>");
                        out.println("<P ALIGN=CENTER>");
                        out.println("<INPUT TYPE=BUTTON VALUE=\"Print Confirmation Page\"
                        onClick=\"window.print()\"></P>");
                        out.println("</FORM>");
                }

                out.println("</BODY>");
                out.println("</HTML>");

                try
                {
                        //close output stream
                        out.close();
                        System.out.println("Output stream has been closed successfully.");
                }
                catch(Exception e){}

                System.out.println("\n   ***   END SESSION   ***\n\n");
        }
}
```

Appendix 3: Implementation – the front end component

HTML code used for loading the interface (home.htm)

```
<HTML>

<HEAD>

<TITLE>ASSIGNMENT SUBMISSION</TITLE>

<SCRIPT SRC="./scripts/browser.js"></SCRIPT>

</HEAD>

<BODY CLASS=body onLoad="checkBrowser()">

</BODY>

</HTML>
```

HTML code used for the one and only front web page (assignments.jsp)

```
<HTML>
<HEAD>
<TITLE>ASSIGNMENT SUBMISSION</TITLE>

<SCRIPT SRC="./scripts/database.js"></SCRIPT>
<SCRIPT SRC="./scripts/utils.js"></SCRIPT>

<LINK REL=StyleSheet HREF="./styles/style.css" TYPE="text/css" MEDIA=all>

</HEAD>

<BODY CLASS=body
onLoad="window.document.database=new Database(window.document.all.dataTable,
                                              window.document.forms.studentInfo)"
onMouseDown="disable_left(event)">

<H1 ALIGN=CENTER>Assignment Submission</h1>

<H5 ALIGN=JUSTIFY>Please provide the information required in the field(s) below and click
the 'Submit' button to proceed. Note that the information you supply will be checked for
validity. Therefore, if you are careful and you check the information yourself before you
submit it, the process will be less time-consuming.</H5>

<H2 ID=infoLine ALIGN=CENTER></h2>

<FORM ID=studentSearch CLASS=visible>

<TABLE ALIGN=CENTER BORDER=0 CELLPADDING=7 CELLSPACING=4>
        <TR>
        <TD CLASS=description>USERNAME:</TD>
        <TD CLASS=data><INPUT TYPE=TEXT NAME=USERNAME SIZE=7 MAXLENGTH=7></TD>
        </TR>

        <TR>
        <TD CLASS=buttons><INPUT TYPE=RESET VALUE="Reset" CLASS=controls></TD>
        <TD CLASS=buttons><INPUT TYPE=BUTTON VALUE="Submit" CLASS=controls
                onClick="find(this.form,window.document.forms.uploadForm)"></TD>
        </TR>
</TABLE>

</FORM>


<FORM ID=studentInfo CLASS=invisible>

<TABLE ALIGN=CENTER BORDER=0 CELLPADDING=7 CELLSPACING=4>
        <TR>
        <TD CLASS=description>TITLE:</TD>
        <TD CLASS=data><INPUT TYPE=TEXT READONLY NAME=TITLE SIZE=4 MAXLENGTH=4></TD>
        </TR>

        <TR>
        <TD CLASS=description>FIRST NAME:</TD>
        <TD CLASS=data><INPUT TYPE=TEXT READONLY NAME=FIRSTNAME SIZE=30 MAXLENGTH=30></TD>
        </TR>

        <TR>
        <TD CLASS=description>SURNAME:</TD>
        <TD CLASS=data><INPUT TYPE=TEXT READONLY NAME=SURNAME SIZE=30 MAXLENGTH=30></TD>
        </TR>

        <TR>
        <TD CLASS=description>MODE OF STUDY:</TD>
        <TD CLASS=data><INPUT TYPE=TEXT READONLY NAME=MODE SIZE=9 MAXLENGTH=9></TD>
        </TR>
</TABLE>

</FORM>
```

```
<FORM ID=uploadForm CLASS=invisible METHOD="POST" ENCTYPE="multipart/form-data"
ACTION="./servlet/UploadData" TARGET="CONFIRMATION" onSubmit="return validate(this)">

<TABLE ALIGN=CENTER BORDER=0 CELLPADDING=7 CELLSPACING=4>
        <TR>
        <TD COLSPAN=2 CLASS=data>
        <INPUT ID=fileToUpload TYPE="FILE" NAME="FILEDATA" VALUE="" MAXLENGTH=255>
        </TD>
        </TR>

        <TR>
        <TD CLASS=description>ASSIGNMENT NO:</TD>
        <TD CLASS=data ALIGN=CENTER VALIGN=MIDDLE WIDTH=30%>
                <SELECT NAME=ASSIGNMENT_NO SIZE=1 CLASS=controls>
                        <OPTION VALUE="1" SELECTED>1</OPTION>
                        <OPTION VALUE="2">2</OPTION>
                        <OPTION VALUE="3">3</OPTION>
                        <OPTION VALUE="4">4</OPTION>
                        <OPTION VALUE="5">5</OPTION>
                        <OPTION VALUE="6">6</OPTION>
                </SELECT>
        </TD>
        </TR>

        <TR>
        <TD COLSPAN=2 CLASS=buttons>
        <INPUT TYPE=SUBMIT VALUE="Submit" CLASS=controls></INPUT>
        </TD>
        </TR>
</TABLE>

        <INPUT TYPE=HIDDEN NAME=USERNAME>

</FORM>

<P ALIGN="CENTER">
<IMG ID=hourGlasses CLASS=invisible SRC="./images/hourglass.gif" BORDER=1></IMG>
</P>

<BR>

<IFRAME ID=confirmationPage CLASS=invisible NAME="CONFIRMATION" FRAMEBORDER=0
ALIGN="CENTER" HEIGHT=30% WIDTH=90% SRC="blank.htm">
</IFRAME>

<BR>
<FORM ID=endSession CLASS=visible>
<TABLE CLASS=buttons ALIGN=CENTER BORDER=0 WIDTH=30% CELLPADDING=10>
        <TR>
        <TD ID=backButton CLASS=invisible WIDTH=50%>
        <INPUT TYPE=BUTTON VALUE="Restart" CLASS=controls onClick="go_back()"></INPUT>
        </TD>
        <TD>
        <INPUT TYPE=BUTTON VALUE="Close Window" CLASS=controls onClick="window.close()">
        </INPUT>
        </TD>
        </TR>
</TABLE>
</FORM>
<TABLE ID=dataTable CLASS=invisible>
        <TR>
        <TH>Username</TH>      <TH>TITLE</TH> <TH>FIRSTNAME</TH>     <TH>SURNAME</TH>
        <TH>MODE</TH>
        </TR>

<jsp:include page="Details.htm" flush="true" />

</TABLE>
</BODY>
</HTML>
```

JavaScript code used for browser checking and loading the front page (browser.js)

```
function checkBrowser()
{
        var browserInfo=navigator.appVersion;
        var versionPosition;

        if((versionPosition=browserInfo.indexOf("MSIE"))==-1)
        {
                alert("This browser is not Internet Explorer!
                Read the instructions carefuly and try again.");
                redirect("instructions.htm");
                return;
        }

        versionPosition+="MSIE ".length;

        if(browserInfo.charAt(versionPosition)<5)
        {
                alert("You need at least version 5 or above of Internet Explorer
                to run this application!");
                redirect("instructions.htm");
                return;
        }

        configureBrowserWin();
}

function configureBrowserWin()
{
        window.open("http://medusa.dcs.bbk.ac.uk:8043/UploadSystem/assignments.jsp",
        "","fullscreen=yes,scrollbars=yes");
}

function redirect(url)
{
        window.location.replace(url);
}
```

JavaScript code used for input validation and dynamic screen alteration (utils.js)

```
function Document_reset_info(info)
{
        var infoLine=this.all.infoLine;

        if(info.length==0)
        {
                infoLine.innerHTML="";
                infoLine.className="invisible";
        }
        else
        {
                infoLine.innerHTML=info;
                infoLine.className="visible";
        }
}

window.document.reset_info=Document_reset_info;

//*********************************************************//

function screen1()
{
        window.document.reset_info("");
        window.document.all.confirmationPage.src="";
        window.document.all.studentSearch.className="visible";
        window.document.all.studentInfo.className="invisible";
        window.document.all.uploadForm.className="invisible";
        window.document.all.hourGlasses.className="invisible";
        window.document.all.confirmationPage.className="invisible";
        window.document.all.endSession.className="visible";
        window.document.all.backButton.className="invisible";
}

function screen2()
{
        window.document.all.infoLine.className="visible";
        window.document.all.studentSearch.className="invisible";
        window.document.all.studentInfo.className="visible";
        window.document.all.uploadForm.className="visible";
        window.document.all.hourGlasses.className="invisible";
        window.document.all.confirmationPage.className="invisible";
        window.document.all.endSession.className="visible";
        window.document.all.backButton.className="visible";
}

function screen3()
{
        window.document.all.infoLine.className="visible";
        window.document.all.studentSearch.className="invisible";
        window.document.all.studentInfo.className="invisible";
        window.document.all.uploadForm.className="invisible";
        window.document.all.hourGlasses.className="visible";
        window.document.all.confirmationPage.className="invisible";
        window.document.all.endSession.className="invisible";
        window.document.all.backButton.className="invisible";
}

function screen4()
{
        window.document.reset_info("");
        window.document.all.studentSearch.className="invisible";
        window.document.all.studentInfo.className="invisible";
        window.document.all.uploadForm.className="invisible";
        window.document.all.hourGlasses.className="invisible";
        window.document.all.confirmationPage.className="visible";
```

```
        window.document.all.endSession.className="visible";
        window.document.all.backButton.className="visible";
}

function checkLoad()
{
        var loadingPage=window.document.all('confirmationPage');

        if(loadingPage.ReadyState==4)
        {
                screen4();
                return;
        }

        setTimeout("checkLoad()",1000);
}

function disable_left(event)
{
        if(event.button==1)
                return;

        alert("You are only allowed to use the left mouse button in this page!");
}


function validate(form)
{
        //find out whether there is a valid username ready to be submitted with the file
        if(form.USERNAME.value=="")
        {
                alert("Internal error. There is no valid username!");
                return false;
        }

        //find out whether the user has selected a file
        var path=new String(form.fileToUpload.value);

        if(path.length==0)
        {
                alert("You have not selected a file to upload!");
                return false;
        }

        //find out whether the selected file has an extension
        var dot;

        if((dot=path.lastIndexOf(".",path.length-1))==-1)
        {
                alert("The file you specified has to have an extension!");
                return false;
        }

        //find out whether the selected file has the correct extension
        var extension=path.substring(dot+1);

        if(!(extension=="cpp"||extension=="C"))
        {
                alert(".cpp and .C are the only permissible extensions for the type of
                file you have to upload!");
                return false;
        }

        window.document.reset_info("Please wait...");
        screen3();
        setTimeout("checkLoad()",3000);

        return true;
}
```

```
function find(inputForm,uploadForm)
{
        //get element to be processed
        var criterion=inputForm.USERNAME;

        //change the case
        criterion.value=criterion.value.toLowerCase();

        //check for validity

        //if the size is zero than expected return
        if(criterion.value.length==0)
        {
                alert("You have to type a "+criterion.name.toString().toLowerCase()+
                " first!");
                criterion.select();
                return;
        }

        //if the size is smaller than expected return
        if(criterion.value.length!=criterion.size)
        {
                alert("The "+criterion.name.toString().toLowerCase()+
                " you submitted is shorter than expected!");
                criterion.select();
                return;
        }

        //if the 1st 5 elements are not letters return
        for(var iterator=0;iterator<5;iterator++)
        {
                if(!checkChar(criterion.value.charAt(iterator)))
                {
                        alert("The "+criterion.name.toString().toLowerCase()+
                        " you submitted does not have the correct format!");
                        criterion.select();
                        return;
                }
        }

        //if the last 2 elements are not digits return
        for(;iterator<criterion.value.length;iterator++)
        {
                if(!checkNum(criterion.value.charAt(iterator)))
                {
                        alert("The "+criterion.name.toString().toLowerCase()+
                        " you submitted does not have the correct format!");
                        criterion.select();
                        return;
                }
        }

        if(window.document.database.binary_search(criterion.value))
        {
                uploadForm.USERNAME.value=criterion.value;
                screen2();
        }
        else criterion.select();
}

function checkChar(char)
{
        return char>='a'&&char<='z';
}

function checkNum(num)
{
        return num>='0'&&num<='9';
}
```

```
function go_back()
{
        screen1();
}
```

JavaScript code used for database implementation (database.js)

```
//constructor
function Database(htmlTable,displayForm)
{
	//store the arguments as member variables
	this.htmlTable=htmlTable;
	this.displayForm=displayForm;

	//create an empty dataTable and add it to the databases member variable
	this.dataTable=new Object();

	//form the headers object and add it to the dataTable
	var tableLength=htmlTable.rows[0].cells.length;
	var headers=new Array(tableLength);

	for(var counter=0;counter<tableLength;counter++)
		headers[counter]=htmlTable.rows[0].cells[counter].innerHTML.trim_spaces();

	this.dataTable.headers=headers;
	//form the data object and add it to the dataTable
	var rows=htmlTable.rows.length-1;
	var data=new Array(rows);

	for(var iterator_1=1;iterator_1<=rows;iterator_1++)
	{
		var htmlTableRow=htmlTable.rows[iterator_1];
		var dataTableRow=new Object();

		for(var iterator_2=0;iterator_2<tableLength;iterator_2++)
		dataTableRow[headers[iterator_2]]=
		htmlTableRow.cells[iterator_2].innerHTML.trim_spaces();

		data[iterator_1-1]=dataTableRow;
	}

	data.sorted=false;
	this.dataTable.data=data;
}

//puts the database entries into alphabetical order on the first field
function Database_sort()
{
	//get a reference to the data
	var data=this.dataTable.data;

	//get the headers of the table
	var headers=this.dataTable.headers;
	var criterion=headers[0];

	//get the size of the data (count -1 because of the headers)
	var rows=data.length;

	//iterate through the entries
	for(var iterator_1=0;iterator_1<rows;iterator_1++)
	{
		//find the lowest username
		var index=iterator_1;
		for(var iterator_2=iterator_1+1;iterator_2<rows;iterator_2++)
		{
			if(data[iterator_2][criterion]<data[index][criterion])
				index=iterator_2;
		}

		//swap the values
		var temporary=data[iterator_1];
		data[iterator_1]=data[index];
		data[index]=temporary;
	}
```

```
        data.sorted=true;
}

//performs a linear search on username and returns a Record object
function Database_linear_search(username)
{
        //get a reference to the data
        var data=this.dataTable.data;

        //get the headers of the table
        var headers=this.dataTable.headers;

        for(var iterator=0;iterator<data.length;iterator++)
        {
                if(data[iterator][headers[0]]==username)
                {
                        this.update_view(data[iterator]);
                        return true;
                }
        }

        //if not found
        alert("Unfortunately, there is no entry in the database for "+username+
        ". Please try again.");
        return false;
}

//sorts the records if not in order, performs a binary search on username,
//and updates the displayForm if necessary
function Database_binary_search(username)
{
        //get a reference to the data
        var data=this.dataTable.data;

        //get the headers of the table
        var headers=this.dataTable.headers;

        if(data.sorted==false)
                this.sort();
        var first=0;
        var last=data.length-1;

        while(first<=last)
        {
                var middle=Math.floor((first+last)/2);

                if(data[middle][headers[0]]==username)
                {
                        this.update_view(data[middle]);
                        return true;
                }

                if(data[middle][headers[0]]<username)
                        first=middle+1;
                else last=middle-1;
        }

        //if not found
        alert("Unfortunately, there is no entry in the database for "+username+
        ". Please try again.");
        return false;
}

//returns the number of records in the database
function Database_get_No_of_entries()
{
        return this.dataTable.data.length;
}
```

```
//iterated through the database and displays the records
function Database_display()
{
	var entries=this.get_No_of_entries();
	var data=this.dataTable.data;
	var headers=this.dataTable.headers;

	var table=new String();

	for(var iterator_1=0;iterator_1<entries;iterator_1++)
	{
		var dataTableRow=data[iterator_1];
		var row=new String();

		for(var iterator_2=0;iterator_2<headers.length;iterator_2++)
			row+=dataTableRow[headers[iterator_2]]+" ";

		table+=row.trim_spaces()+"\n";
	}

	alert(table);
}

function Database_update_view(record)
{
	var displayForm=this.displayForm;
	var headers=this.dataTable.headers;

	for(var iterator=1;iterator<headers.length;iterator++)
		displayForm[headers[iterator]].value=record[headers[iterator]];

	var txt=new String();
	txt+="Hello "+record[headers[1]]+" "+record[headers[2]]+" "+record[headers[3]];
	window.document.reset_info(txt);
}

Database.prototype.sort=Database_sort;
Database.prototype.linear_search=Database_linear_search;
Database.prototype.binary_search=Database_binary_search;
Database.prototype.get_No_of_entries=Database_get_No_of_entries;
Database.prototype.display=Database_display;
Database.prototype.update_view=Database_update_view;

//************************************************************//

function String_trim_spaces()
{
	var result=new String(this);
	while(result.length!=0)
	{
		var left=0;
		var right=result.length-1;

		if(result.charAt(left)==' ')
		{
			result=result.substring(left+1);
			continue;
		}

		if(result.charAt(right)==' ')
		{
			result=result.substring(0,right);
			continue;
		}
		break;
	}
	return result;
}

String.prototype.trim_spaces=String_trim_spaces;
```

CSS code used for the appearance of the interface (style.css)

```
.invisible
{
	display		:none
}

.visible
{
	display		:block
}

.body
{
	background-color	:#FFFFD2;

	letter-spacing		:normal;
	text-decoration		:none;
	text-align		:center;
	line-height		:normal
}

.description
{
	background-color	:#b2b2b2;
	width			:auto
}

.data
{
	background-color	:#FF9933
}

.buttons
{
	background-color	:#777777
}

.controls
{
	width			:100%
}
```

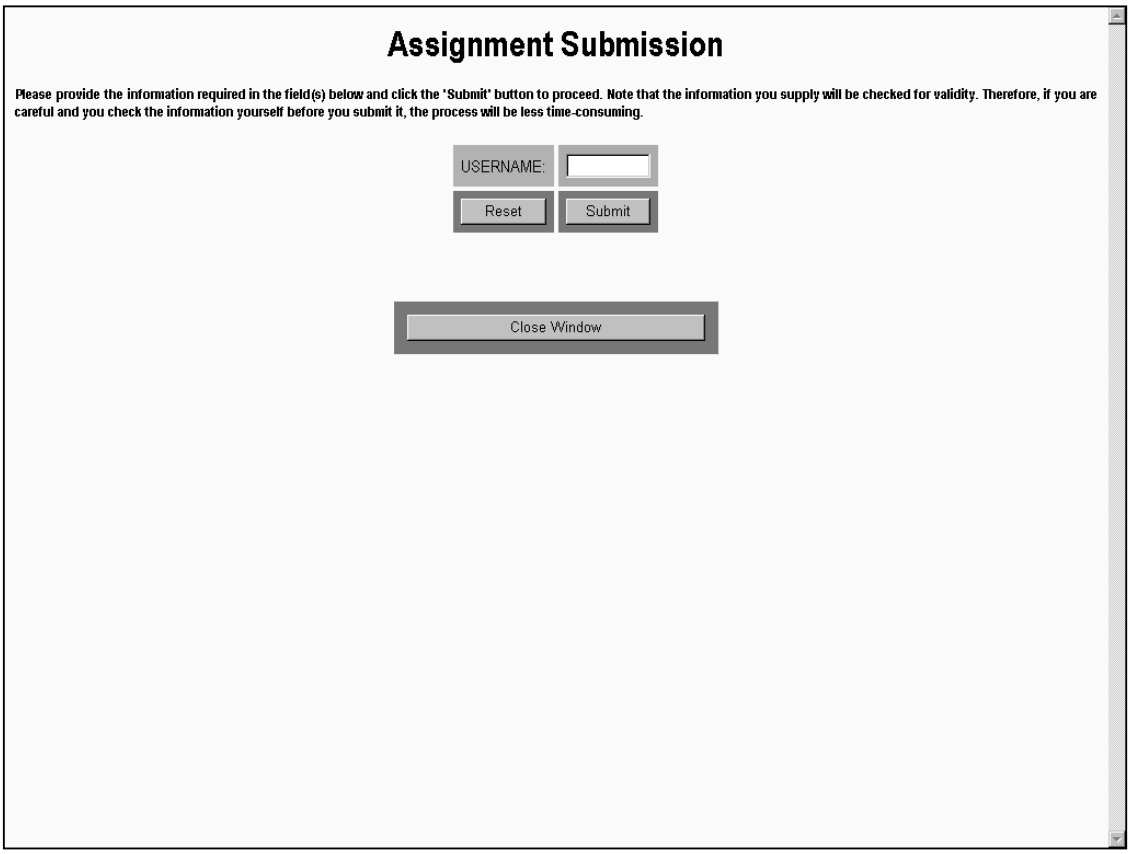
Appendix 4: Usage instructions given to students

General instructions

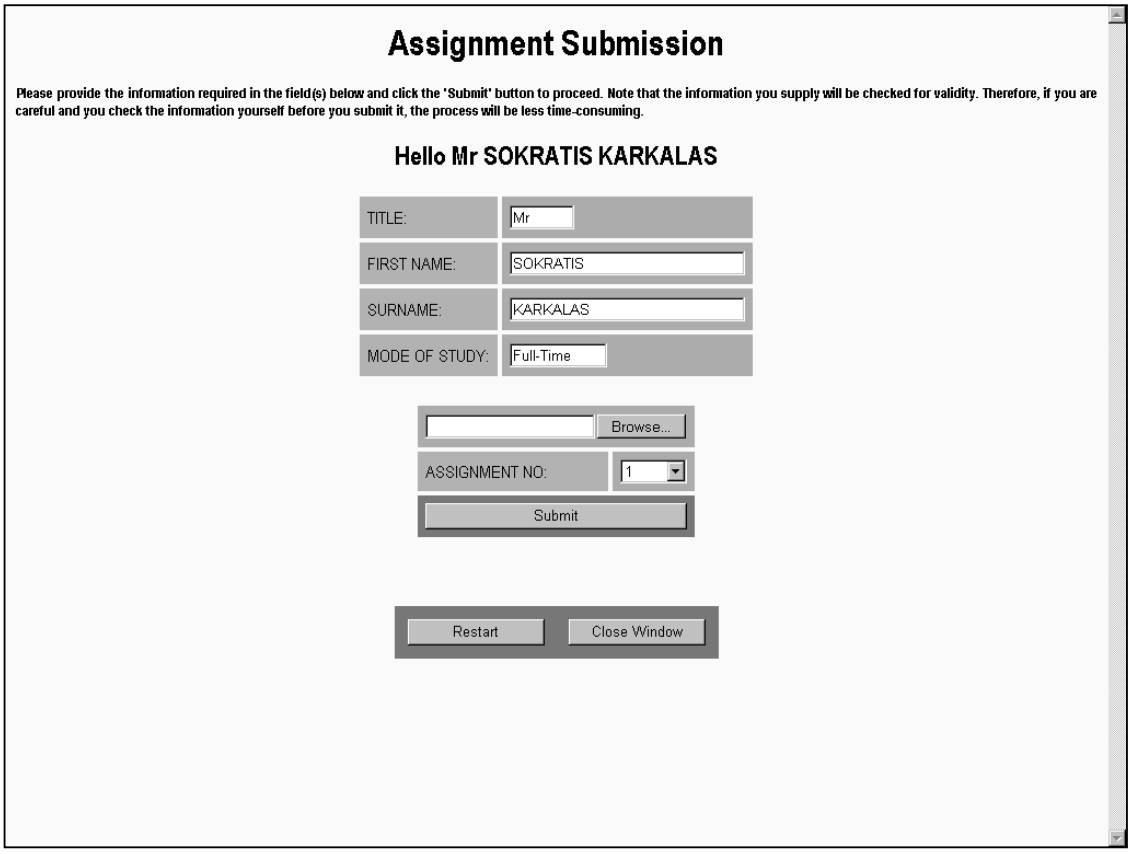
First of all make sure that Microsoft Internet Explorer v.5 or later is properly installed on your machine. It won't work if you use Netscape or any other browsers for accessing this service. Also, note that your browser needs to be "JavaScript enabled" (which is the default setting).

You can access this service directly typing http://www.dcs.bbk.ac.uk/assignment.html. Alternatively, you can use the links on the departmental web page. If you choose the second option, the procedure is the following: Browse the departmental web site typing http://www.dcs.bbk.ac.uk. On the left side of the page you will see a column with links. Click on "Support" to browse the Support page and then click on "MSc Computing Science" and "C++ Assignment Submission".

After you have done that you will be presented the following screen:

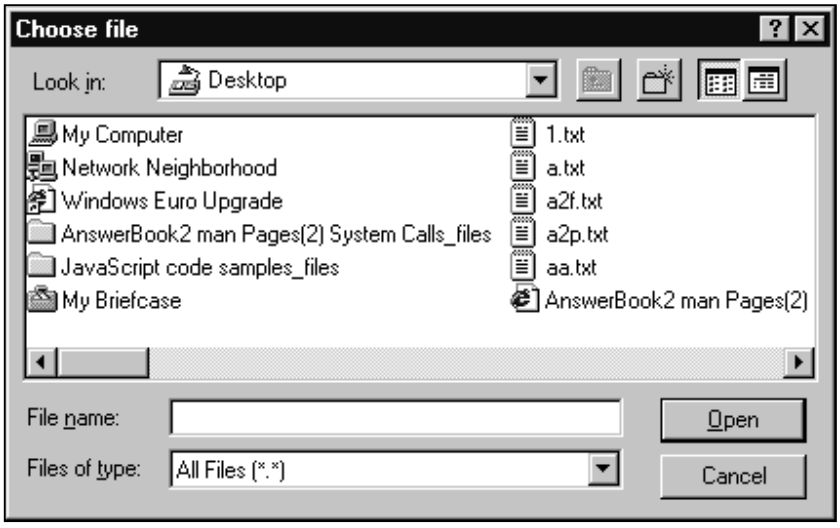


Key in your username in the corresponding field and click the "Submit" button with your mouse. If the information is correct you will be presented a window like the following.

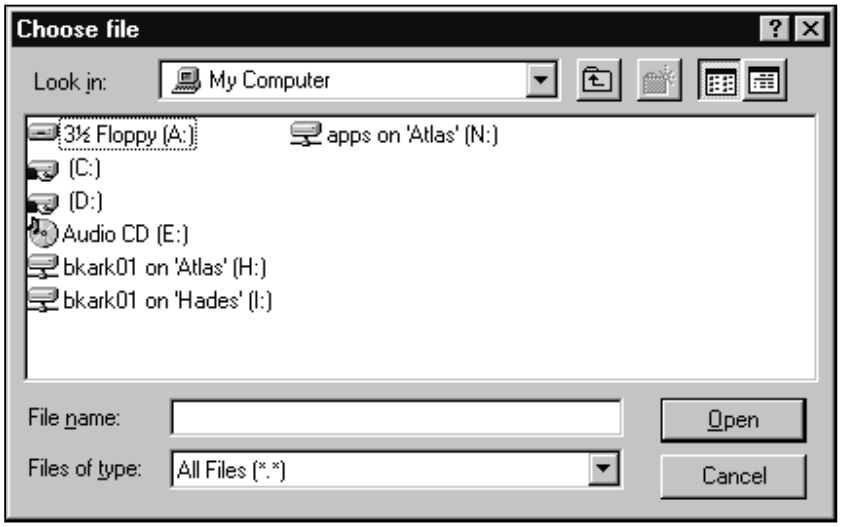


If it is not your name that appears on this screen, you should click the "Restart" button (located at the bottom of the page) with your mouse to go back to the first screen and key in your username (carefully) again.

If you are happy with the submission information you have passed so far, browse the file that contains the source code for your assignment on your system using the "Browse…" button. When you click the "Browse…" button the window that will appear on the screen will look like the following screenshot.



If you use Microsoft Windows a lot and you are familiar with the environment it won't be difficult to find the specific location of your file. Assuming that you use the system from within the Department and you have stored your file on your home directory (which is H:), you will have to double-click on My Computer to get the available drives on your computer (shown on the next screenshot).



Then double click on "[your username] on 'Atlas' (H:)" to get access to your files, select the file that contains the source code for your assignment and press the button Open. That will automatically place the filename (including the whole path) of your file in the field. You then have to confirm which assignment you are submitting (use the drop-down menu).

Then all you have to do is to click the "Submit" button to upload your assignment to the system. After you have done that successfully, you get the last screen, which confirms that you submitted your assignment. DON'T FORGET to print that page (just click the "Print Confirmation Page" button) and to attach it to the back of the hard copy of your assignment, which you have to submit to Dr Roger Mitton.

The confirmation page will look like the following screenshot.

REMEMBER that a hard copy of the above message is the only way to prove that you have used the system to submit your assignment.

After you have completed the submission you can return to the first page to submit another assignment using the button "Restart", otherwise you can close the window using the "Close Window" button.

Things to remember

- This system is not an alternative to hard copy. You should always submit a hard copy of your assignment to Dr Roger Mitton.
- It is mandatory to use the system to upload your assignment before you submit the hard copy of it to Dr Mitton. The confirmation page should be attached to the hard copy.
- Make sure that you upload the file containing the source code, NOT the executable OR the numbered listing you get from the compiler.
- If you want to submit more that one assignment you have to repeat the WHOLE process as many times as necessary.
- You CANNOT submit more than one file of source code for the same assignment.
- You can REsubmit an assignment as many times as you like. Only the latest submission before the deadline will get marked.
- If you make a mistake in submitting an assignment (eg. you submit assignment No.4 as No.3) then you have to resubmit (electronically) both assignments again (otherwise your latest submission of assignment No.3 will be invalid).

If there is a problem with the system you should inform Mr Sokratis Karkalas, email: sokratis@dcs.bbk.ac.uk