Requirements Document

Table of Contents

Introduction

The brief was given to design a questionnaire-generating programme for Sheffield University's Legal Practice Centre course in the Department of Law. Its main purpose is to allow students [18] to answer weekly exercises for specific topics [21] using a computer as an aid. This is in contrast to the approach used previously where exercises are handed out on paper, the student answers the questions and this is marked manually.

The proposed system automates this task by removing the need for the lecturer [13] to do any marking or for the students to mark their own work. The student instead answers the exercise [7] on the computer and a mark is returned immediately. The system also monitors certain statistics on the student's performance on these tests so that lecturers can see how individual students are progressing and if they have been doing the exercises that they should have been doing.

The system is for use on the Legal Practice Centre course that typically has 120 students for the duration of a one year long course consisting of two semesters. Each student takes four compulsory topics during Semester One and three chosen topics during the second semester. The chosen topics for the second semester are not known at the start of the year.

The system must allow for different topics and for topics to be changed, so it has to be possible to add and remove topics. Also, since the students on the course will change it will also be necessary for the system to add and remove students, especially at the end of the course when all students will need to be removed.

The system must maintain details of topics and their exercises, students and the topics they take and the required statistics for how students have performed on certain topics. This information will change often so it must be easy to change any of the information stored by the system.

The current manual procedure is as follows:

- A lecturer who teaches a particular topic would write several exercises for their topic.
- Each exercise consists of a series of questions relating to the current studies in that topic.
- These questions take the form of multiple choice, ordering of lists, pair matching and those requiring written answers.
- These exercises relate to specific weeks during the semester.
- The exercises are photocopied and handed out to the students in workshops.
- The exercises are either done within the particular workshop, or taken away for the student to do in their own time.
- The work is carried out individually instead of in groups, but students may collaborate with each other.
- These exercises are self-marked, with the lecturer either telling the students the correct answers or alternatively the answers can be found in the student packs given out for revision.
- No marks are recorded for the students so lecturers have no means of knowing how students are performing on the exercises or even if they have done them.
- The exercises carry no weight towards a student's final grade in the course; they are used purely as an aid to the students' learning.

Due to the fact that it would be extremely difficult for any system to recognise direct written English, it was deemed necessary for the questions to be asked in a different way. Four types of questions, such as multiple choice, have been identified. Further

details about these can be found in the ‘Elementary Data Modelling’ section of this document.

As it is the lecturer for a particular topic that sets the questions, it is necessary for any of the lecturers to be able do to this on the system, giving the system two types of users - students and lecturers.

A lecturer is likely to either add exercises as the course proceeds or alternatively add them all at the start of the course. Because of this it must be possible for exercises to be added, removed or modified at any time, as long as a student isn’t currently answering the exercise.

For the purpose of statistics the system must store a student’s mark for an exercise, but only for their first completed attempt. Because of this, it is necessary for the system to maintain details on students such as their username [22] (from their email address given out by the university) and their name. Also, the system must know what topics students are studying so that a student can only answer exercises on a topic they take.

The lecturer can add topics, exercises and questions (of the four described types), and also view topics.

In this system, the [X] notation after a word means that the meaning of that word or phrase can be found in the ‘Glossary of Terms’ section of this document. Only the first occurrence of every word in the ‘Glossary of Terms’ within the document has this notation after it.

Elementary Data Modelling

In this section, some of the concepts in the system about which data is to be stored will be defined in detail.

Topics:
- Each topic is uniquely identified by a topic name.
- The information to be stored about a topic is:
  - The topic's name.
  - Whether the topic is compulsory or not.

Students:
- All students have a unique username.
- The information to be stored about a student is:
  - The name of the student (first name and surname).
  - The student's student number [19].
  - The topics the student is taking.
  - The score the student achieved when they first attempted each exercise they have attempted.

Exercises:
- An exercise consists of one or more questions.
- There may be a different number of exercises for each topic.
- Each exercise in a topic has a unique number, specifying the week when the exercise should be undertaken.
- An exercise contains a certain number of questions.
- Different exercises can contain different numbers of questions.
- An exercise may have questions of different types.

Questions:
- The question types are multiple choice, matching pairs, ordering a list and multi-answer choice.

Multiple choice:
- A question is given with a list (up to six options in length) of possible answers, only one of which is correct.
- The lecturer can (optionally) associate with each answer a comment that will appear if that answer is chosen. This is to help students realise why their answer was wrong and to help them work out what the correct answer is.

Pair Matching:
- Two different lists are given (up to fifteen pairs in length) and a student must match the items in one list with the corresponding matching item in the other list.
- To help students who got a question of this type wrong, the system shall indicate which pairs were correctly matched.

List Ordering:
- A given list (up to ten items in length) must be ordered into a certain order.
- To help students who got a question of this type wrong, the system shall indicate which items in the list were in the correct position.

Multi-answer-choice:
- Similar to a multiple choice, a question is given along with a list of possible answers (up to six in length) but more than one of these may be correct.
- To help students who got a question of this type wrong, the system shall indicate how many possible answers were marked correctly.

User Characteristics

There are two types of users who will use this system – lecturers and students. Both could be considered to be competent on computers and on Windows-based [23] programs in general. However, none of them are experts on computers. This means that provided that our system is not complicated to use and it is clear how do something at every stage, there should be no problems with lecturers and students being able to use the system.

Functional Requirements

Listed below are the functional requirements of the system that we are to create. In the table, the following shorthands are used:

M – a mandatory requirement (something the system must do).

D – a desirable requirement (something the system preferably should do).

O – an optional requirement (something the system may do).

For ease of reading, the requirements have been split according to which of the four main parts of the system they relate to.

Recording details of students and topics:

Req. ID	Description	Priority
1	Lecturers can add topics.	M
2	Lecturers can remove topics.	D
3	Lecturers can edit details of a topic.	M
4	Lecturers can view details of a topic.	M
5	Lecturers can add a student.	M
6	Lecturers can remove a student.	M
7	Lecturers can edit details of a student.	M
8	Lecturers can view details of a student	M
9	Lecturers can easily remove all students from the system at the end of each year.	D
10	The system stores which topics each student is studying, which are entered by the lecturer.	M
11	The compulsory topics are automatically entered into each student's record.	D
12	Lecturers can add topics to a student's record.	M
13	Lecturers can remove a topic from a student's record.	D

Entering details of exercises:

Req. ID	Description	Priority
14	Lecturers can add exercises to a topic.	M
15	Lecturers can edit exercises.	M
16	Lecturers can remove exercises.	M
17	Lecturers can view the questions in an exercise.	D
18	Lecturers can add questions to an exercise.	M
19	Lecturers can edit questions.	D
20	Lecturers can remove questions.	D
21	Lecturers can enter a comment to be displayed for each answer to a multiple-choice question.	D

22	A lecturer should be able to control when students can attempt an exercise. A lecturer should be able to 'publish' an exercise when he or she has finished creating all of the questions for it.	D
23	New types of questions can be added to the system without any of the original system needing to be modified.	O

Students running the program:

Req. ID	Description	Priority
24	Students enter their unique username in order to use the program.	M
25	Students must be able to select an exercise and then attempt it.	M
26	When a student gets a question right, a comment is displayed (if one was entered).	D
27	When a student gets a question right, some kind of "reward" is displayed.	O
28	When a student gets a question wrong, the system informs them that they got it wrong.	M
29	When a student gets a question wrong, the system shows a comment (if one was entered).	D
30	The student proceeds to the next question after being told if they got the current question right or wrong.	M
31	At the end of the exercise, the system displays the student's score and a list of the questions that the student got wrong.	M
32	When the student has attempted all questions, but not got all of them right, the student can retake only the questions that they got wrong.	D
33	When a student has got all of the questions in an exercise right, they can retake the entire exercise.	D
34	Students should be able to use the system 24 hours a day.	D

Lecturers monitoring students scores:

Req. ID	Description	Priority
35	The first time a student answers an exercise, their score in it is stored.	D
36	Lecturers can look at a list of which exercises a particular student has attempted and the marks they got for those exercises.	D
37	Lecturers can look at a list of which students have attempted a particular exercise and each students' score for that exercise.	D
38	Lecturers can look at a list of which students have not completed a particular exercise.	D
39	Lecturers can print out a list of which exercises a particular student has attempted and the marks they got for those exercises.	D
40	Lecturers can print out a list of which students have attempted a particular exercise and each student's score for that exercise.	D
41	Lecturers can print out a list of which students have not completed a particular exercise.	D

Details of the hardware the system must run on**:**

Req. ID	Description	Priority
42	The system must run on 18 networked PCs running Windows 98 and on the computers on lecturers' desks.	M
43	The students should be able to run the question-answering part of the program from a remote computer.	D

Security issues:

Req. ID	Description	Priority
44	The system should only allow students on the Legal Practice Centre course and lecturers teaching on the Legal Practice Centre course to access the system.	D
45	Lecturers have to enter a password in order to use the parts of the program relating to altering student details, topic details and exercises.	D
46	Students have to enter a password in order to use the program.	O

Concurrency issues:

Req. ID	Description	Priority
47	The system should allow many students to access the same exercise at the same time.	M
48	Two or more lecturers should be able to alter exercises on different topics at the same time.	M
49	A published exercise can only be unpublished so that it can be modified, if a student is not doing that exercise at that particular time.	M
50	The system will not allow two users to update the same student record, topic, exercise or question at the same time (to prevent one of the changes from being lost).	M

Documentation:

Req. ID	Description	Priority
51	The system should provide a user manual for students.	M
52	The system should provide a user manual for lecturers.	M

Non-Functional Requirements

In this section, the non-functional requirements of the system are detailed. A non-functional requirement either describes how well the system should perform (a quality attribute) or a constraint or limit that the system must adhere to (a resource attribute). The non-functional requirements have been split into the categories of reliability, usability, efficiency, maintainability and portability.

Reliability:

Req. ID	Description
53	For a single user, the system should crash no more than once per 10 hours.
54	The system should produce the correct scores for students attempting exercises 100% of the time.
55	If the system crashed, it would behave perfectly normally when loaded up again.

Usability:

Req. ID	Description
56	A lecturer should be able to add a new topic to the system within 1 minute.
57	A lecturer should be able to add a new student to the system within 1 minute.
58	A lecturer should be able to add a question to an exercise within 5 minutes (will vary with question type).
59	A lecturer should be able to access monitoring statistics within 1 minute.

Efficiency:

Req. ID	Description
60	The system should load up within 15 seconds.
61	The time taken for the system to retrieve data from the server should never exceed more than 30 seconds.

Maintainability:

Req. ID	Description
62	The system should be designed in such a way that makes it easy to be modified in the future.
63	The system should be designed in such a way that makes it easy to be tested.

Portability:

Req. ID	Description
64	The client system [5] should work on the 18 computers for student use in the Legal Practice Centre, the computers on lecturers' desks and any students' computer that is connected to the Internet and has got at least Windows 95.
65	The system should be easy to install.

Dependencies and Assumptions

For our system it is assumed from information gathered from the client that they have a network of eighteen PC's running Windows 98. Each lecturer teaching on the course also has a computer on his/her desk, and they should be able to access the system from these machines as well.

It is assumed that this system will be capable of running any program created using the programming language Java [11]. The system is also dependent on the computers being able to communicate as part of the network so that everyone has access to the latest versions of exercises and so lecturers can see how students are performing in exercises.

The system is also dependent on there being somewhere to host [8] a server for the application that can communicate with all of the computers running the application via some means.

It is assumed that the computers used to run the application will have sufficient processor and memory capabilities. This includes that the computers must allow Java programs to run, have Windows 95 or later, and be fast enough (i.e. Pentiums) for the system to run in a reasonable amount of time. Fortunately, the computers in the Legal Practice Centre meet these requirements.

Constraints

Initially it was hoped that the application could be implemented with the use of servlets [17] via the Internet, meaning that it would in theory be accessible from anywhere in the world. However, after discussions with CICS [4] and the Department of Computer Science it was found that this was not a viable option, as the necessary technology was not available.

Another proposition that would have worked in a similar way to the one above was to use CGI scripts [3] instead of servlets. However, research into the workings of these led to the discovery that they are required to be implemented using programming languages such as C [2] and Perl [15]. Since nobody in the group has knowledge or experience in the languages required to implement CGI scripts, it was felt that they were not a viable implementation option due to the difficulty in creating them in the short time available to deliver the application.

Because of these constraints and the fact that all members of the group had knowledge of programming in Java, it was decided that both the client and server parts of the program should be implemented in Java. The client part will communicate with the server part using RMI [16].

To permanently store the data, both storing it in a file and storing it in a Microsoft Access [14] database were considered. It was decided in the end to store the data using a Microsoft Access database, as this would probably make data retrieval more efficient. The database would then communicate with the server via JDBC [12]. However, if for any reason we find that this method will not work, we know we can always use the file storage alternative.

User Interface Characteristics

All the users of the system have experience with Windows applications, so it was felt sensible for the system to look and feel like a normal Windows program. By this, we mean it is a 'graphical direct manipulation' interaction style. This style reduces the time necessary to perform certain operations, and we know that the client would value this, having, for instance, to enter in the details of approximately 120 students at the beginning of each academic year.

With this style, interactive objects such as buttons, list boxes and radio buttons are used to enable things to be done quicker. These are used in most Windows applications so we have no doubt that the students and lecturers who will use the system will understand how these types of object work.

In Human Computer Interaction [10], the following factors (heuristics guidelines) identified by [Nielson] are often used to try to create a user interface that is easy to use. In designing the user interface for our system, we shall try to follow these principles:

Simple and natural dialogue – having no irrelevant information, using a natural and logical order.

Speak the users' language – writing things in such a way that it is easy for the user to understand them.

Minimise the users' memory load – reduce the amount of information the user has to remember, by presenting it on the screen.

Consistency – make the user interface consistent to encourage users to experiment with things.

Feedback – ask for confirmation when about to perform an irreversible action and give an indication of how long an action will take.

Clearly marked exits – allow the user to exit to a higher level menu, or from an action, at any time.

Shortcuts – reduce the time taken to perform operations.

Good error messages – make error messages informative and suggest a solution.

Prevent errors – stop problems occurring in the first place.

Help and documentation – make well-written help available.

We intend to have the system finished well before the deadline so that we can install it for the client and they can use it. This will enable us to spot any problems they have in using the system, and the user interface could then be appropriately altered.

Plan

Below is a rough indication of when we plan to do which stages of this project.

Task	Week number										
	5	6	7	8	9	Easter		10	11	12	
Automating the testing process											
Implementing the system (and testing it as we go along)											
Final testing											
Source code sample											
User documentation											
Project commentary											

Glossary of Terms

[1] **Applet –** A small Java program embedded with a Web page.
[2] **C –** a programming language.
[3] **CGI scripts –** Common Gateway Interface, a method of communicating information between hosts and a server via the Internet.
[4] **CICS –** the university's Corporate Information & Computing Services. It handles all of the universities computer systems.
[5] **Client –** any computer communicating with the server.
[6] **DCS –** Department of Computer Science, our home department.
[7] **Exercise –** a given group of questions for a topic.
[8] **Host –** a computer hosting an application.
[9] **HTML –** hypertext mark-up language, the programming language used to create Web pages on the Internet.
[10] **Human Computer Interaction** (HCI) – 'the design, evaluation and implementation of interactive computing systems for human use.' Basically, HCI is about making computer systems as easy to use as possible.
[11] **Java –** an object-oriented programming language used to write the system
[12] **JDBC** – a Java technique that enables a Java program to get data from and pass data to a database.
[13] **Lecturer –** a person that teaches a particular topic/s, has access to change information on students, topics and exercises. No differentiation is made between lecturers.
[14] **Microsoft Access** – a database application that allows data do be stored in a structured way that enables it to be easily accessed.
[15] **Perl –** a programming language
[16] **RMI –** remote method invocation, a protocol for sending information over a network.
[17] **Servlets –** an internet based application, for communicating information between hosts and a server via the internet
[18] **Student –** anyone studying the course, who is taking topics contained within the system, and is capable of answering questions. A student is identified by their username, which is unique.
[19] **Student number** – see username [22].
[20] **TCP/IP –** transmission control protocol/internet protocol, a protocol for sending information over a network and the internet.
[21] **Topic –** a particular module within the course
[22] **Username –** each student has their own username based on the e-mail address given to them by CICS. These are of the form 'lwp<Year><Initials>', such as lwp00arc.
[23] **Windows** – the Microsoft Windows operating system that most PCs work using. There are many versions, the latest ones being Windows 95, Windows 98 and Windows 2000.

References

[Nielson] – Nielson, J., ‘Usability Engineering’, Academic Press, 1993.