WIZ – DYNAMIC WEB – BASED COURSE PRESENTATION SYSTEM

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER

ENGINEERING AND INFORMATION SCIENCE

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

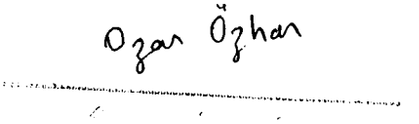
By

Uzan Yılmaz

August, 1997

WIZ - DYNAMIC WEB-BASED COURSE PRESENTATION SYSTEM

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER
ENGINEERING AND INFORMATION SCIENCE
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE



By
Ozan Özhan
August, 1997

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. David Davenport (Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. M. Erol Arkun

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Dr. Seyit Koçberber

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet Baray, Director of Institute of Engineering and Science

ABSTRACT

WIZ - DYNAMIC
WEB-BASED
COURSE PRESENTATION
SYSTEM

Ozan Özhan
M.S. in Computer Engineering and Information Science
Supervisor: Asst. Prof. Dr. David Davenport
August, 1997

The educational potential of the World-Wide-Web is clear. It provides not only a means of communication, but, just as importantly, a means for storing and conveniently accessing a massive amount of information. Such characteristics make it an ideal platform for distance education. One limitation however is the fixed nature of the web itself, making guidance and individualized presentation difficult. This thesis presents a project, the WIZ system, aimed at overcoming this problem. The WIZ system is a dynamic hypermedia system that presents interactive courses over the Internet. It monitors the progress of each student in order to provide a presentation that best matches the individual. Students have the freedom to browse the course material but can return back to the guided presentation by simply clicking on a button. In addition, the WIZ system exploits the communication (newsgroup and email) facilities of the Internet in order to provide a collaborative discussion environment for courses.

WIZ employs sound educational design and software engineering principles to generate affordable, individualized, interactive distance education, independent of time and location.

Key words: Distance Education, Internet, WWW, Intelligent Tutoring Systems, Dynamic Hypertext, Hypermedia, User Modeling

ÖZET

WIZ - DİNAMİK
AĞ TABANLI
DERS SUNUM
SİSTEMİ

Ozan Özhan
Bilgisayar ve Enformatik Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Yrd. Doç. Dr. David Davenport
Agustos, 1997

Dünya Çapındaki Bilgisayar Ağı (WWW) eğitim açısından büyük bir potansiyele sahiptir. Bu ağ sadece haberleşme imkanı sağlamakla kalmayıp, bilgilerin saklanması ve arzu edildiğinde bunlara kolayca erişilmesine de imkan vermektedir. Bu özellikleri, Dünya Çapındaki Bilgisayar Ağı'nı uzaktan eğitim için ideal hale getirmektedir. Mevcut sorunlar bilgisayar ağının sabit doğasından kaynaklanmaktadır. Bu sabit yapı kişiye uygun bireysel sunumu çok zor hale getirmektedir. Bu tez bu sorunun üstesinden gelmek için geliştirilmiş bir projeyi; WIZ sistemini sunmaktadır. WIZ sistemi İnternet üzerinde etkileşimli ders sunan bir dinamik hipermedya sistemidir. Bu sistem bireye en uygun ders sunumu sağlayabilmek için her bir öğrencinin dersteki durumunu sürekli olarak takip etmektedir. Öğrenciler ders materyali içerisinde istedikleri şekilde dolaşma özgürlüğüne sahip olmakla beraber, bir tuşa basmaları tekrar bireysel (kılavuzlu) sunuma dönüşü sağlamaktadır. Bununla birlikte, WIZ sistemi, derslerde, birlikte çalışma ve tartışma ortamı sağlamak için İnternet'in iletişim olanaklarını sonuna kadar kullanmaktadır.

WIZ sistemi yer ve zamandan bağımsız, hesaplı, bireysel, etkileşimli uzaktan eğitimi sağlamak için somut eğitimsel tasarım ve yazılım mühendisliği prensiplerini kullanmaktadır.

Anahtar kelimeler: Uzaktan Eğitim, İnternet, WWW, Akıllı Eğitim Sistemleri, Dinamik Hipermetin, Hipermedya, Kullanıcı Modelleme

To my family

ACKNOWLEDGMENTS

I am very grateful to my supervisor, Asst. Prof. Dr. David Davenport for his invaluable guidance and motivating support during this study. I would like to thank my colleagues Hüseyin Kutluca, Çağlar Günyakti, Alper Selçuk and Yücel Saygın for their friendship and technical support, my girl-friend Arzu Altunok who has provided me with moral support, and especially my family who has always been with me.

Finally, I would like to thank the committee members Prof. M. Erol Arkun and Dr. Seyit Koçberber for their valuable comments, and everybody who has in some way contributed to this study by lending moral and technical support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Educationalist, striving to improve teaching methods and efficiency, frequently look to technology for solutions. Indeed, new technologies often reshape expectations, needs, and opportunities within education. Clearly, the rapid development of computer technology is having just such an effect, as increasingly powerful machines come within school, training and even home budgets. While personal computers have certainly had a positive effect on education, their full potential is only just about to be realized. The real revolution will come when they are networked, or linked together, in such a way as to provide rapid connectivity and access, to people and information anywhere in the world. The Internet is a global network of networks connecting millions of users worldwide via many computer networks using a simple standard addressing system and communications protocol called TCP/IP (Transmission Control Protocol/Internet Protocol). The Internet promises to change not only how subjects are taught within the school, but even the nature of schooling itself. With the almost instantaneous nature of communication provided by the net, education need no longer be restricted to particular places, times or even people.

The educational potential of the world-wide-web is thus clear. Not only does it allow private and public communication, independent of time and place, but it also provides a means for storing a vast amount of information and a rapid and convenient means for accessing it. While many schools, colleges and, especially, universities, already make considerable use of the web as a local teaching

resource, its role as a distance education medium is only just beginning to be explored [HWV96]. A number of problems with using the web for educational purposes are already apparent. The most obvious difficulties result from the huge volume of information available. Locating relevant pages and keeping focused on the topic in question can be very problematic. As in all large hypertext systems, when students are given freedom to explore, they run the risk of becoming disoriented and lost, and hence missing important concepts. Careful design and structuring of material can help alleviate this problem, but is unlikely to cure it completely. Another difficulty of using the web for educational purposes, stems from the fixed structure of the links between pages. Tailoring teaching to the individual student has obvious advantages in terms of efficiency and retention, yet, unlike a human mentor, the web is unable to adapt its presentation to the individual learner. Many modern computer aided learning systems, particularly so-called intelligent tutoring systems, do achieve such individualization of instruction [Cos92]. They do this by maintaining a "student model," a representation of the presumed knowledge of the individual student. This information is used to guide the presentation of subsequent material, the model being updated by the student's answers to tests and quizzes. As a result, bright students can skip through the course very quickly, while slower learners can get the extra help they need.

This thesis describes a hypermedia system designed to exploit the advantages of the Internet for distance education. WIZ brings a new approach for presenting courses on the web. Current courses (tutorials) on the Internet lack individualization, there is no guidance and no variation in course content. What is presented is in the form of static html pages that are linked together in some order. What the WIZ system offers to students is dynamically varied presentation, freedom to browse, guidance when needed and communication facilities between the lecturer and students. In addition to these, the WIZ system brings the concept of student leveling. When starting a course for the first time, not all the students start with the same level of knowledge; for instance, a management department student taking a Pascal language course, may not have the same background programming knowledge as a computer science student. Thus, while the management student starts the course as a beginner, it should be more convenient for the CS student to start the same course at

an intermediate level. The detail and variety of the course content changes according to the student's level. Other than student leveling, course flow and content might be different for two students at the same level. This may happen according to the students' answers given to the tests in the course. Answers to each individual question give some idea of the student's current knowledge in the course. So, while slower learners review some sections, see more examples and answer more questions, bright students proceed faster in the course.

The next chapter, chapter 2, looks at tutoring systems, and introduces the concept of *distance education.* It discusses the advantages and disadvantages of using the WWW as a distance education medium.

Chapter 3 presents the design philosophy of the WIZ system and introduces the student modeling concept in intelligent tutoring systems.

Chapter 4 provides some basic technical background on the Internet necessary to produce interactive courses. This includes HTML (Hypertext Markup Language) forms and the CGI (Common Gateway Interface) specification.

Chapter 5 gives a picture of the WIZ system as whole, and introduces its components.

Chapter 6 examines the *Course Presentation Subsystem* of the WIZ system. This subsystem provides the interface between the students and the system during course presentation on the Internet.

Chapter 7 presents the database system in WIZ which is used to store the student model.

Finally chapter 8 is the summary and the future work section.

Chapter 2

Background

Almost as long as there have been computers, people have been interested in using them for education. The hope has been that computers could offer individual instruction, freeing up the time of the human educator and spreading educational resources farther. Computer assisted education has long been an interdisciplinary field, involving educators, computer scientists, psychologists, and artificial intelligence researchers. Research in this field has bumped up against many of the difficult problems of artificial intelligence, including natural language processing, heuristic search, and knowledge representation. Much of the early work in the field, then called Computer Assisted Instruction (CAI), tried to skirt these difficult issues.

2.1 CAI Systems

Traditional Computer Assisted Instruction programs are often described as simple branching or frame-based programs. The path of instruction was completely laid out in advance by the programmer. At runtime the computer made only the specified pre-programmed responses to previously anticipated student inputs. Wenger [Wen87], compares Computer Assisted Instruction programs to sophisticated books, which can be intricate, well-planned and entertaining. However, these systems are completely static and can only handle situations to

which they have a programmed response. This makes it impossible to generate new problems for the student or analyze the student's response. Most early programs considered Computer Assisted Instruction programs did not make any attempt to separate domain knowledge and pedagogical knowledge.

2.2 Intelligent Tutoring Systems

Later, a shift to the title of Intelligent Tutoring Systems signaled an attempt to focus on incorporating the ideas and methods of Artificial Intelligence. The range of programs that qualify as Intelligent Tutoring Systems is vast. The general idea is that rather than hard-coding responses to a student's possible actions, the program is instead given enough knowledge, both about the domain and about the act of teaching itself, to act autonomously.

A general model of many Intelligent Tutoring Systems includes three basic modules: domain knowledge, student model, and tutor. A rough test for Intelligent Tutoring Systems is that using their domain knowledge, they are able to solve the problems the pedagogical module puts to the students. The tutor module controls the interaction with the student, based on its own knowledge of teaching and comparisons between the student model and the domain knowledge. Intelligent Tutoring Systems can range from very primitive to extremely sophisticated in these modules and their interactions. Many Intelligent Tutoring Systems focus on one aspect of the larger teaching problem, improving one of the three main modules, for example, perfecting student models so that tutoring modules can better tailor their performance to the individual.

2.3 Distance Education

2.3.1 Definition

Distance education is an umbrella term that covers many different types of teaching and learning activities, not only locally, but also throughout the world.

A fairly commonly accepted definition of the term has been developed by the International Council for Distance Education [ICD97].

Distance education is a mode of instruction in which the student and teacher are separated in time and/or space and where two-way communication takes place through non-traditional means for the most part.

2.3.2 Need for Distance Education

Improving and expanding education are essential ingredients of any national development policy. Countries look to the future's well educated generations as the best way to improve their overall social and economic standing. National educational programs mainly rely on conventional or formal education methods, the sort of methods based for the most part on the traditional classroom contact. However, conventional methods of education are expensive particularly in the Third World countries that have no easy access to conventional schools, and may not be suitable for segments of the population such as adult learners who must combine studies and work. For these reasons, distance education is a rapidly expanding field nowadays.

2.3.3 WWW as a Distance Education Medium

As mentioned in [Syn96] and [BP96], there are many advantages of using the WWW as a teaching tool. First of all it is very easy to put information on the WWW. All you have to do is prepare a Web page containing text, pictures, applets, etc. and a few straightforward code instructions in HTML (Hypertext Markup Language-see chapter 4), a coding system that can be mastered in a few hours. Course materials can be updated immediately. Unlike paper documents, which must be revised, retyped, reprinted, and redistributed every time you update them, a Web page can be updated immediately by modifying and reloading the file. The next reader to access the page gets the updated information. Students can access the WWW 24 hours a day. So, if they miss a class, they don't need to wait for the next class meeting to catch up, or if

they lose or discard a subject, they can quickly replace it. Students with dial-in access to the Internet can work at home, and any student can access the WWW from any of several on-campus labs. The instructor can add new pages or update pages from remote locations, he/she doesn't need to be in the office. Since the Web pages are always available electronically, there is no need to print them. The students just have to know where the web pages are and connect to those pages when necessary. They can, however, send any Web page to a printer from the Web browser if they want a paper copy. Another important benefit of the WWW is the ability to link to other information resources. This feature known as hypertext allows students to take advantage of unique sources of information such as research institutions, libraries, government agencies, and foreign Web sites, as well as sites maintained by colleagues at other universities. Web pages can be used as lecture aid (in conjunction with a projection device or by the entire class in a lab setting). Maybe the most important thing is that web pages can allow interaction with students, whereas a book page can not. As will be seen later in this thesis, the WIZ system uses this property of web pages to provide interaction with students.

A number of problems with using the web for educational purposes are already apparent. The most obvious difficulties result from the huge volume of information available. Locating relevant pages and keeping focused on the topic in question can be very problematic. As in all large hypertext systems, when students are given freedom to explore, they run the risk of becoming disoriented and lost, and hence missing important concepts. Careful design and structuring of course material can help alleviate this problem, but is unlikely to cure it completely. Another difficulty of using the web for educational purposes, stems from the fixed structure of the links between pages. Tailoring teaching to the individual student has obvious advantages in terms of efficiency and retention, yet, unlike a human mentor, the web is generally unable to adapt its presentation to the individual learner.

Preparation of course material can be a very time consuming process. Although it is not difficult to set up and maintain a series of web pages, it can take a lot of time. Depending on the scope of the courses, the level of detail you want to present, and the frequency of updating, the course creator may end

up spending much more time than he/she expects. Another disadvantage of the WWW is that anyone can copy your work. Password protection can limit this, but not prevent it completely. And finally, although no special training or skill is required to browse the WWW, students who are completely lacking in basic computer skills may be disadvantaged in terms of WWW access.

This thesis describes a system intended to overcome many of these disadvantages while retaining the best features of previous instructional systems. The next chapter describes the design philosophy of the WIZ system.

Chapter 3

WIZ Design Overview

The design philosophy of WIZ, is to divide a course into modules where each module corresponds to a domain concept [OD96]. This philosophy has been used in TUTOR [DGB89] and has shown success when compared to the conventional approach in tutoring systems. The WIZ system takes this basic design philosophy (originally proposed in [Dav90b],[Dav90a],[DGB89]) and applies it to the Internet.

Traditionally systems have been frame-based, a frame corresponding to a screen presenting the material to be learned, and/or some appropriate question. The difference between a frame and a module is that a frame is basically equivalent to a screen display, a little bit like the leaves of a book. Its content is set/limited by the screen size, not by the material being taught. In contrast a module is bounded by a domain concept and may be composed of any number of screen displays. It is also interesting to note that most conventional systems group frames into lessons, and lessons into courses. This seems natural enough, courses in school are given in lessons so why not on the computer too. Should not modules likewise be grouped into lessons? Actually no, because the contents of a lesson are rarely defined by the course content but rather by time considerations. As one of the major advantages of computerized tutoring systems is self-paced learning, this is something of an enigma to the concept of lessons. Therefore the module remains the only logical sub-divisor of a course in WIZ.

Having decided to divide course material into modules, the question arises as to the order in which these modules should be presented. As they are based on pre-requisite relationships there is a certain amount of natural ordering incurred, however this is rarely complete. Two possibilities exist, either the lecturer can decide on the overall ordering when designing the course, or it can be left to the student and the system to select whatever sequence they consider appropriate while learning is in progress. While it is tempting to allow the student complete freedom in sequence of material he/she learns, subject of course to pre-requisites constraints, this is not always as good an idea as it may sound. For one thing it places an extra burden on the student, who may experience great difficulty and even stress, when called upon to decide something which is essentially beyond his comprehension. Another problem is continuity, or linking between modules or groups of modules. Although not strictly necessary it is an observed fact that people learn more efficiently if they have some idea of the purpose and progress they are going to be expected to make. This is usually in the form of an introduction. Often too it is useful to summarize a set of concepts after teaching, so as to reinforce the main points, this is usually in the form of a summary. Such things do not fit exactly into the *module equals concept* model proposed so far, since they are not themselves concepts to be learned. One solution is to create special purpose modules for just these purposes, such as introduction, summary and transition modules. However special purpose modules can only be used if the course sequence is more or less known. Therefore it was decided to take the former option and have the teacher to specify the sequence of presentation in the course. This sequence is taken to be the default course flow; it may be overridden by the student or the system during the actual course presentation.

A course in WIZ contains references to the modules it consists of. Unlike frame systems, this allows the same module to be used in different courses and fits in with the requirement that courses be easily modifiable. A course is viewed as a linear sequence of modules however it is important that the student still have the opportunity to review material he has covered. Such a facility can be treated like an interrupt in the default flow of the course, such that at any point the student may request the system to redisplay a module. When the student finishes the module he/she wants to review, returning back

to the point of interruption and continuing with the default flow is as simple as clicking on a *default flow* button. Likewise, a similar technique is used if it is decided that the student should undertake remedial work, the only difference is that the system initiates the interrupt not the student. How and when to do will be considered in the following section.

3.1 Student Model

Knowing whether the student should undertake remedial work requires questioning him/her. Questioning has two possible purposes, formative and evaluative. Although not essential, it makes sense to separate these two functions. Thus we can envisage a system which presents the material, then questions the student with a formative aim, i.e. with the intent of reinforcing and redirecting learning as necessary. On the other hand, questions with an evaluative aim have the intent of finding out how successful the student was in actually learning the material. Whereas formative questioning is usually closely related to individual concepts, evaluative questioning is in effect an examination and may or may not be associated with an individual concept. The basic operational difference between the two is that while in the formative phase every effort is made to discover and correct the students weaknesses, usually through appropriate feedback in response to the student's answers, in the evaluative phase no such feedback is required. In the latter case scoring is important. Conversely scoring during the formative phase has no meaning in the classical sense, making a mistake here is simply part of the learning process, however incorrect responses obviously indicate difficulties. If the number and seriousness of the wrong answers is significant then remedial work should be undertaken. While the definition of *significant* is rather arbitrary, a simple scoring mechanism can serve to control this stage of the process. Before going on with scoring, it is important to determine the place of the formative questions. Because formative questioning is important and since it is closely related to the concepts being taught, it seems to make sense to include such questions at the end of the module containing the corresponding material.

Returning back to the subject of scoring, answers to the formative questions

update values in a student model. In essence the student model is simply a representation of what the student knows or does not know, at any particular time. It can be used during teaching to control the course flow, i.e. deciding when to present remedial material etc. and subsequently as a record of the students progress.

Designing and maintaining a model of the individual student user, is arguably the most important aspect of any educational software. While there has been considerable research in this area, it stills remains more of an art than a science. There are several forms the model could take. McCalla [McC92], classifies them along three basic dimensions: temporal scope, cognitive scope and generative ability. The first determines whether the model simply retains a snap-shot of the student's current knowledge or whether it keeps a record of the long-term evolution of this knowledge. Along the cognitive dimension, systems vary from ones which maintain deep models, which divide the domain into sub-parts and retain details of each part separate, to shallow ones which simply record overall test scores. Finally, models having generative ability are potentially the most sophisticated. Typically, they would store domain knowledge in the form of machine-usable rules, then, when trying to understand a student's response to a question, they would try to produce the same result using some subset of these rules. Those rules found necessary would be presumed to be the ones the student had also used and hence knew. Note that producing a set of rules representing the material to be taught is currently an extremely difficult, if not impossible, task for most subjects. For this reason, there are no general purpose tutoring systems which employ generative models. Conversely, if the domain knowledge is stored in the form of text and pictures it becomes unusable by the machine, so that understanding the student's state of knowledge and hence producing sophisticated teaching behavior, becomes impossible. WIZ aims to bridge this divide to produce a semi-intelligent, but quite general purpose teaching mechanism.

Intuitively it seems sensible to represent the student's knowledge of a subject in terms of the concepts composing it, and since these are readily apparent in the WIZ philosophy, it is simply a matter of *mapping scores to modules*. The system should allow a response to update the scores of any module in the

course, including its own module's scores. The rationale behind this is that questioning may uncover difficulties, not just in the concept presently being taught, but also in the pre-requisite concepts. This may simply be a matter of forgetfulness, or it may be related to more serious misunderstandings which were not uncovered at the time. Allowing evidence to be collected over a period of learning is likely to be more effective in locating such problems.

3.1.1 Control Mechanism

The final part of the design is the control mechanism itself. This is the part which determines which module should be displayed, when remedial work or revision is necessary.

Our design has come to a set of modules, each of which contains some course material together with a set of questions. Which modules, and in what sequence they will be presented, is also available (called the default flow). At its most basic, the control program is simply required to follow this sequence, ask questions, get student responses and update scores as directed. The student may either give a correct answer or an incorrect answer to a question. Therefore we have thought of having two types of scores for every module, namely the positive and negative score, where a correct answer increases the positive score and an incorrect one increases the negative score. If the *negative score over positive score exceeds the specified thresholds* remedial actions have to be initiated. This essentially means displaying, or redisplaying, material from an appropriate module, the one indicated by the scores, and possibly asking further questions, before returning to the original sequence. Other than the control mechanism, the student should also have the freedom to display or redisplay any module he/she wishes.

As each module may contain a set of questions the control mechanism must determine which ones to ask, when, how, and to whom. This is an important and a difficult problem for while tailoring of the teaching material to the needs of the individual student is possible, the major individualizing factor would seem to be questioning. Therefore questions are designed to have a difficulty

level in the WIZ system so that only questions suitable to the student's background and knowledge are asked during course presentation. This ensures that students receive questions which challange them and hence provide further learning opportunities.

Chapter 4

Internet Basics

4.1 HTML and CGI Programming

HTML (Hypertext Markup Language) is the standard used in displaying Web pages on the Internet. In this chapter, HTML, and in particular the *form* feature of HTML will be presented. *Forms* allow the creation of interactive Web pages. Later the CGI (Common Gateway Interface) specification will be presented. CGI programs are used in handling and processing the data entered in a *form*.

4.1.1 HTML Pages

Information on the Internet is stored electronically, in the form of web pages which can be any size in length. Like a book, each page may contain text and pictures, but it may also contain active elements such as animations, audio and video sequences, information request forms, database access and, most significantly, 'links' to other pages. Following a link to another page is as simple as pointing at it with the mouse and clicking. This interconnecting of pages, technically known as hypertext or hypermedia, gives the impression of a huge spider-like web, hence the phrase world-wide-web. Although the WWW is a relatively new phenomenon, there are already hundreds of millions

of pages around the globe. These pages have been created by individuals, organizations, educational, governmental and commercial establishments, and anyone connected to the network is free to browse (look around) this vast storehouse of information and even add to it.

The pages on the Internet are in HTML (Hypertext Markup Language) format. HTML is an evolving language which is used to construct web pages that can be viewed by WWW browsers. In practical terms, HTML is a collection of platform-independent styles (indicated by markup tags) that define the various components of a World Wide Web document. For example characters written between and markup tags are displayed as bold characters on WWW browsers. More information on HTML can be found from [fSA96], [BLCM95], [Wer96].

4.1.2 Forms in HTML

Forms are dictated between <FORM... >... </FORM> tags in HTML. Forms add the dimension of interactivity, that bring the Web to life. HTML pages can be glitzy and glamorous, but without forms, they are 'read-only'. Forms let users send back information to you. In WIZ, forms are very important in the sense that they provide feedback from students.

A sample HTML code for a form is given in table 4.1 and it's appearance in a Web browser can be seen in figure 4.1. This sample form contains all types of elements that provide interaction on the Internet. As can be seen from the figure, a form is a GUI (Graphical User Interface) with text entry fields and areas, pull-down menus, buttons, checkboxes, scrolling lists, etc. When you write a form, each of your entry fields should have a *name*. When the user places data in these fields and presses the submit button[1] of the form, the *name* and *value* of each entry field is encoded into the form data and sent to a CGI program for processing. A CGI program decodes the form data to obtain the *name* and *value* pair of each entry field.

[1]In every form there exists a submit button, which when pressed, transmits all the user entered information in the form to a Web server for processing by a CGI program

```
<HTML>
<BODY>
<H1>Registration Page</H1>

<FORM METHOD=POST ACTION="/cgi-bin/WIZ.cgi">
<P>Name: <INPUT TYPE=text NAME="name" SIZE=16>
Surname: <INPUT TYPE=text NAME="surname" SIZE=16>
<>Address: <TEXTAREA NAME="address" ROWS=2 COLS=40>
</TEXTAREA>
<P>Country: <SELECT NAME="country">
<OPTION SELECTED>Turkey
<OPTION>USA
</SELECT>
<P>Sex:
<br>Male <INPUT TYPE=radio NAME="sex" VALUE="male">
Female <INPUT TYPE=radio NAME="sex" VALUE="female">
<P>Which OS do you use?:<br>
<INPUT TYPE=Checkbox NAME="win95" VALUE="ON">MS Windows 95?<br>
<INPUT TYPE=Checkbox NAME="unix" VALUE="ON">Unix?<br>
<INPUT TYPE=Checkbox NAME="solaris" VALUE="ON">Solaris?
<P>Programming Languages: <SELECT NAME="pl" MULTIPLE>
<OPTION> C
<OPTION> Pascal
</SELECT>
<P><INPUT TYPE=submit NAME="submit" VALUE="Submit">
</FORM>

</BODY>
</HTML>
```

Table 4.1: A sample form in a web page

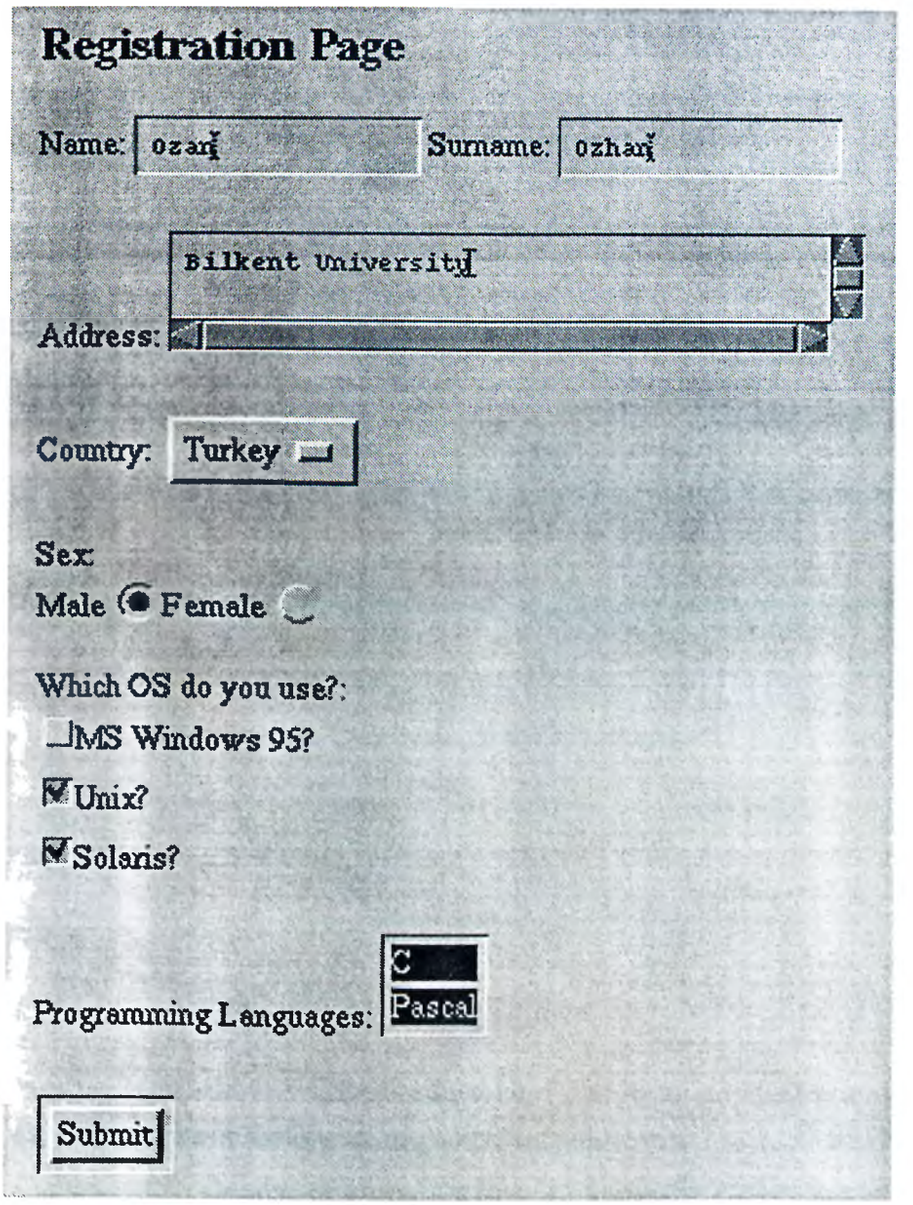


Figure 4.1: A sample form on a Web browser

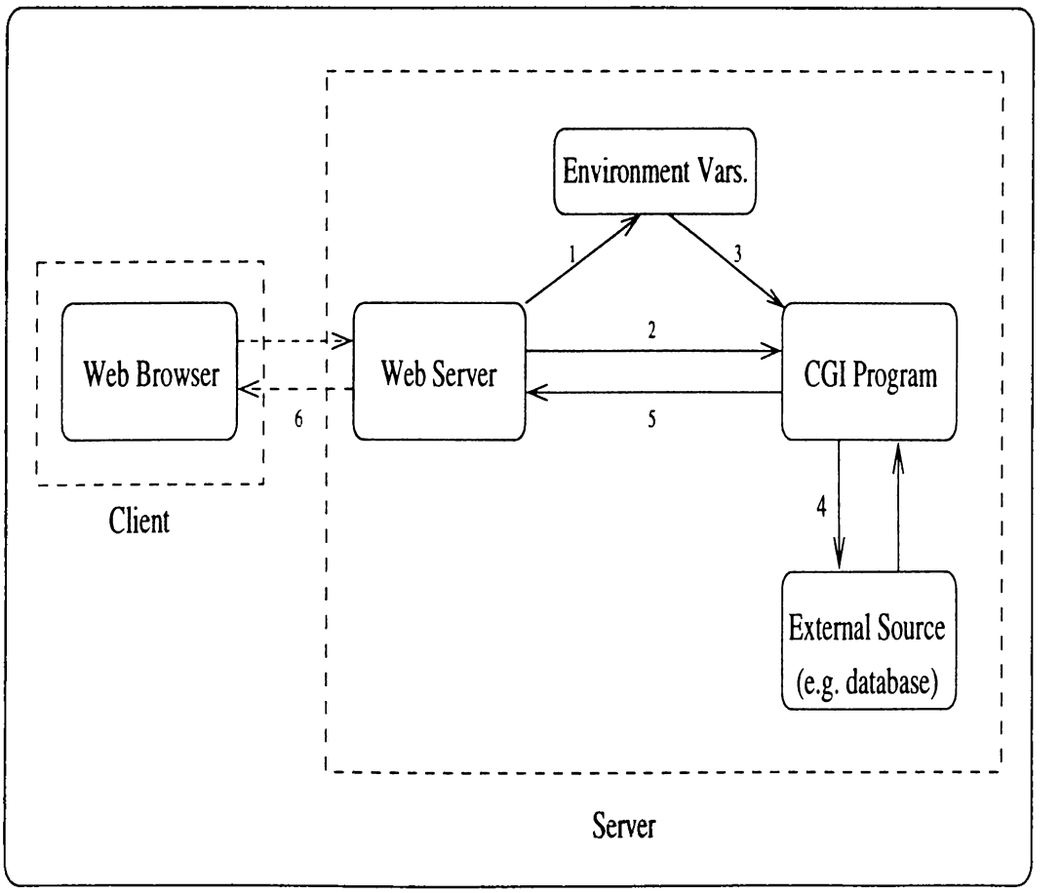


Figure 4.2: Execution of a CGI program

For more information on the HTML structure of forms see appendix A:

4.1.3 CGI Specification

CGI stands for the Common Gateway Interface. The Common Gateway Interface (CGI) is a standard for interfacing external applications to HTTP servers[2] effectively, extending the server's functionality. In particular, when you submit an HTML form, the server needs to have a CGI program which will take the information you submitted and properly process it.

CGI programs handle data sent by a client and as a reply, return the appropriate document or generate a document on the fly. In this way Web server

[2]A HTTP (Hypertext Transfer Protocol) server is another name for a Web server

can send information dynamically to the client (e.g. from an SQL database), and act as a mediator between the two to produce something which clients can use.

Execution of a CGI program is given in figure 4.2. The figure labels the steps taken in the execution of a CGI program. A CGI program is executed when some *form* data is submitted from a client. The Web Server receives the *form data* and starts the execution process:

1. The Web server makes environment variables available including;
   - form variables entered by the client
   - a series of CGI environment variables
2. The Web server fires the CGI program
3. The CGI program reads:
   - form variables entered by the client
   - a series of environment variables
4. The CGI program accesses external resources such as a database system, a file etc.
5. The CGI program returns HTML data using standard output
6. The Web server returns the HTML data to the client

For communicating information with CGI programs, the Web server defines a number of environment variables. The environment variables can be loosely grouped into three categories [Byt]: *program input* variables, *remote client* variables, and *HTTP server* (Web server) variables. These variables hold various information about web server, client and the data that is submitted by the client. Appendix B provides further information on environment variables.

CGI programs can be written in almost any language. Some of the more popular languages include C, C++, Java, Ada, Fortran as compiled languages, and Perl, TCL, any Unix shell for interpreted languages. For more information on CGI programming see [Doc96], [Man97], [Byt].

[Ibr94] mentions about the sophisticated uses of the WWW for distance learning and gives an interaction scheme between the remote user and some

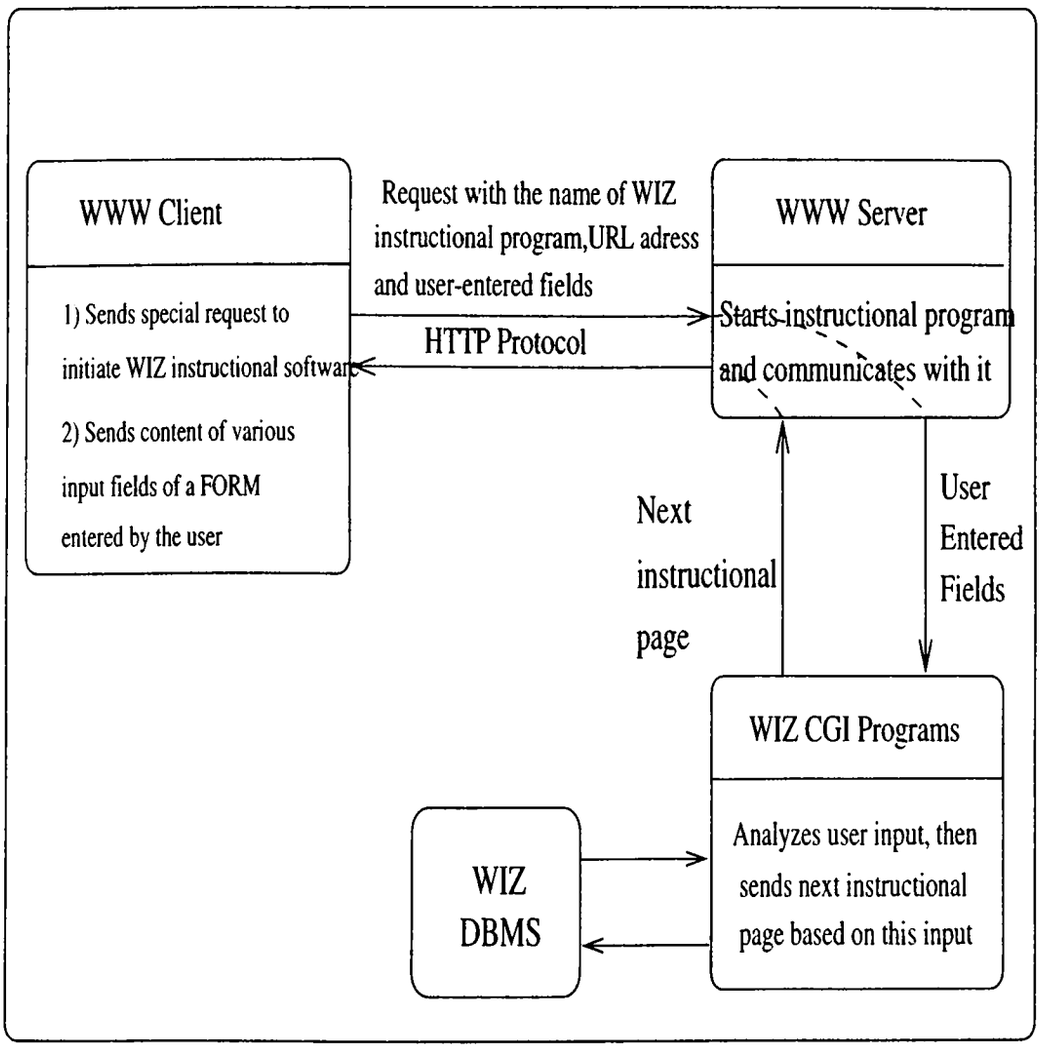


Figure 4.3: Interaction Scheme between User and WIZ via Internet

instructional software via WWW. The interaction scheme of the WIZ system is quite similar and is given in figure 4.3. In this figure, the instructional program, WIZ, is seen to be a CGI program. In fact, the WIZ system actually consists of two CGI programs; one for registering students to the system and another for course presentation on the Internet. Both of the CGI programs are implemented in C language and described in the following chapters.

Chapter 5

WIZ Implementation

This chapter presents an *implementation overview* of the WIZ system whose design was outlined in chapter 2. Subsequent chapters will be concerned with more technical details and the user interface of the WIZ system.

As mentioned before, WIZ is a Internet based course presentation system. From an educational point of view WIZ is quite conventional; present some course material, show examples, give some tests to determine the knowledge of the student and finally decide on which piece of course material to present to the student next. This sequence is embodied in a *module* in WIZ. A course is a sequence of modules. A module consists of a set of *web pages* plus *test templates* that contain questions about the information presented inside. Every module has a specific learning outcome. Generally a module explains a particular concept to the student. A module is the core structure in WIZ and the *student model* is defined over this set of modules.

5.1 Overview

A general view of WIZ system is shown in figure 5.1.

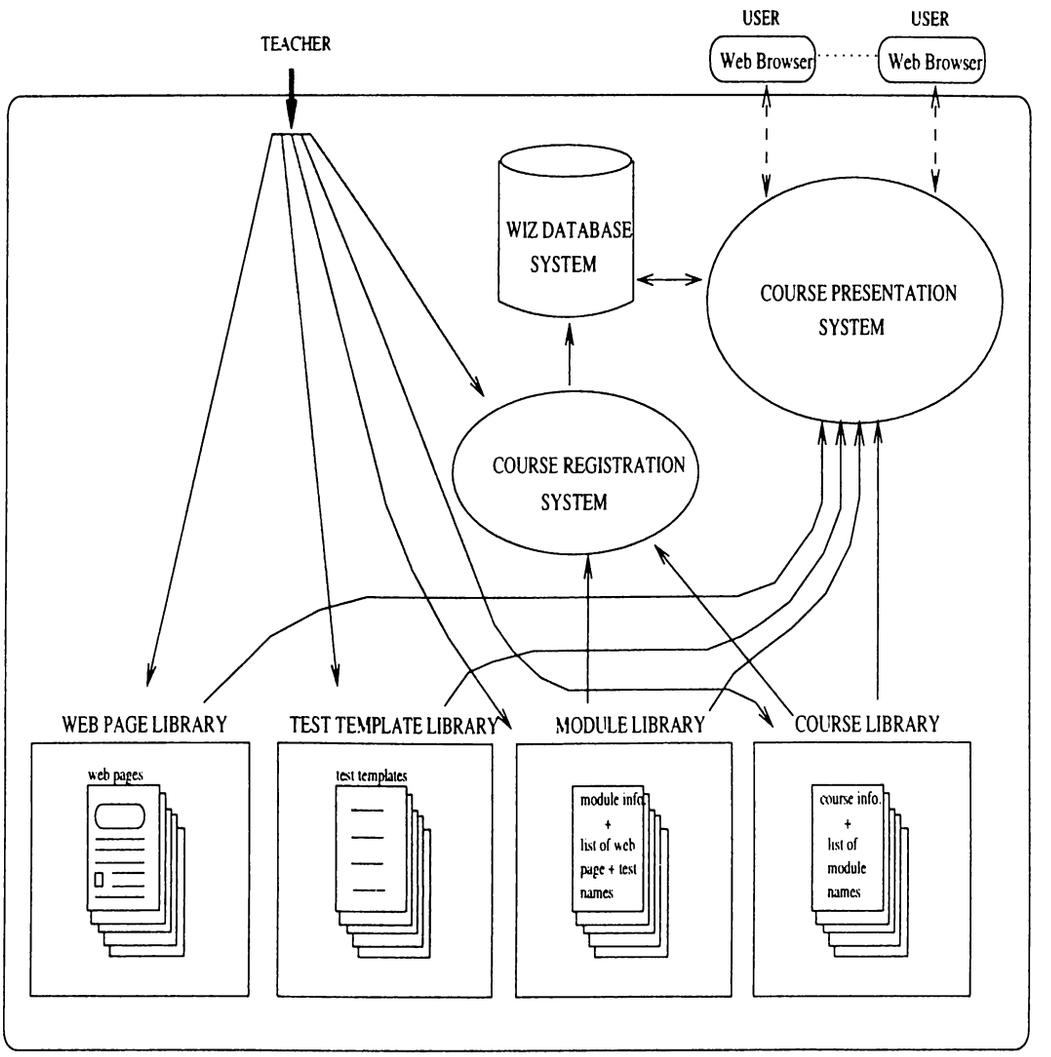


Figure 5.1: Basic Structure of WIZ

5.1.1 Libraries

As can be seen from figure 5.1 there are four libraries; the web page, test template, module and course libraries. Libraries are in fact directories. For example, the current default path of web page library in WIZ is '/project/webpage/'. Path settings can be changed from a library file called the 'settings.h'. All the web pages are stored under this directory. Web pages are recommended to be stored in some logical subdirectory order. What this means is, for instance, web pages of say 'Perl Tutorial' course are stored under 'project/webpage/perl/' directory and the first page, say 'page1.html', of the course is referenced as 'perl/page1.html' in any module. It is not recommended to have a situation where all the web pages of different courses are mixed together under the same directory. Having subdirectories allow web pages of a course to be grouped together under a subdirectory. This helps a course creator to find a web page more easily during course preparation. The default path settings of other libraries are: '/project/test/' for test templates, '/project/module/' for modules and '/project/course/' for courses. The same subdirectory logic is recommended to be used in test template and module directories.

5.1.2 Course Preparation

Course preparation is the most time-consuming process in WIZ. However, once the course is prepared, it can be used for years and modifications in the course material can be made as needed.

The course preparation process is realized as follows: First design the course material and prepare them in the form of web pages. Web pages can be written in HTML format using any text editor. Alternatively HTML editors can be used such e.g. Microsoft's Frontpage for preparing web pages.

Later relevant web pages are grouped together to form modules. Each module definition contains a set of web page names. The number of web pages in a module and the length of web pages are determined by the course creator. But it is important to notice that the number of web pages, and the length of

web pages should be reasonable, since long pages are boring to read from the student's point of view.

Modules do not only contain web pages. Test templates can be appended at the end of modules. Test templates are optional in modules. Tests are used in evaluating the student's knowledge. According to the student's answers, if it is determined that the student has *some missing* concepts he/she is forced to review related modules. If no tests are used in a course, then the course flow becomes a normal sequence of course material presentation where every student follows the same course flow. Test template names are recommended to be placed in modules except for some special purpose modules which handle introduction and transitions. Preparing test templates is a task which requires considerable attention. Test templates may contain different types of questions. Each question contains some modification information on the student model. Test templates are explained in *Test Templates* subsection of this chapter.

Student Leveling

As mentioned in the introduction chapter, there is a student leveling system in WIZ. Students start a course in one of three levels; *beginner level, intermediate level* or *expert level.* While a student who has no background in a course should start as a beginner, another one who knows the course more or less should start as an expert. This leveling system is optional. Student leveling does not have to exist in a course, but if it is going to exist, then each module should contain different sections; each section containing web pages whose number and content are designed for its level of student. These sections will be explained in detail in the *Modules* subsection of this chapter. Briefly, if student leveling system is going to be used, three copies of the same course with different detail should be prepared. As a result, course preparation is a time consuming process where the course creator should be patient, attentive and should have a good knowledge about the course.

5.1.3 Web Pages

Web pages are in HTML format. They can contain links to other web pages, text, picture, audio, video, animation, applets etc. Web pages in WIZ can contain all the elements listed above. As mentioned before web pages are stored in web page library. There is no way for a client to reach, read or update the web page library[1]. During course presentation, a web page is read from this library by the *Course Presentation Subsystem* which is a CGI program, and printed to the client's browser through the WWW server by this CGI program. If a web page contains pictures or applets, then these elements should be in a place where they can be accessed by browsers. All these elements should be stored under *public_html* directory of the Unix system where they can be accessed and read. Notice that Web browsers can only access files and directories which are under *public_html* directory on the Internet. Otherwise, for example if a picture (gif file) in a web page is stored in the web page library, it can not be seen on the client's browser since the client's browser can't access that directory.

WIZ presents web pages that are available in the system. It may seem to be a closed corpus system[2] as dictated in [IF95] but it is not the case since web pages in WIZ may contain hypertext links to outside information resources. For example, when writing a web page, the lecturer (course creator) may put links to some outside resources for further information about a concept or a subject. For instance, a link to say 'sample.html' at site 'http://sample.com' can be inserted in a WIZ web page as <A HREF= "http://sample.com/sample.html" TARGET=new>Sample Link</A>. The 'TARGET=new' phrase is highly recommended, since this opens a new web browser window and displays the specified page on that window. This enables the external information resource to be displayed in another window and thus the course presentation in the initial window is not disturbed.

[1] Also there is no way for a client to reach, read or update any file in the *test template*, *module* or *course* library

[2] A closed corpus system is the one which presents information in itself

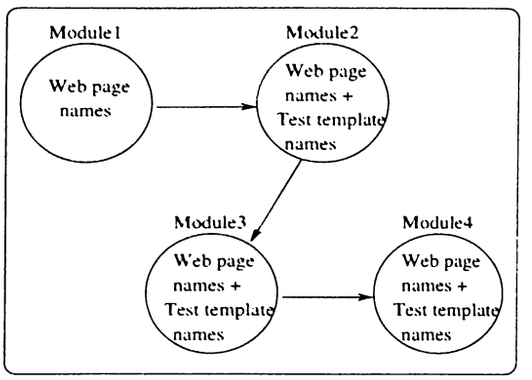


Figure 5.2: A sample course with four modules

5.1.4 Test Templates

Tests are prepared after course web pages are prepared and modules are formed. Tests are different than web pages. Web pages are in HTML format but tests are not. Thus they are referred to as test templates. Test templates are translated into HTML during course presentation. Test templates contain questions and their answers. For every question, the question type, level of difficulty, choices (if there are any), the modification information on the *student model* and the answer to the question are held.

Test templates are used for two purposes. One for reading and interpreting questions into HTML format and displaying to user. Another for answer comparison which is when a user answers the displayed test and submits his/her answers, the same template is used in updating the *student model.*

A small sample course with four modules is given figure 5.2. A sample test template for *Module3* of figure 5.2 is in table 5.1.

This test template contains only one question. There is no limit for the number of questions in a template. The template begins with a line that tells the subject of the test and ends with <EOT>. Main tags used in a test template are <questionlevel> tag, <type> tag, <question>...</question> tags and <answer>...</answer> tags. The <questionlevel> tag is followed by the level of difficulty of the question, it is *intermediate* level in this example. The <type> tag is followed by the type of the question, it is *multiple* choice

```
Questions about Turkish History?
<questionlevel>
intermediate
<type>
multiple
<question>
In which year was Turkish Republic established?
<choices>
1922
1923
1900
1815
1820
</question>
<answer>
2
<1>
-0.5
history/Module2
<2>
1
<3>
-1
<4>
-1
history/Module2
<5>
-1
<unanswered>
-1
history/Module2
</answer>
<EOT>
```

Table 5.1: A sample test template with one question

question in this example.

Tags and information that lay between <question>...</question> tags will be called the *question section.* Tags and information that lay between <answer>...</answer> tags will be called the *answer section.* The tagging structure of question and answer section vary according to the question type.

A test template is used first for displaying the questions and then for evaluating the answers to the questions. When displaying a test, question level, type and *question section* sections are used and the *answer section* is skipped in the test template. When evaluating the answers, question level, type and *answer section* sections are used, the *question section* is skipped.

Question leveling, question types, *question section* and *answer section* will be explained in the following subsections.

Question Leveling

Course preparation section of this chapter mentions about student leveling in WIZ. Since there can be different levels of students in a course, there has to be different levels of questions appropriate to each level of student. Question levels available in WIZ are beginner, intermediate, expert and mandatory levels. Mandatory level questions are displayed to every student whatever the student's level is. Other than this, beginner level questions are only displayed to beginner level students, likewise intermediate level questions to intermediate level students and expert level questions to expert level students. There can be many different levels of questions in a test template, however only the questions whose level match with the student's level plus the mandatory questions are displayed to students during course presentation.

Question Types

Question types available in WIZ are, *multiple* which is a set of radio button elements where the user can select only one choice, *classic* which is an edit box element where the user can type text, and finally *select* which is a select many

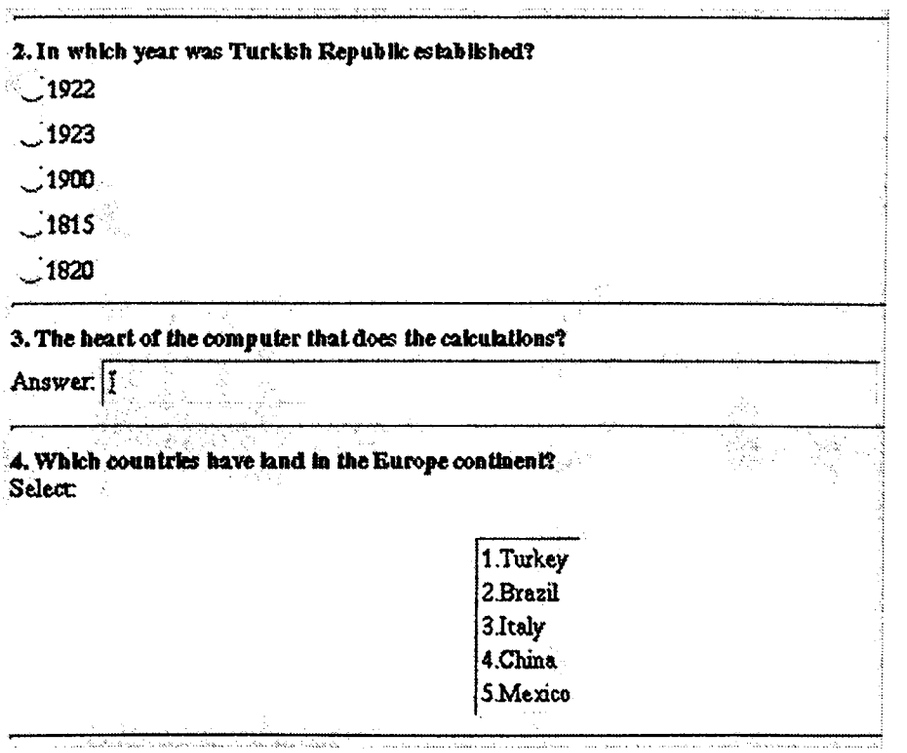


Figure 5.3: Question types available in WIZ

element where the user can select more than one from a set of alternatives.

Figure 5.3 illustrates each of these question types. The WIZ system provides every type of question that forms allow. A question type whose answer will be entered into an edit box is the same with a question whose answer will be entered into a textarea by means of functionality. Likewise, a set of radio button elements and a select one element are the same, and a set of check box elements and a select many element are the same.

Question section

If we examine the *question section* of the sample question in table 5.1, it can be seen that this section contains the question text and the choices that start with <choice> tag. In this example, the question type is *multiple*. Question section structure is the same for *multiple* and *select* type of questions. On the other hand, *classic* type questions do not have a choice section, they only have

multiple type	*classic type*	*select type*
<question> In which year was Turkish Republic established? <choices> 1922 1923 1900 1815 1820 </question>	<question> The heart of the computer that does the calculations? </question>	<question> Which countries have land in the Europe continent? <choices> Turkey Brazil Italy China Mexico </question>

Table 5.2: Question sections of three different question types

the question text. Sample question sections for the three types can be seen in table 5.2.

Answer section

Student model[3] modification information lays in the answer sections of questions. Let's examine the answer section of the question in table 5.1. In this example, the answer to the question follows right after the <answer> tag. In this question type, the answer, 2 means the second choice in this question is the correct answer. Right after the correct answer follows the *modification sections* for every choice plus the *modification section* for unanswered question which makes six modification sections. The *modification sections* update the student model. Out of these *modification sections* only one of them, that is the student's choice is visited. For example, if the student chooses the fourth choice as the answer, the *modification section* that starts with <4> is visited, other *modification sections* are skipped. Now let's examine this *modification section*:

<4>
-1

[3]The *student model* stores student's positive and negative scores for every module in a course. These scores are used in figuring out the knowledge of the student in every module. The *student model* will be explained in detail later in the next chapter.

multiple type	*classic type*	*select type*
<answer>	<answer>	<answer>
2	CPU	<1>
<1>	<true>	1
-0.5	1	<2>
<2>	<false>	-1
1	-1	<3>
<3>	<unanswered>	1
-1	-1	<4>
<4>	</answer>	-1
-1		<5>
<5>		-1
-1		<unanswered>
<unanswered>		-1
-1		</answer>
</answer>		

Table 5.3: Answer sections of three different question types

history/Module2

Remember that the sample test template in table 5.1 belonged to *Module3* of the sample course in figure 5.2. The first line right after the <4> tag is the *coefficient* of the fourth choice which is -1 in this example. Negative coefficient increases the negative score for *Module3* in *student model*. Notice here that negative score of *Module3* has been increased even though *Module3* is not mentioned anywhere in the *modification section*, this is because this question belongs to *Module3*. Likewise, a positive coefficient increases the positive score of the student in the *student model*. Negative *coefficients* are used in incorrect choice sections and in <unanswered> section. Positive coefficient is used in the correct choice section. Coefficients usually range between $-1.0 \leq coefficient \leq 1.0$. This is not a strict rule and it doesn't always have to be the case, for example, say -2.5 can be used as a coefficient for an incorrect choice, but $-1.0 \leq coefficient \leq 1.0$ range is the recommended range. After the coefficient line comes the *history/Module2*[4] line. Notice that in the sample course in figure 5.2, *Module1* and *Module2* are prerequisite modules of *Module3*. In other words, *Module1* and *Module2* come before *Module3* in the course flow, and a question in *Module3* may be relevant with these prerequisite modules.

[4] *history/Module2* is the full path for *Module2*, see the subdirectory ordering in the *Libraries* section of this chapter

Thus, the *history/Module2* line dictates that this response is also related with the information presented in the previous module which is *Module2*. Since the student couldn't answer this question correctly, this means he/she might have missed or forgotten some of the concepts in *Module2*. So for this sample *modification section*, negative values of *Module3* and *Module2* has been increased in the *student model*.

In general, the *modification section* starts with an *modification section* tag, e.g <4> which is followed by a *coefficient*, e.g. -1. The *coefficient* can be followed with prerequisite module names. So the student's answer to a question affects the positive or the negative score of the module to which it belongs, and the positive or the negative scores of the prerequisite modules. The differences between the *answer sections* of three question types available in WIZ can be seen in table 5.3.

5.1.5 Modules

A sample module template can be seen in table 5.4. The first line in the module template holds the name of that module. The second line holds the *concept description*[5] or the subject of that module. The rest of the module template consists of four sections, *beginner level*, *intermediate level*, *expert level* and *review sections*. The first three sections are used in student leveling and the fourth one is used by the *student model*. For every section <test> tag is optional; tests can either be used or not. In every section, after the <test> tag more than one test can be placed, but it is recommended not to have many tests in each section. Also it is not recommended to have one test which contains many questions. Ten to fifteen questions to be displayed at a time to the student is a reasonable number[6]. In course presentation, a module can be visited for two purposes; normal visit and review visit. Review visit to a module can only be decided by the *student model*. Review visit is performed if the system decides the student shall not continue the course before reviewing a module. Normal visits can occur either in the normal course flow,

[5] A module usually explains a specific concept to the student

[6] Notice here that only the matching questions with the student's level are displayed to the student

Course Template	*Module Template*
History Turkish History <modules> history/Module1 history/Module2 history/Module3 history/Module4 <EOT>	Module3 Turkish History 1923-1930 period <beginner> history/begin1.html history/begin2.html history/begin3.html <test> history/begin1.test history/begin2.test </beginner> <intermediate> history/inter1.html history/inter2.html <test> history/inter1.test </intermediate> <expert> history/expert1.html </expert> <review> history/review1.html <test> history/review1.test </review> <EOT>

Table 5.4: A sample course and module template

or by manually returning back to that module in the course presentation by the student. Whatever the reason for the visit, the student can see only one of the four sections. For example, if it is a normal visit, web pages (and tests) of the section that matches the student's level are displayed to the student. If it is a review visit, then web pages (and tests) of the review section are displayed.

As a result, a student has a level in the range of *beginner, intermediate* and *expert* in every module. When registering to a course (how to register to a course will be presented in the next chapter), the student is asked to select a level of presentation that best matches him. The student's level in every module is set to the selected level initially. According to the student's level, the length and depth of the presentation varies. For example, in a *beginner level* section of a module, there may be four web pages plus two test pages for explaining a concept, where as for an *intermediate level*, there may be two web pages plus one test page for explaining the same concept. As you have realized each section has its own web pages and test pages, and the number of web pages varies in every section. So two students with different levels see different web pages (and may be different number of web pages) in a module during course presentation.

In the *course preparation* section of this chapter it was said that student leveling is optional and does not have to be used. How this happens is as follows; copy the same web and test page names from the beginner section to intermediate and expert level sections. Then we have three sections which are identical. Thus, whatever the student's level is, every student sees the same web pages during course presentation. The student's level becomes to have no effect in the content of the course presentation.

5.1.6 Courses

A sample course template can be seen in table 5.4. The first line in the course template holds the name of that course. The second line holds the description or the full name of that course. The rest of the course template holds the sequence of module names which make up the course.

5.1.7 Subsystems

As can be seen from figure 5.1, there are two subsystems in WIZ; the *Course Registration Subsystem* and the *Course Presentation Subsystem.*

Course Registration Subsystem

The course creator, after preparing web pages, test templates, module templates and a course template for a particular course, has to register this course to the WIZ system by using this subsystem. This subsystem compiles a course and checks if all the files; web pages, test templates, module templates that belong to the course, exist in the system. Once all the files for the course are available, it is registered to the system by updating the *WIZ database.* The process of course registration to the WIZ database takes place as follows: The course creator runs the *Course Registration Program* and enters some information about the course plus some information about himself/herself, such as, email address etc. The program takes the course name, checks whether a course template with that name exists. If it exists, it opens that template file and reads the module template names; while reading the module template names, it checks whether such module template files exist in the module library. Also the web page names and test template names in every module are read and checked whether they exist in web page and test template libraries. If every file needed in the course presentation exists, the program registers the course to the *WIZ database*, otherwise it aborts with an error message that contains the names of the missing files. A course is recognized by the *Course Presentation Subsystem* only after it is registered to the *WIZ database.* So even if a course is ready, it can not be presented to students unless it is registered to the *WIZ database.*

Course Presentation Subsystem

This subsystem is the heart of the WIZ system. It provides the interface between the students and WIZ. This system consists of two CGI programs; the

first one, *register.cgi* allows students to register the system over the Internet and the other one, *WIZ.cgi* handles all the course presentation. The user interface facilities that this subsystem provides is presented in the next chapter.

Chapter 6

Course Presentation Subsystem

The *Course Presentation Subsystem* provides the interface between the WIZ system and the student. In this chapter, how to use the system, system facilities and the user interface will be presented.

6.1 Registration to the WIZ System

Only registered students can use the WIZ System. Therefore the registration process has to be accomplished in order to be a member of the system. Registered students have a unique user-ID and a password to log on to the system. The entrance page for the WIZ System is given in figure 6.1. As can be seen, this page includes a link to the registration page for non-members to follow. This registration page can be seen in figure 6.2. All the entry fields are mandatory to fill in this form. If the form is submitted with some empty entry fields then a page with the same entry fields is redisplayed (figure 6.3) and the user is requested to fill the empty entry fields. After the user correctly submits the registration form, a new page that requests a user-ID and password appears. By default user-ID is set to the surname of the user in small letters. User enters his/her password twice in this page (figure 6.4). When this form is submitted, there can be four possible replies, three of which are error messages. The first reply type is a page with an error message of 'password pairs conflict' if the

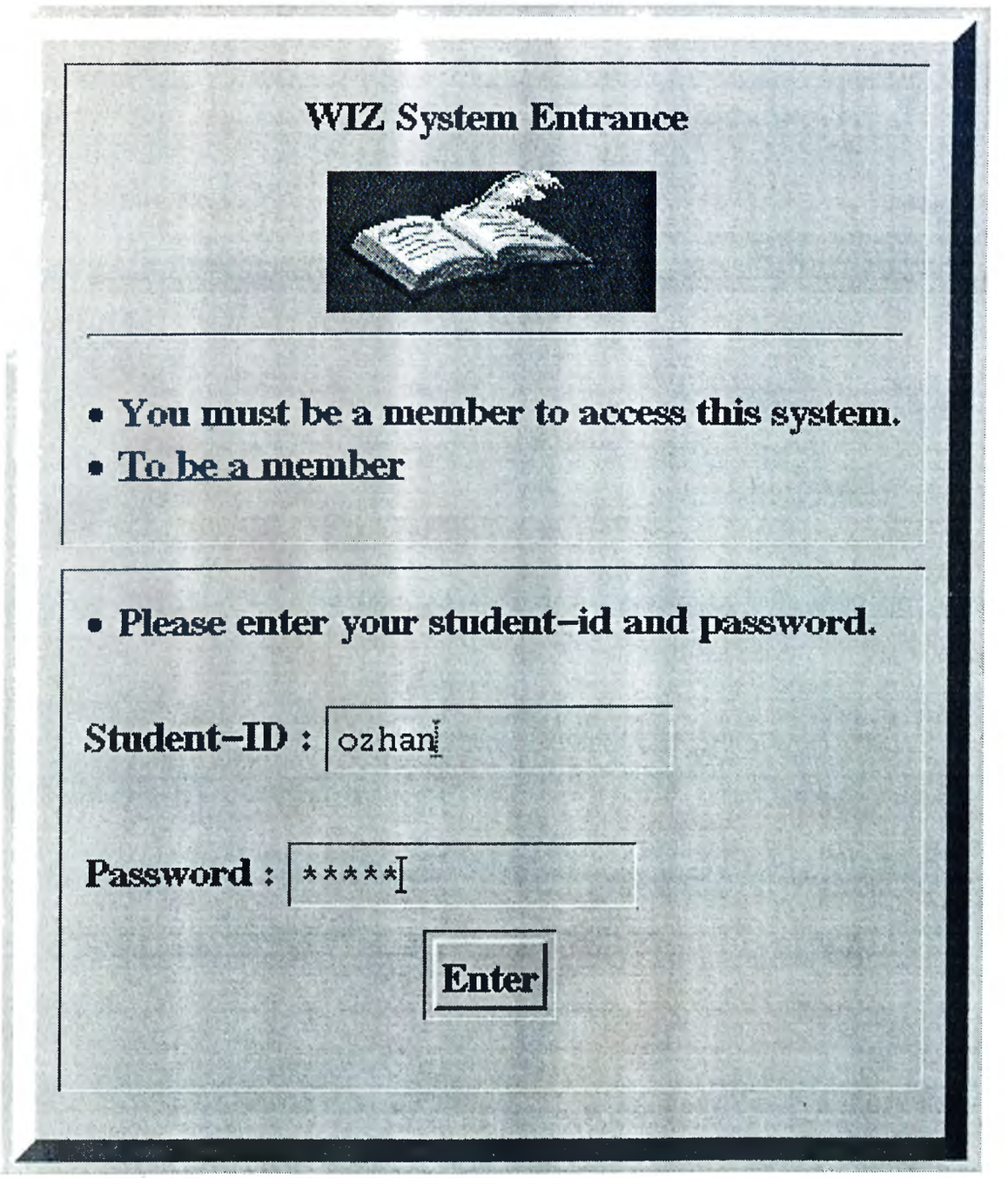


Figure 6.1: Initial page for connecting the WIZ System

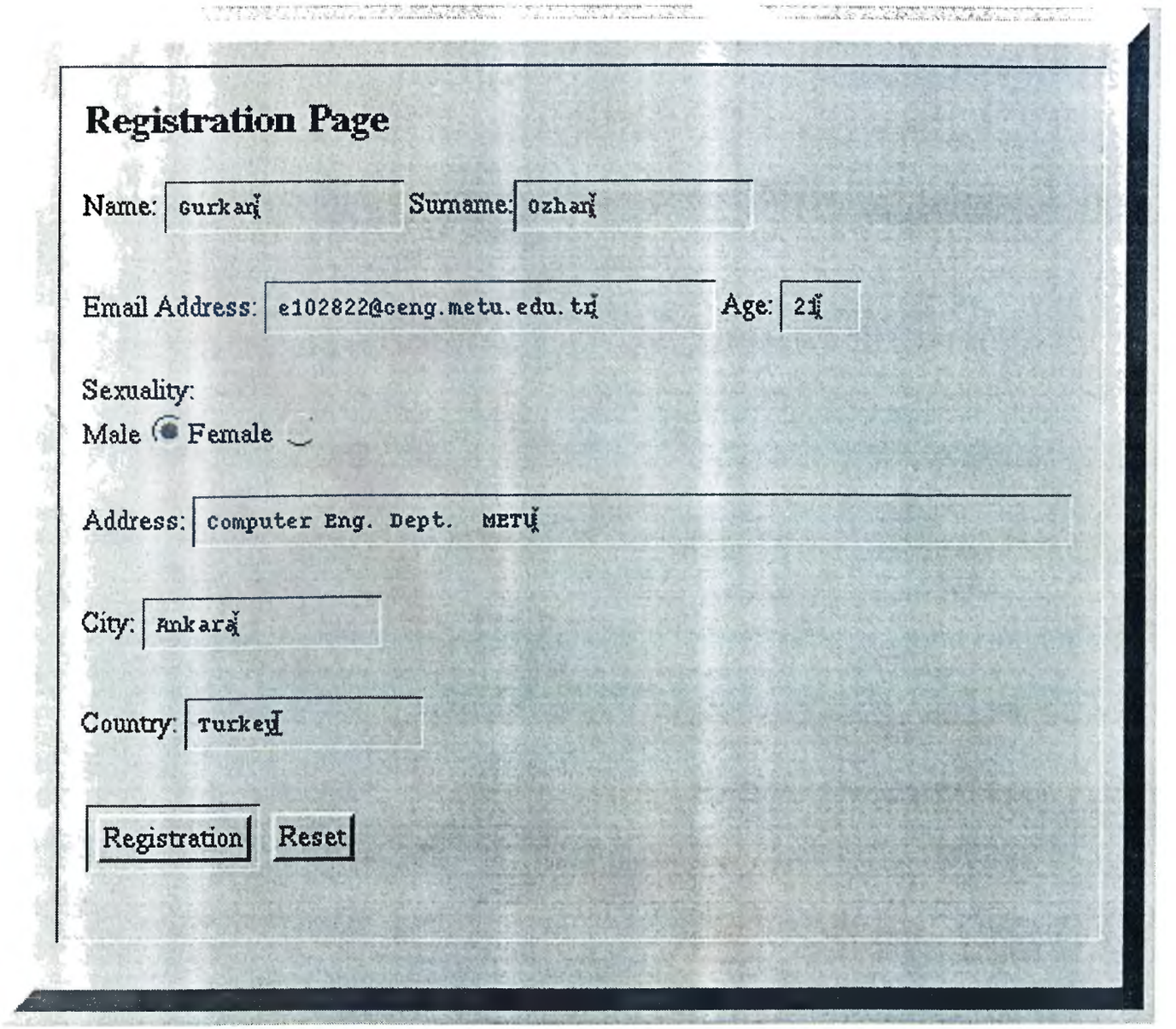


Figure 6.2: Student Registration Page

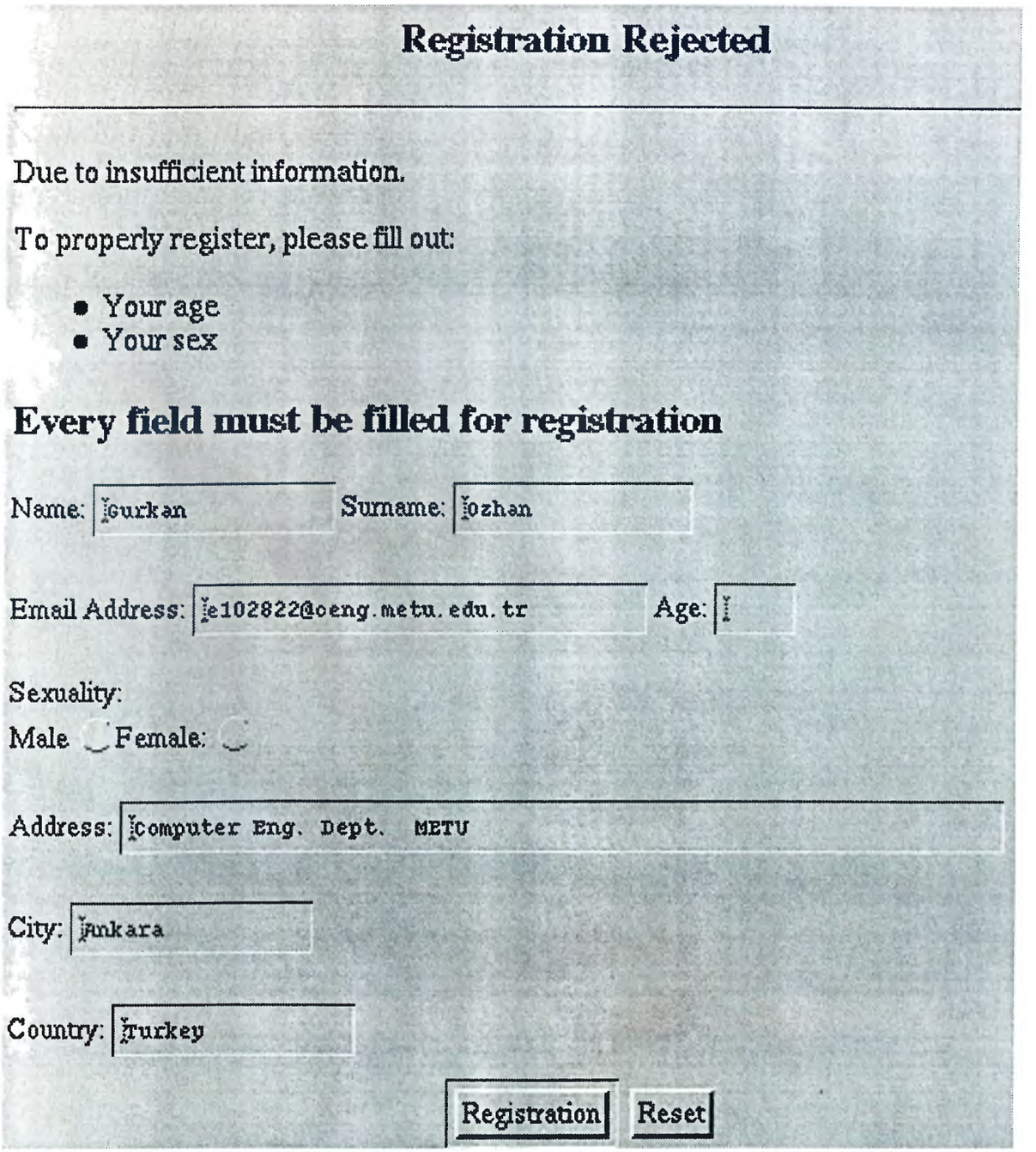


Figure 6.3: A response to registration page that is submitted with some empty entry fields

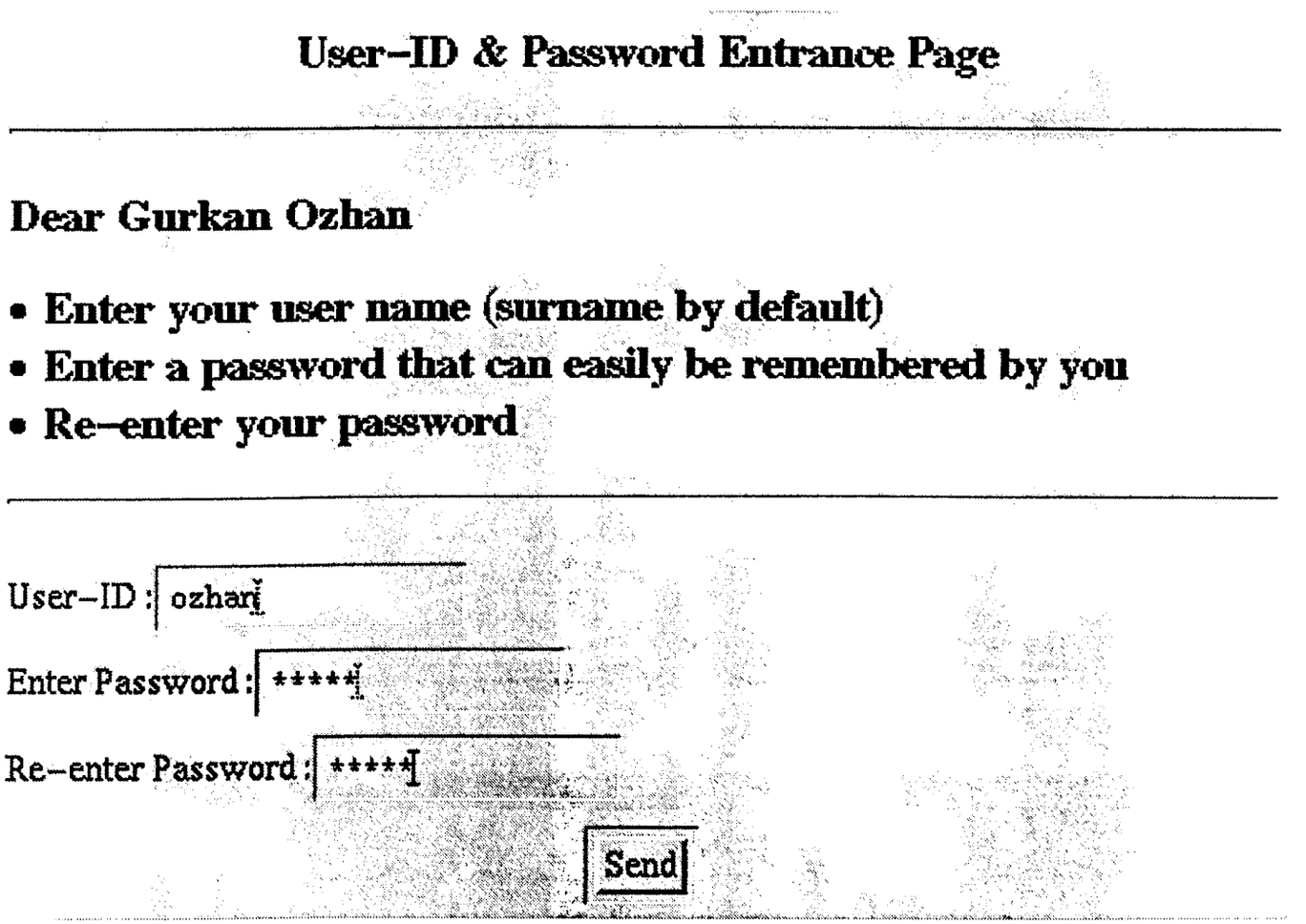


Figure 6.4: Final step in student registration; defining user-ID and password

password pairs are not the same. The second reply type is a page with an error message of 'this user-ID is already in use' if the entered user-ID is occupied by some other user. The third reply type is a page with an error message of 'empty entry fields' if one or more entry fields are left empty. If each case, the same form is redisplayed so that user can get rid of the error by doing some modification on the form. Finally, the fourth type of reply informs user that registration is completed.

The *Register.cgi* program is responsible for the registration process. This program handles the process of registering new users to the system and displaying the appropriate reply messages during registration. Once registration to the system is completed, the user can connect the WIZ system properly.

Any user can register to the WIZ system over the Internet. This access can be switched of so that only the system administrator/lecturer, can handle the student registration process. In this way the WIZ System can be turned into a commercial product.

6.2 WIZ Main Page

A user connects to the system by entering his/her user-ID and password (see figure 6.1). Once connected, the WIZ main page is displayed (see figure 6.5). At the center of the page, there is a pull down menu which is labeled *Choose a course below* and a course start button. At the bottom of the WIZ main page are four options; course registration, course unregistration, communication facilities and help options. When a student registers and connects to the WIZ system for the first time, the pull down menu in the first table contains no course names. This is because the student is not registered to any course. Thus, the first thing to do is to press the *Register to Courses* button to connect to the *Course Registration* page.

6.2.1 Registration to Courses

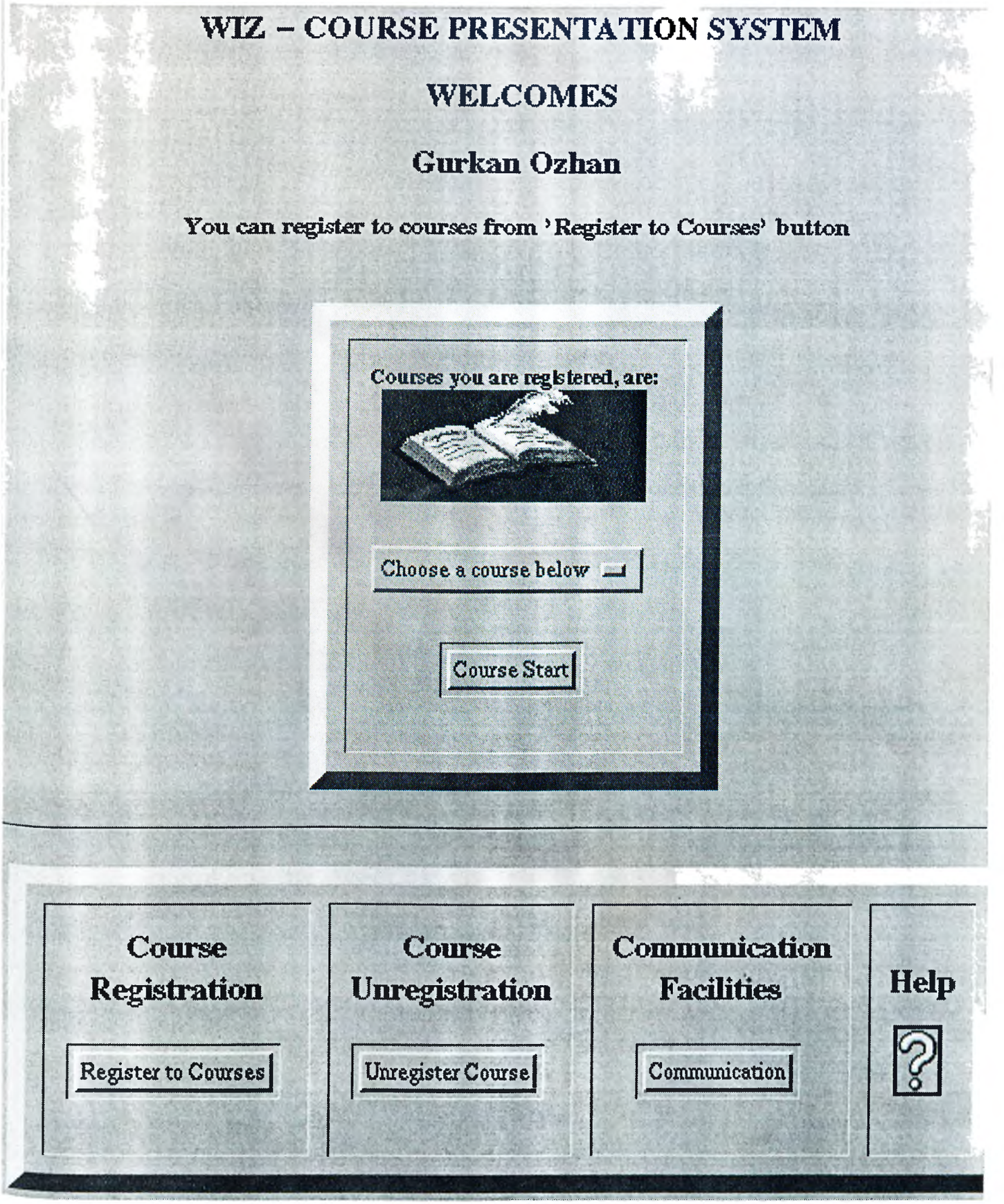


Figure 6.5: WIZ Main Page

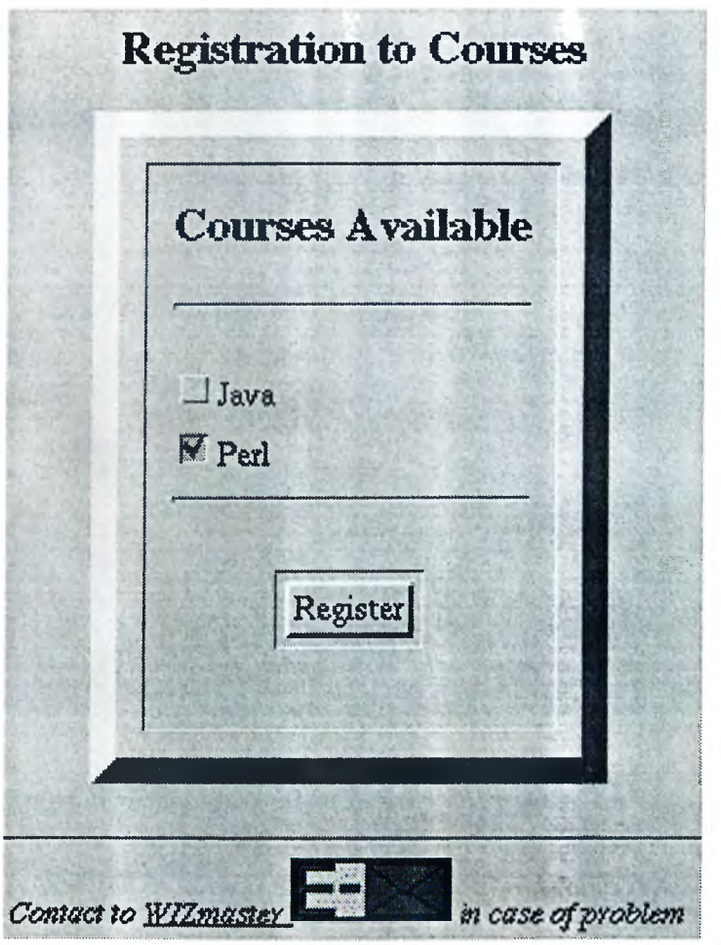


Figure 6.6: Course Registration Page

Figure 6.7: *Course Level Selection* Page

Figure 6.8: Results of Course Registration

The *Course Registration* page (figure 6.6) lists all the courses that are available in WIZ. Available courses are the ones that are registered to the WIZ database. There are two courses available in the example of figure 6.6. There is no upper limit to the number of courses in the system. Users can register to any course simply by clicking on the checkbox next to each course name. When the user presses the register button, he/she receives the *Course Level Selection* page which can be seen in figure 6.7. All the courses are set to *beginner* level by default. The course content may change according to level selected. While a student who does not have a background in the course should start at a beginner level, a student who knows the subject well enough may start at an expert level. When the student clicks on the *Set Level* button, *the student's level in every module of the course is set to the selected level*, and the *Registration Result* page is displayed (see in figure 6.8). The *Registration Result* page indicates whether the student successfully registered to the selected course(s) or not. If the student tries to register a previously registered course, he/she receives an error message indicating that the student is already registered to the course. From this page, the user returns back to the *WIZ main page*. After course registration, the user can see the course names that he/she is registered to in

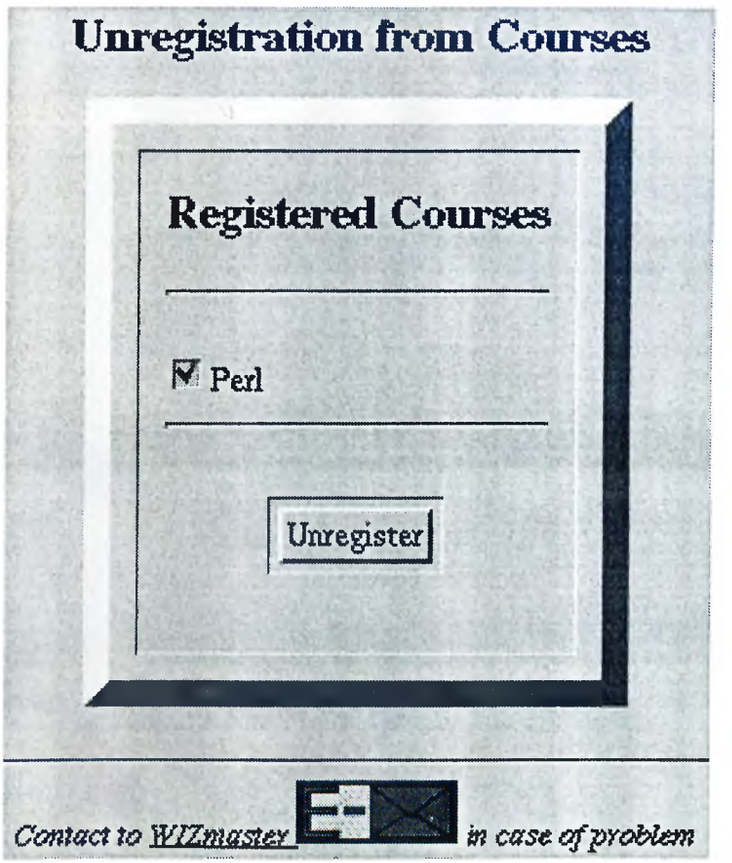


Figure 6.9: Course Unregistration Page

the pull down menu of the WIZ main page.

6.2.2 Unregistration from Courses

Course unregistration process is just the opposite of course registration process. The student can reach the *Course Unregistration* page (see figure 6.9) by clicking on the *Course Unregister* button in the *WIZ main page.* The *Course Unregistration* page lists all the courses that the student is registered. User can unregister from any course he/she wishes by clicking on the checkbox next to each course name. After pressing on the *Unregister* button, the student receives the *Unregistration Result* page (see figure 6.10) which displays the result of the unregistration process. All the records of the student in a particular course are deleted when the student unregisters from that course. Therefore,

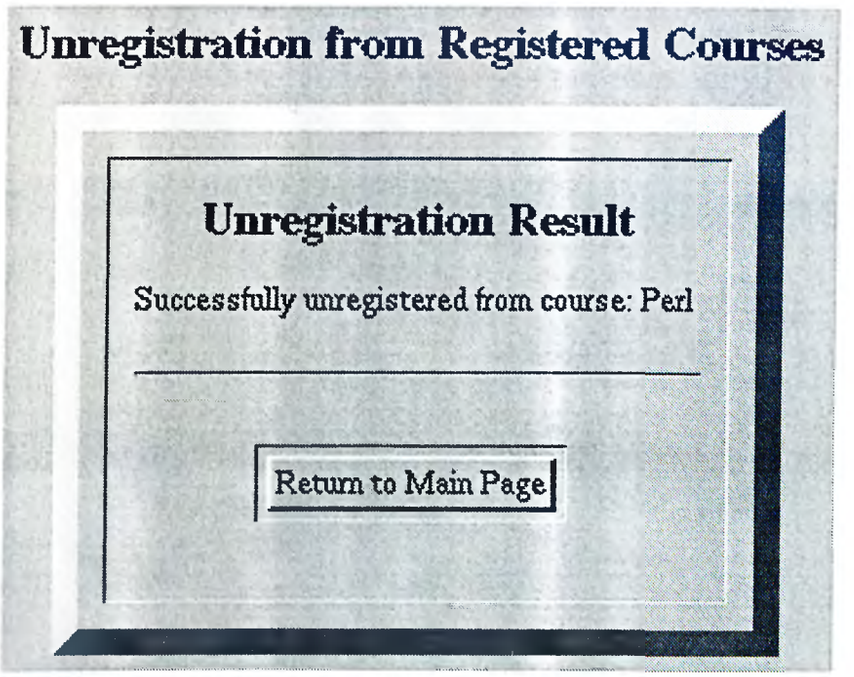


Figure 6.10: Results of Course Unregistration

unregistration from a course is not recommended unless the student will not use that course anymore.

6.2.3 Communication Facilities

The *Communication Facilities Page* is shown in figure 6.11. The WIZ system provides two types of communication; by newsgroups and email. Each course has a newsgroup, this is not mandatory but it is recommended. The course creator decides whether a course will have a newsgroup or not when registering a course to the WIZ database using the *Course Registration Subsystem.* The newsgroup address is entered during this course registration process. The communication facilities page contains three options. The first lists links to the newsgroups of registered courses. Pressing on one of these newsgroup links opens a new *news* browser connected to that particular newsgroup (see figure 6.14. Messages can be read, replied to and new messages can be sent. The threaded structure of newsgroups allow information to be exchanged in a structured way. Newsgroups allow discussions to be carried out between students

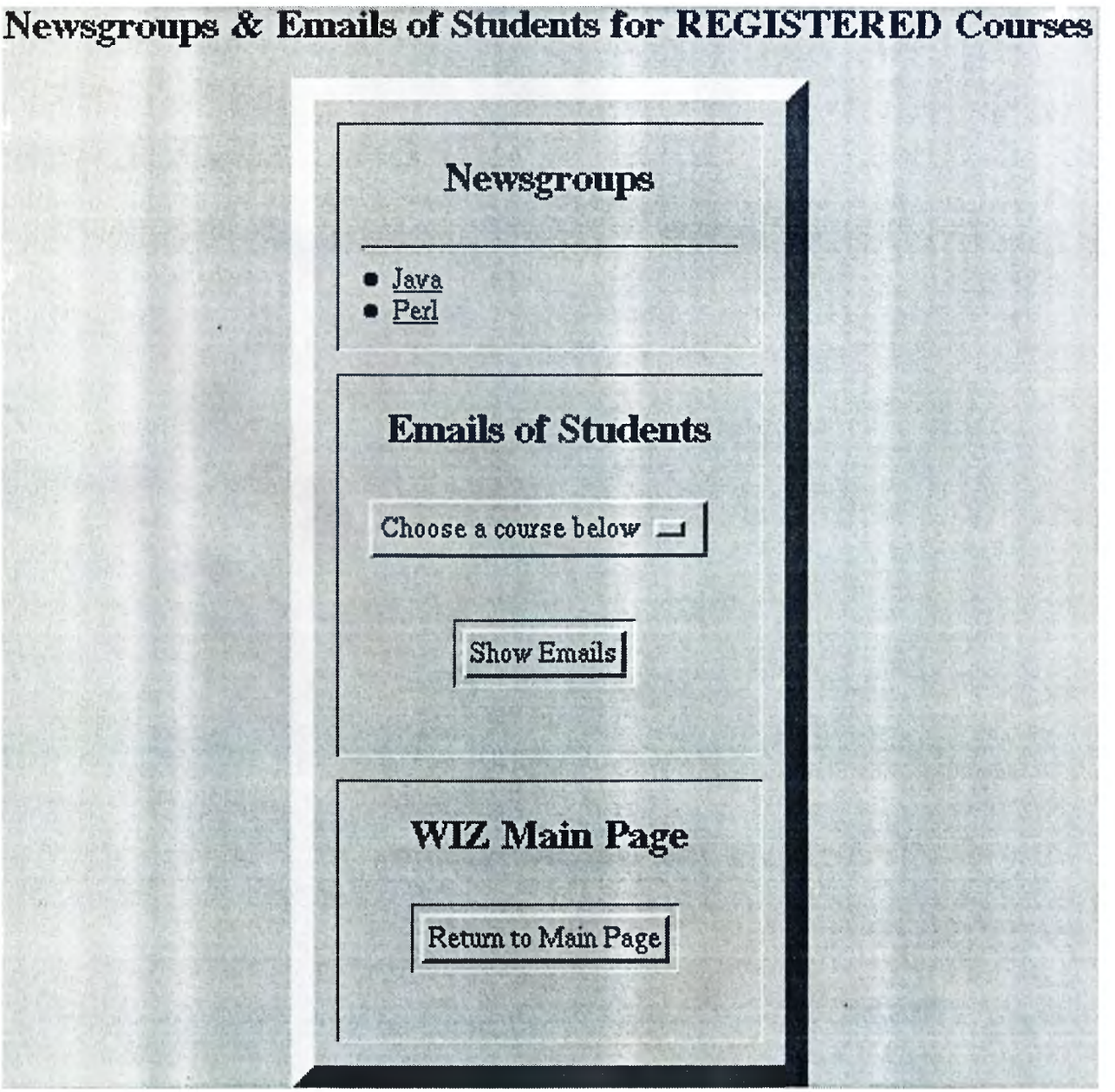


Figure 6.11: Communication Facilities in WIZ

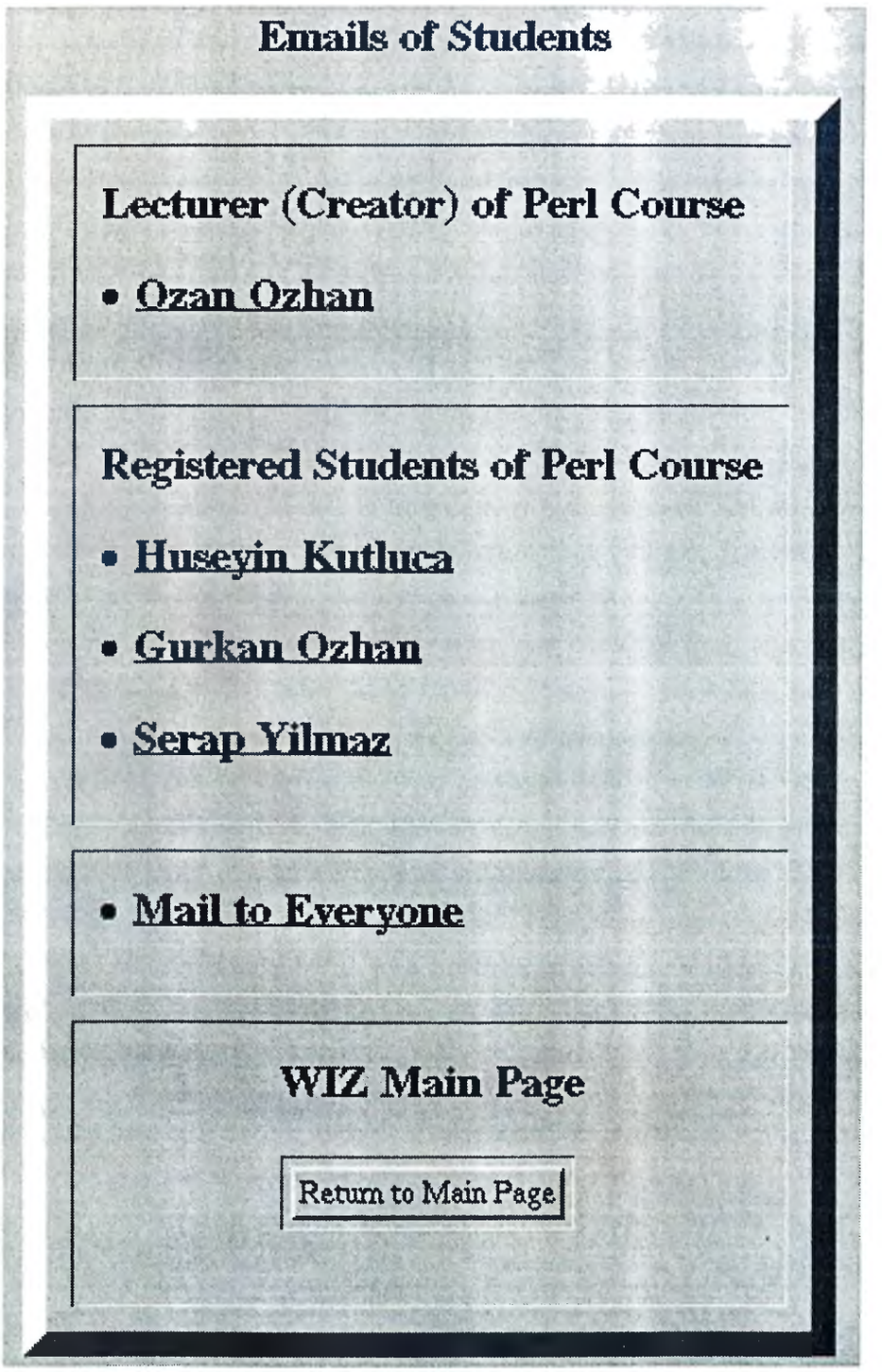


Figure 6.12: List of registered students with e-mail links in a sample course

although they are geographically and chronologically separated. Thus, newsgroups are very suitable for collaborative learning. The second option contains a pull down menu and a *Show Emails* button. The pull down menu contains the names of registered courses for a particular user. If the user selects a course and presses the show emails button, this connects him/her to the *Emails* page, and finally the third option contains a button which returns the user to the *WIZ main page*. The *Emails* page can be seen in figure 6.12. This page contains names, with email links of all students (and lecturer) of the selected course. The *emails* page contains a table that has four cells. The first cell contains the email address of the course creator, or the lecturer in other words. Students can directly send an email by pressing on the name of the course creator. In the second cell, there is a list of emails of students registered to that particular course. The third cell contains a link which when clicked on sends an email to everyone; lecturer and students taking the course. The main difference between newsgroup and email type communication is that newsgroup is a public communication medium where everyone can read everyone's message and reply to it. On the other hand email is generally used as a one-to-one private communication medium. Personal lecturer-student, student-student communications can be carried out by the email facility in WIZ. Other than personalized communication, WIZ allows to send an email to everyone taking the course, allowing one to all type of communication. Finally, the button in the fourth cell takes the user back to the *WIZ main page*. The WIZ system exploits the facilities of the Web browser in communication. Displaying newsgroups and sending email facilities are provided by the Web browser, not by the WIZ system. Newsgroup and email facilities are also available during course presentation.

6.3 Course Presentation

Course presentation starts after selecting a course name and pressing on the *Course Start* button in *WIZ main page*. A section from a course page can be seen in figure 6.13. The first line of the page is the title line. The title includes the subject (concept description) of the module, page number in the module

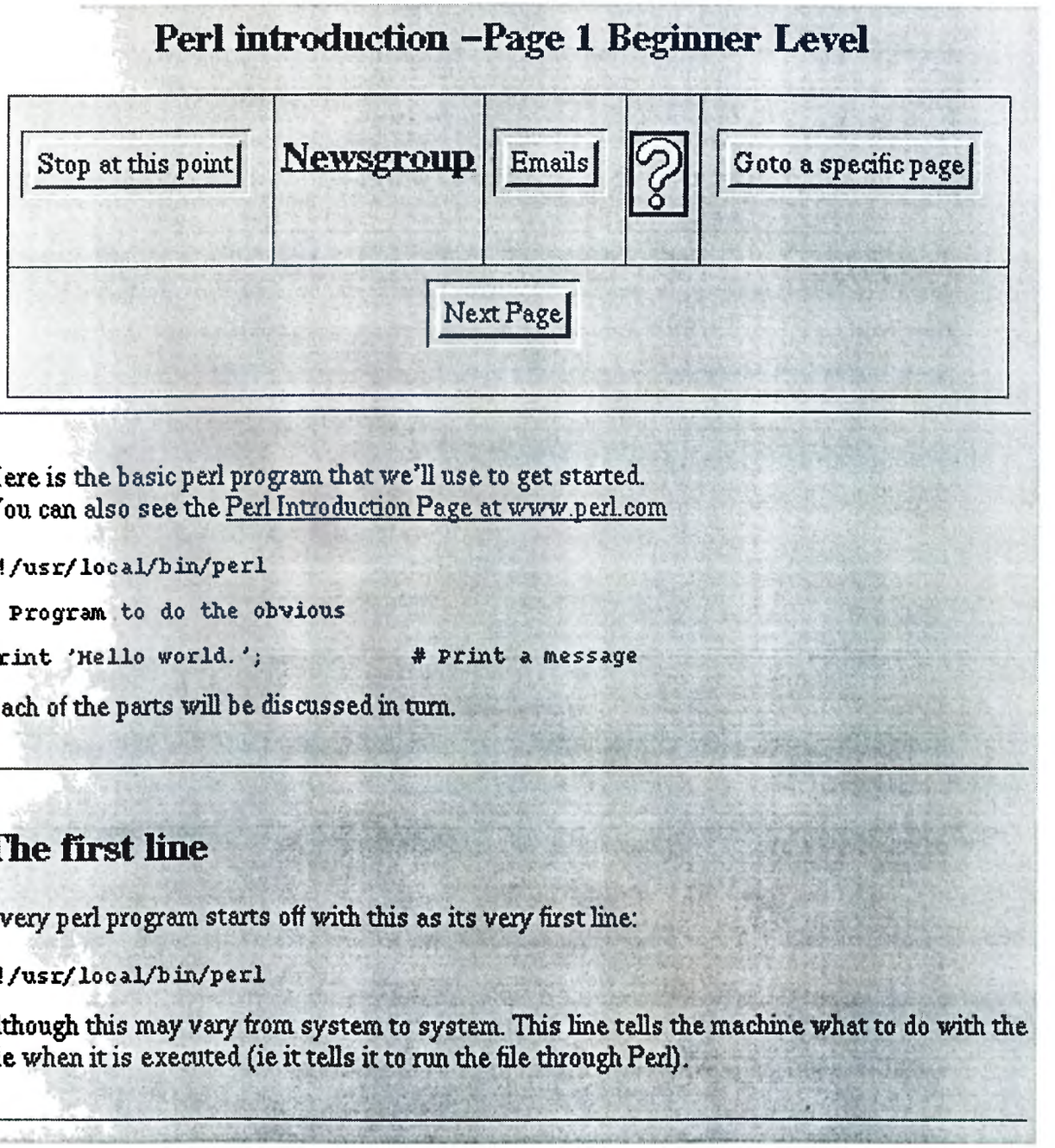


Figure 6.13: A web page in a course presentation

and the level of presentation to the student. Right after this title line comes a control panel. A similar control panel (it can not be seen in this sample figure 6.13) also exists at the end of each page. The web page that is displayed lays between these control panels. Having two control panels, one at the top and one at the bottom, provides quicker and easier access to the control panel buttons. The title line and the control panels are independent of the Web page that is presented. They are generated automatically from the WIZ database and the module template by the Course Presentation Subsystem.

In this sample web page in figure 6.13, there can be seen an *outside information link* written as *'Perl Introduction Page at www.perl.com'*. When clicked, the web browser opens a new window and displays that specified web page. Notice that this web page does not belong to the course and is called an *outside information resource.*

6.3.1 Control Panel

Each control panel consists of two rows. First row usually consists of five cells. The first cell contains the *Stop at this point* button, the second contains the *Newsgroup* link, the third contains the *Emails* button, the fourth contains the *help* link and the fifth one contains the *Goto a specific page* button. This row may contain six cells in certain cases. In such cases, the sixth cell contains the *Default Flow* button. The elements of the first row is explained below:

First Row Buttons

'Stop at this point' Button: When this button is pressed, the course presentation ends (bookmarking the current page) and the WIZ main page (figure 6.5) is displayed. If the student decides to start the same course after sometime, the course presentation starts from this same (bookmarked) page. It is advised that the student to press this button whenever he/she wants to leave the course presentation. This allows the student to continue the course flow from the point where he/she has left, when he/she later starts the course presentation again.

If the student leaves the course presentation without pressing on this button, when the student restarts the course presentation, he/she continues from the last bookmarked page (if no page was bookmarked previously, he/she starts from the first page of the course), *but* in the control panel, there exists the *Default Flow* button, (only in the case when the student leaves course presentation without using the *Stop at this point* button) which when pressed, takes the student to the latest point where he/she was in the default course presentation.

'Newsgroup' Link: This link connects the student to the newsgroup of the course on a new newsgroup browser window (see figure 6.14). Using this students can discuss issues about the course, cooperate on assignments and ask/answer questions with the other students taking the course.

'Emails' Button: This button connects the user to the *Emails* page which can be seen in figure 6.12. The *Emails* page has been explained previously in the *Communication Facilities* section of this chapter.

'Help' Link: This link opens a new web browser and connects to the help web pages. Help web pages give information about the usage of the WIZ system.

'Goto a specific page' Button: This button provides the freedom to go any web page of the course during course presentation. When pressed, this button connects the user to the page seen in figure 6.15. The pull down menu in this page is set to *WIZ main page* by default. The student may select any web page of any module using the pull down menu. Pressing on the *Take me there* button takes the student to the specified page. Using this facility, the student may change the course flow and continue the course presentation from the selected web page. The main aim for providing this facility is to give students the freedom to browse in the web pages of the course. Usually it is expected from the student to use this facility for returning back to the previously displayed pages in order to review previously displayed information.

As mentioned previously, there is a student leveling system in the WIZ system. A student has a level in the range of *beginner, intermediate* and *expert*

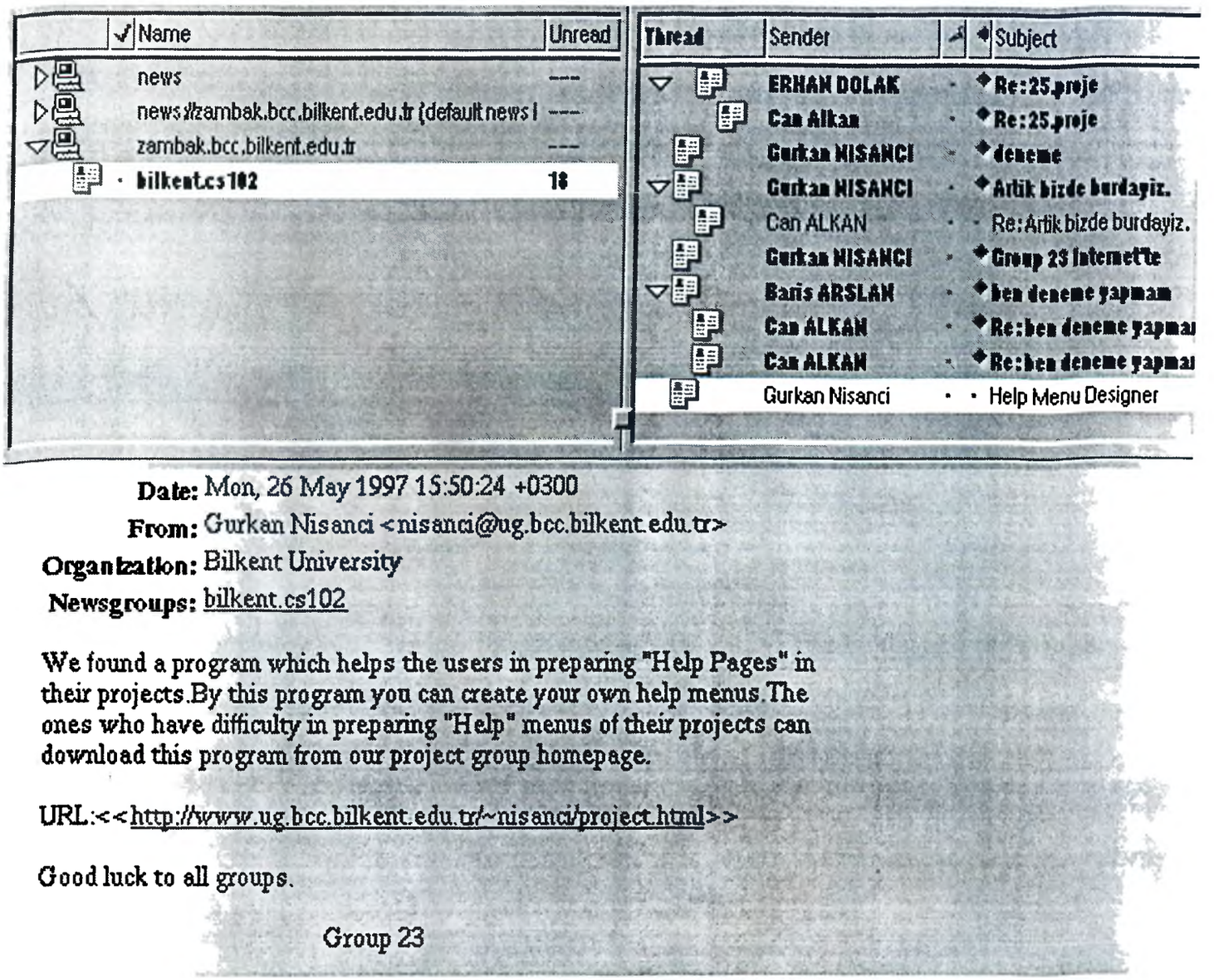


Figure 6.14: A sample newsgroup for CS102 (Algorithms & Programming II) course

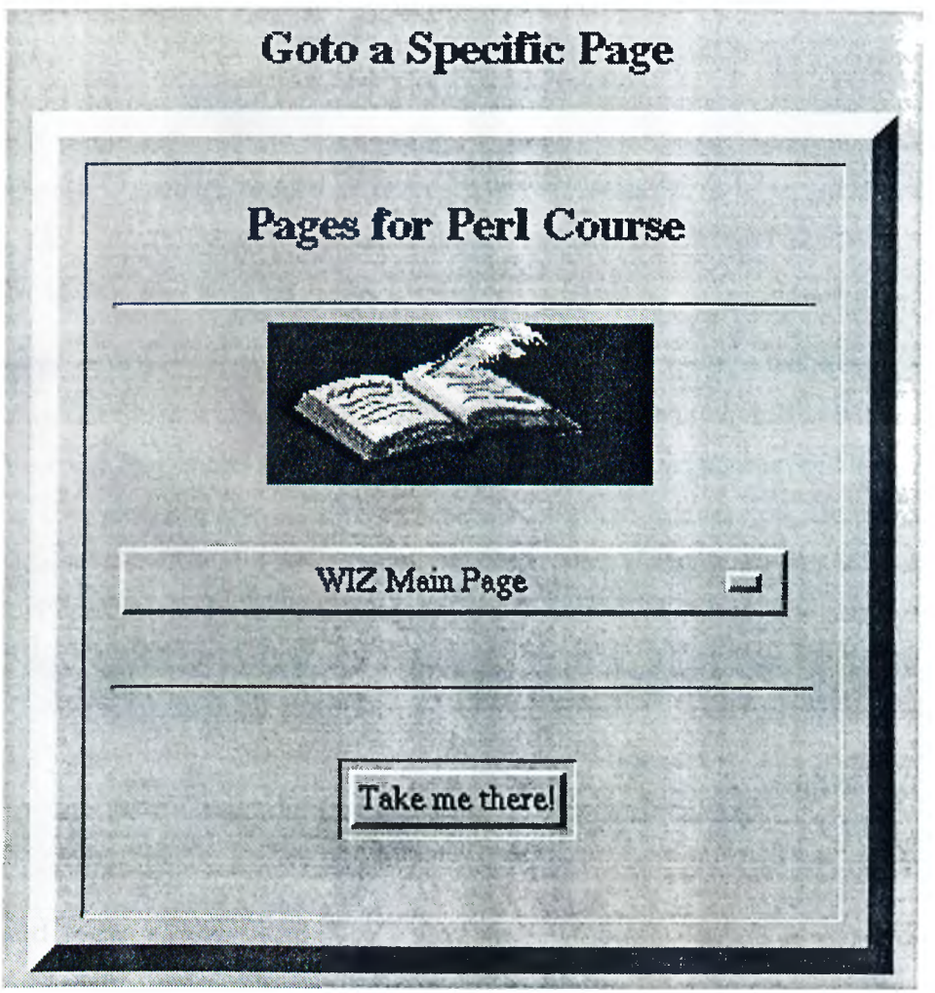


Figure 6.15: *Goto a specific page*

in every module. When registering to a course, the student is asked to select a level of presentation that matches him the best (see figure 6.7). The student's level for every module in the *student model* is set to the selected level initially. As mentioned in the *modules* section of chapter 5, the web pages (and even the number of web pages) to be displayed in a module varies according to the student's level in that module. During course presentation whenever a module is covered successfully, the student's level in that module is increased by one level by the *Course Presentation Subsystem*. So when a beginner level student covers a module, the *Course Presentation Subsystem* increases the level of the student in the covered module (to intermediate) modifying the *student model*. So, when the student wants to return back to the previously covered module using the *Goto a specific page* button, he/she can only access the web pages for the (new) intermediate level.

'Default Flow' Button: The student may change the course flow manually by using the *Goto a specific page* facility. The student may continue the course presentation from here using the *Next Page* button (the button at the second row of the control panel). The *Default flow* button appears as the sixth cell only when the student goes to a specific page and changes the default course flow. Pressing on this button returns the student back to the point (web page) where he/she has left the default course flow, which is the point where the student has changed the flow of the course presentation using the *Goto a specific page* facility. When the student returns back to the default flow, this button disappears from the control panel.

Second Row

'Next Page' Button: This button is the most used one and thus is separated from the other buttons in the control panel. The user continues course presentation by pressing on this button. When pressed, the *Course Presentation Subsystem* figures out the web page that the web browser will display, usually the next web page in the module. Web pages are followed by test templates that contain questions about the material that is presented.

There can be four types of possible replies when pressed on the *Next Page*

Figure 6.16: The *End of course* page

button. As a first type of reply the next web page in the module can be displayed. A second type of reply can be a test template, therefore the test template is processed by the *Course Presentation Subsystem* and a test web page can be displayed. As mentioned before, tests are optional. The currently displayed web page can be the last page of a module that does not contain any test templates, therefore the first web page of the next module in course sequence can be displayed as a third type of reply. And finally, if the last page of the last module is being displayed then pressing on the *Next Page* button ends the course, displaying the *end of course* web page (figure 6.16).

6.3.2 Test Web Pages

Usually teaching material is followed by test templates in modules. Test templates consist of different levels and types of questions together with their answers and student model modification information (see *Test Templates* section of chapter 5). A test web page is tailored by the *Course Presentation Subsystem* according to the student's level in that module. A sample test web page can be seen in figure 6.17. It is important to note here, that although there is no lower and upper limit to the number of questions, a test web page usually contains between five to fifteen questions. There are some differences between the control panel of a course web page and a test web page. Other than the previously explained buttons and links, there are three new buttons; *Skip the test* button, *Done with the test* button and the *Reset* button.

The student does not have to answer the test that is being displayed. *Skip the test* button provides the flexibility to skip the displayed test and go on with the next web page. Skipping a test does not have any positive or negative effect on the scores of the student in the student model. The *Skip the test* button can be used in cases where the student feels he/she knows the subject (concept) presented in the module and does not want to waste time answering the questions.

After answering the questions in a test, the student has to press the *Done with the test* button. After pressing on this button, the student's answers are evaluated, the positive and negative scores of the related modules are updated

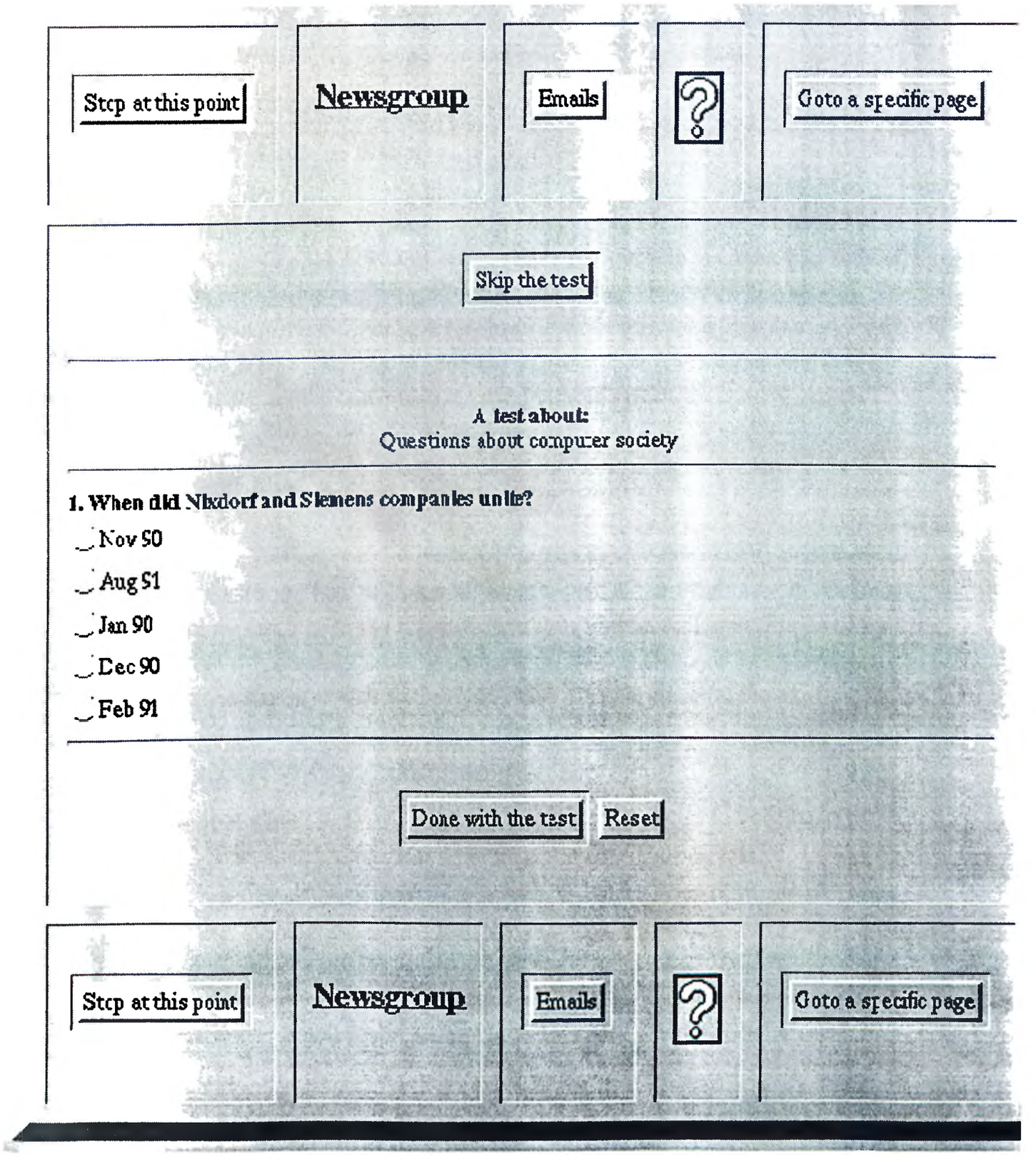


Figure 6.17: A sample test web page with one multiple choice question

in the student model, and the next test page or the next web page (the first web page of the next module) is displayed. Finally the *Clear* button clears the answers entered to the questions and initializes the test again.

6.3.3 Course Flow

When designing the WIZ system, the objective was to produce a system which would provide the flexibility and freedom to browse in the course as well as providing the student the guidance to go through the course. To this end, the WIZ system has different types of course flow during course presentation which are the *default flow*, *manual flow* and *dynamic flow*. The *manual* and *dynamic* flow types are the ones which deviate from the *default flow*.

Default Flow

A course is a sequence of modules. In the default course flow, presentation starts with the first web page of the first module, and continues in sequence until the last page of the last module in the course is displayed. The user views all the web pages by pressing on the *Next Page* button and by answering the test pages (well enough so that the system does not decide to deviate from the default course flow) or by skipping them.

Manual Flow

Manual flow gives the freedom to browse in the web pages of the course. Manual flow begins when the student selects the *Goto a specific page* button. The student may change the course flow at any time and can continue the presentation from the web page he/she has selected. He/she may later may return back to the *default course flow* by pressing on the *Default Flow* button.

Dynamic Flow

This is the kind of flow where the *Course Presentation Subsystem* decides that the student is not ready to go further in the course presentation without first reviewing a module. After evaluating the answers' of a student in a test, the *Course Presentation System* checks data stored in the *student model* to decide on whether that student needs to review a module or not. The *student model* is a table in the WIZ database (which will be explained in the next chapter) which stores the students' progress and statistics in every module. The decision of whether the student should review a module is determined by looking at the ratio of the negative score over the positive score of the student for every module in the *student model*. If this ratio exceeds a certain threshold value for a module, than that module should be reviewed by the student. It is important to note here that every module has a review section that contain web pages specially created for reviewing the concept in presented in a module. In other words, when the student reviews a previously displayed module, he/she does not have to see the previously displayed web pages again, but the web pages in the review section.

After answering a test web page, if the *Course Presentation Subsystem* decides the student needs to review a module, the web page in figure 6.18 is displayed to the student. In this figure there are two buttons; the *Skip Review* button which provides the flexibility to skip review and continue with the default flow if the student wants to, and the *Continue* button which takes the student to the first review web page of the module that will be reviewed. After reviewing a module, the course presentation returns back to the point (web page) where the review was started and continues with the default flow.

Reviewing Modules

The title line and the control panel of a review page can be seen in figure 6.19. The word *'REVIEW'* is attached automatically in the title line of a review web page by the *Course Presentation Subsystem*. Test templates can be attached also in the review section of a module. When the reviewing ends,

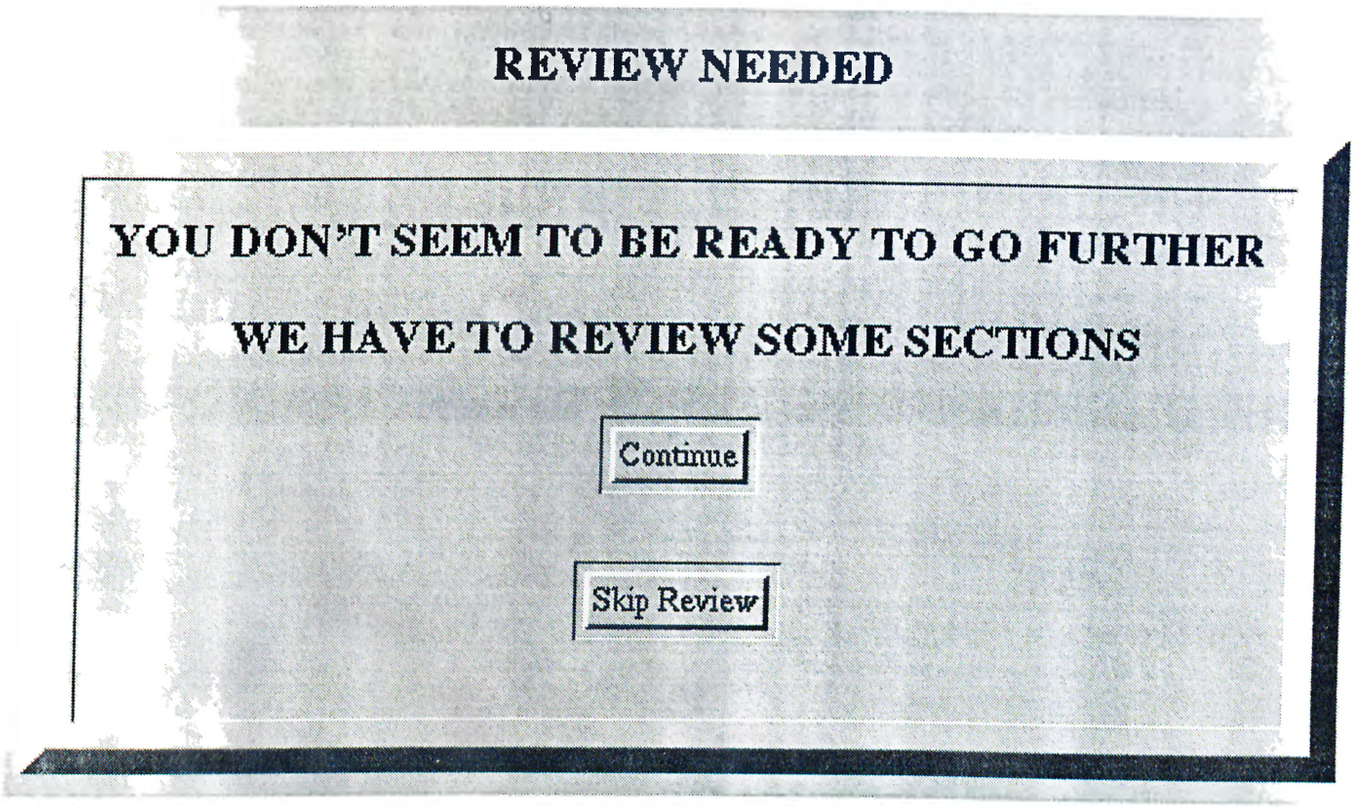


Figure 6.18: The *Course Presentation Subsystem* decides that the student needs to review a module

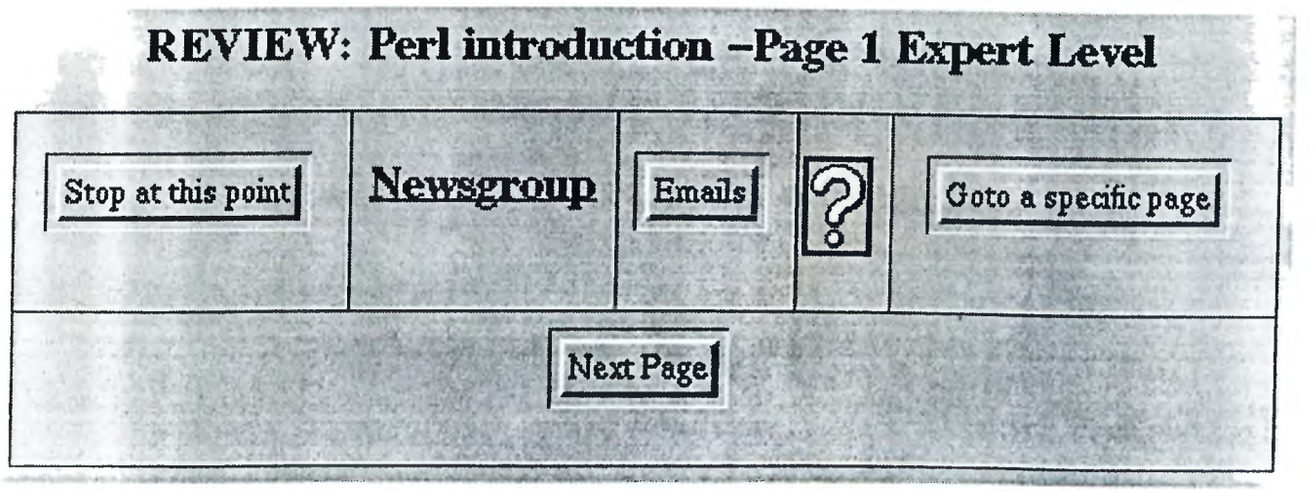


Figure 6.19: A title line and the control panel of a review page

the course presentation continues from the web page in the default flow before the reviewing was started. When a module reviewing is about to start, the negative and positive scores of the student in that module are initialized in the *student model*. In very rare cases during the reviewing of a module, the student may not answer well enough in the test(s) at the end of the review section, and the initialized negative score of the student may increase again causing the to student to review the same module again until he/she passes or skips the test(s) at the end of the review section.

Chapter 7

WIZ Database System

A database system is used to store all the information about students, courses, modules, students' progress in courses and the student model. In this chapter, the database structure and the student model will be presented. The DBMS used in WIZ is Mini SQL (mSQL) [Hug95], [Man96].

7.1 Mini SQL DBMS

Mini SQL, or mSQL, is a lightweight database engine designed to provide fast access to stored data with low memory requirements. As its name implies, mSQL offers a subset of SQL as its query interface. Although it only supports a subset of SQL, everything it provides is in accordance with the ANSI SQL specification. The mSQL DBMS package includes the database engine, a terminal 'monitor' program, a database administration program, a schema viewer and a C language API. The API and the database engine have been designed to work in a client/server environment over a TCP/IP network.

7.1.1 The Database Engine

The mSQL database engine, or daemon, is a standalone application that runs continuously as a background process listening for connections on a known

TCP/IP socket. The daemon can accept multiple connections and serialize the queries received. It utilizes memory mapped I/O and cache techniques to offer rapid access to the data stored in the database. Tests performed by a regular user of mSQL has shown that for simple queries, the performance of mSQL is comparable to or better than other freely available database packages. For example, on a set of sample queries including simple inserts, updates and selects, mSQL performs roughly 4 times faster than University Ingres and over 20 times faster than Postgres on an Intel 486 class machine running Linux [Hug95].

The Mini SQL Database Server supports multiple simultaneous user sessions. It handles multiple sessions without the need for multiple processes by queuing client requests and processing them in order. This design offers two advantages and one disadvantage as listed below:

- Advantages
  - No extra memory overhead regardless of the number of clients
  - No performance degradation due to lock handling between server processes
- Disadvantages
  - If multiple queries are received simultaneously there is a processing delay as the queries are queued.

As a result, mSQL allows multiple students to have the same course presentation concurrently, but as the number of operations on the database increases, the processing delay of the queries increases. Just to give an idea of how fast the queries are performed in mSQL, a simple test suite that is provided with mSQL using a 500 row test table produced the following results on a SPARCstation IPC running SunOS 4.1.3:

- 122 keyed inserts per second
- 98 keyed selects per second

- 172 non-keyed inserts per second
- 74 non-keyed selects per second

The main reasons for choosing mSQL are; it has a lightweight database engine, it provides fast access to stored data with low memory requirements, allows multiple simultaneous database operations, operations can be performed by a CGI program over the Internet, it has a C language API and finally it is free to download for scientific purposes. Fast access to stored data is very important since we don't want a student to wait for seconds in the course presentation because of the latency of database operations.

7.1.2 Mini SQL Specification

The mSQL language offers a significant subset of the features provided by ANSI SQL. It allows a program or user to store, manipulate and retrieve data in table structures. It does not support relational capabilities such as table joins, views or nested queries. Although it does not support all the relational operations defined in the ANSI specification, it does provide the capability of "joins" between multiple tables. It is thus sufficient for the database operations needed for the WIZ system.

7.2 WIZ Database Structure

There are five tables in the WIZ database system. The WIZ database system holds various information about the students, courses and the modules in the system. The table definitions and their explanations are given in appendix C. Briefly, the information that each table stores is as follows:

The Student Table: This table holds the personal student information of all the students registered in the system, including the user id, password, name, surname, email address, age, sex, address, city and the country of the student. When registering to the system over the Internet, the user is requested to fill

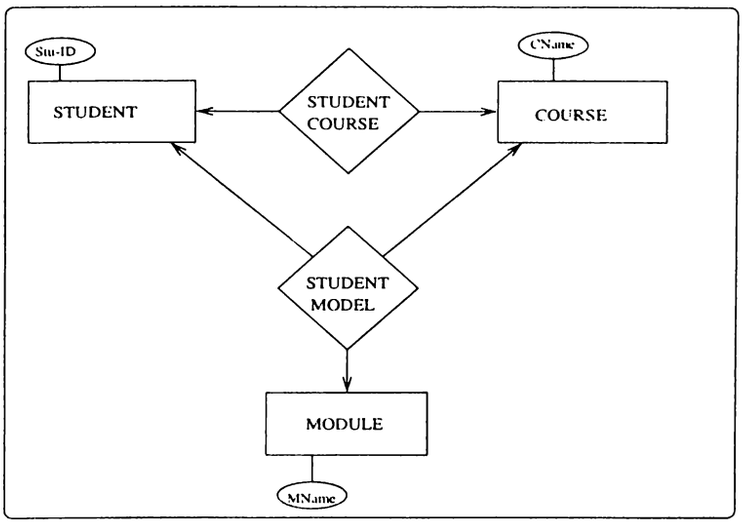


Figure 7.1: Entity-Relationship Diagram of WIZ Database

out a form with this information. Then, with this information, a new record is inserted into student table by the *Register.cgi* program.

The Course Table: This table holds information about the courses available in the system. This information includes the course name, course description, name, surname, e-mail address of the course creator and newsgroup address of the course. When a course is registered to the system, a new record is inserted into the *course table.*

The Module Table: This table holds information about the modules of the courses available in the system. This information includes the module name (this is the module template name), module description and name & surname of the lecturer (course creator). During course registration, new records are inserted into this table, each record corresponding to a module of the course. The *Course Registration Subsystem* automatically inserts these records whenever the lecturer (course creator) registers a new course to the system.

The Student-Course Table: A record in *Student-Course (stucour)* table holds information about a student's progress in a course. This information includes the number of times the user has started a course by clicking on the *Course Start* button of the WIZ main page (see figure 6.5), the page where the student has bookmarked and ended the course presentation, the page where the

student has left a course without bookmarking that page and the information of whether the student's email address is private or not (will be listed the class communication list or not).

The Student Model Table: Each record of the *Student Model* table holds a student's statistics and scores in a module of a course. This information includes the number of answered and unanswered questions, the number of correct answers to questions, the number of skipped and unskipped tests, the level of the student in the module, the number of times a module reviewed, the number of times module reviewing is skipped and finally the negative and positive score of the student in the module.

The *Course Presentation Subsystem* periodically checks the negative and positive scores of a student from the Student Model table during his/her course presentation to figure out whether he/she needs to review any module.

7.3 Database Operations

Database operations are performed by the subsystems in WIZ (see figure 5.1). The *Course Registration Subsystem* is executed by the lecturer locally, and the *Course Presentation Subsystem* is executed by the student from a distant location. Let's examine what happens when a student wants to register to the WIZ system.

Course Registration

Briefly, the client (the student at a distance location), using a web browser connects the web page seen in figure 6.2, fills up the registration form and submits the information. The web server receives the submitted information, fires the registration CGI program of the *Course Presentation Subsystem* and passes the submitted information to the program (see figure 4.3). The registration program processes the received information, connects to the database server and inserts a new record in the *student* table of the WIZ database. As can be

seen the client (the student) only sends information to a CGI programs at the server side and the CGI program performs the necessary database operations. A sample C language code for inserting a new record into the *student* table of the WIZ database can be seen in appendix D.

There is no certain limit on the number of courses that the WIZ system can present. There is also no specific limit on the number of students that can be registered to a course, but it is important to note here that as the number of students working concurrently increases, the response time to the students increases due to the latency of the database operations.

Course Presentation

During course presentation database operations are performed mainly on the *student-course* and the *student model* tables. In general, the next page that will be displayed during the course presentation is determined from the WIZ database. The current page information is stored in the *student-course* table. When the student presses on the *Next Page* button, using the current page information, the following web page or test page in the module is displayed, and the fields which store the current page information are updated in the *student-course* table. If the currently displayed page is a test page and the student presses on the *Done with the test* button, then the related fields of several records in the *student model* table (including the score fields of several records[1]) are updated according to the answers of the student (Refer to appendix C for detailed information about the fields of the tables in the WIZ database system).

Student Model Working Mechanism

After the student's answers are evaluated, the score fields of related records are updated in the *student model* table and the scores of the student in every module in the course are examined to see if the student needs to review a module. A module should be reviewed if the ratio of the negative score over the positive score exceeds a certain threshold value for a module. This threshold

[1] A record in the *student model* table maps to a student's scores and statistics in a module

value can be arranged and it is currently set to a value of '1.0'. In other words, if the student's negative score is higher than his/her positive score in a module, then that module is determined to be reviewed. There can be several modules where this ratio can exceed the threshold. In such a case, the *Course Presentation Subsystem* begins reviewing from the module with the highest ratio first. When the student starts to review a module, his negative and positive scores for that module are initialized to zero in the *student model* table.

Chapter 8

Summary and Future Work

This thesis presents the design and the implementation of a web-based hypermedia system which exploits the advantages of the Internet for distance education. The WIZ system presents web-based courses to geographically and chronologically dispersed students. The courses presented are interactive in the sense that tests are given to the students and the feedback from them is used by the system.

WIZ employs a student leveling system. The depth of course presentation and questioning varies according to the student's ability/knowledge. A student registers/begins a course from the level that best matches him/her. According to the student's answers given to the tests, the system varies the course content; adapting its presentation to the individual.

Students have the freedom to browse the during course presentation. There is also provided a button (the *Default Flow* button) that takes the student back to the guided presentation. The student can bookmark a specific page so that the presentation begins from the bookmarked page when the same course is restarted. The *Default Flow* button can be used here again in order to return back from the bookmarked page to the last page in the guided presentation.

The WIZ system provides communication facilities during course presentation. Each course has a newsgroup and the student can connect to the newsgroup of the course send/read/reply to messages and later return back to

the course presentation. Likewise, he/she can send an email to the lecturer, to any individual student taking the course or to everybody associated with the course (lecturer and students).

The WIZ database stores information about the students, courses, modules and the student's progress in the courses. The stored information can be queried by the lecturer/ administrator in order to obtain specific information about the students and/or the courses.

Finally, the WIZ system allows courses/modules to be modified and therefore be maintained for years. A web page can be used in more than one course satisfying the reusability criteria. Also the WIZ system allows web pages to have links to external information resources. This provides students to learn not only from the web pages inside the WIZ system but also from external resources.

As for the possible future extensions, we are planning to use the WIZ system to present course(s) to freshman computer science students. The feedback we'll get from them will be used in improving the system.

For the lecturer, a tool for simplifying the task of course/module creation and maintenance facilities can be implemented. As mentioned before modules and courses have special sections separated with special tags. Such a tool can be helpful to the lecturer in getting rid of details in the creation and maintenance of courses and modules.

Test pages are used to provide interaction with students in a course. The course creator prepares test templates that contain different types and levels of questions including the answers to the questions. In other words, each question type has a special format in the test template, and the course presentation system translates these test templates into test web (HTML) pages during course presentation. While preparing test templates, the course creator can make mistakes in question formats. A tool for compiling test templates can be prepared for avoiding syntax and question format errors.

Tests have a formative aim in the WIZ system. They are used in discovering and correcting the student's weaknesses through appropriate feedback in

response to his/her answers. This testing point of view can be extended with *evaluative tests* which have the intent of finding out how successful the student was in actually learning the course material. An evaluative test can be thought of as a final exam where only the score of the student is important. Another thing is that currently no feedback is displayed back to students about the result of a test. It might be better to give a feedback to the student about his mistakes in the tests given during course presentation.

The *student model* decision mechanism, the mechanism which decides whether the student needs to review a certain module in the course, can also be improved. Currently this mechanism uses only the ratio of the negative score over the positive score of the student in a module in deciding whether the student needs to review that module or not. If this ratio exceeds a certain limit for a module, the system decides to review that module. So, the default course flow changes when only the student is not successful. The decision mechanism can be improved such that when this ratio goes below a certain value, when the student is very successful, the system can decide to skip some modules in the course flow. The *student model* not only stores the scores of the student in every module but also other statistical information about the student. The decision mechanism can be improved so that the other information can be used in determining whether a module has to be reviewed or not.

A tool for providing administrative facilities can be implemented. In fact, the MiniSQL DBMS provides such a tool that allows the lecturer/administrator to interactively submit queries to the database engine and perform operations on the WIZ database.

Finally, the user interface can be improved in order to provide a more attractive environment to the student.

Appendix A

Form Components

HTML forms provide interaction on the Internet. The HTML structure of a form is presented in the this appendix. Chapter 4 presents HTML forms. A sample HTML code for a form is given in table 4.1 and it's appearance in a Web browser can be seen in figure 4.1. Refer to these figures if needed while reading this appendix.

Let's examine the HTML structure of a from from top to down. Every form begins with a header line similar to the one below:

<FORM METHOD=POST ACTION="URL/cgi-bin/WIZ.cgi>

Every form starts with a declaration line like the one above. This line represents a form whose content will be sent to a CGI program called the *WIZ.cgi* at the specified *URL*. The Web server will fire *WIZ.cgi* program when this form is submitted. The CGI program is stored in the *cgi-bin* directory of the server. CGI programs are usually stored in a special directory, *cgi-bin* in this example, for security reasons at the server side. Method type can be *get* or *post*. It is *post* in this example. The difference between the two method is in the way how the CGI program handles the incoming data from the form.

Get method

The get method appends the incoming data at the end of the path which is

"URL/cgi-bin/WIZ.cgi" in this example. One long line with the incoming data appended at the end (parameter string) is returned in GET method.

Post method

If the parameter string becomes long, the Get process may die in the Web server. The post method overcomes this limitation by allowing large sets of data to be blocked together and sent to the server as a data block rather than as a URL with data appended at the end. The post method is always used in WIZ.

<INPUT TYPE=text NAME="name" SIZE=16 VALUE="Ozan">

An entry field with a "name" name and a value "Ozan". It can have a value of at most 16 characters long. *Type* attribute can be text, radio, checkbox, hidden, password, submit, reset.

Text Type

Single line text entry field.

Radio Type

For attributes which can take a single value from a set of alternatives. Each radio button field in the group should be given the same NAME. Only the selected radio button in the group generates a name/value pair in the submitted data. Radio buttons require an explicit VALUE attribute.

Checkbox Type

Used for simple Boolean attributes, or for attributes that can take multiple values at the same time. The latter is represented by a number of checkbox fields each of which has the same NAME. Each selected checkbox generates a separate name/value pair in the submitted data, even if this results in duplicate names. The default VALUE for checkboxes is ON.

Hidden Type

No field is presented to the user, but the content of the field is sent with the submitted form. This value may be used to transmit state information about client/server interaction.

Password Type

Same as TEXT attribute, except that text is not displayed as it is entered.

Submit Type

This is a button that when pressed submits the form.

Reset Type

This is a button that when pressed resets the form's fields to their specified initial values.

<SELECT ... >

The SELECT element allows the user to chose one of a set of alternatives described by textual labels. Every alternative is represented by the option element. SELECT is typically rendered as a pull down or pop-up list. An example for a select is below:

```
<SELECT NAME="country">
<OPTION SELECTED>Turkey
<OPTION>USA
</SELECT>
```

If no option is initially marked as selected, then the first item listed is selected. Another possible attribute for SELECT element is MULTIPLE. The MULTIPLE attribute is needed when users are allowed to make several selections, e.g. <SELECT MULTIPLE>. If the MULTIPLE attribute is not used, only one alternative can be selected out of the set.

<TEXTAREA ... >

Allows users to enter more than one line of text. An example for a textarea is below:

<TEXTAREA NAME="address" ROWS=4 COLS=64>
Default value here if exists.
</TEXTAREA>

The example above, opens a text area of 64 characters wide, 4 lines of height.

</**FORM**>

Finally, this HTML tag indicates the end of the form.

Appendix B

CGI Environment Variables

The Web server defines a number of *environment variables* for communicating information with CGI programs. The *CGI environment variables* can be loosely grouped into three categories: program input variables, remote client variables, and HTTP server variables. They will be discussed in this appendix.

B.1 Program Input Variables

QUERY_STRING

Information which follows the '?' in a URL. The information is needed to be encoded.

Example:

This URL sends 'a query string' to the sample CGI program (env.sh): *http://www.sample.com/cgi-bin/env.sh?a+query+string.*

PATH_INFO

The extra path information specified on the URL.

Example:

This URL sends '/extra/path/info' to the sample CGI program: *http://www.sample.com/cgi-bin/env.sh/extra/path/info.* This URL sends both '/extra/path' and 'query string' info to the sample CGI program: *http://www.sample.com/cgi-bin/env.sh/extra/path?query+string.*

PATH_TRANSLATED

The extra path information specified on the URL after it has had virtual to physical translation done on it.

Example:

The document root for this server is '/usr/local/mosaic/htdocs', which is pre-pended to PATH_INFO and the result is PATH_TRANSLATED. This URL sends '/some/doc' to the sample CGI program:
http://www.sample.com/cgi-bin/env.sh/some/doc

REQUEST_METHOD

The HTTP method being used to send client data to the server. The possible methods are GET and POST. This information is only used with HTML forms.

CONTENT_TYPE

The type of data being sent to the server, from the client. This can be any valid MIME type (but is currently limited to "application/x-www-form-urlencoded"). This information is only used with HTML forms.

CONTENT_LENGTH

The length of the data being sent from the client. This information is only used with HTML forms.

B.2 Remote Client Variables

REMOTE_HOST

The fully qualified domain name of the requesting client. If the name is not available, this variable is not set.

REMOTE_ADDR

The IP address of the requesting client.

REMOTE_USER

The userid that the remote user authenticated with. Example: This document is protected with authentication. Use userid cgi and password wanabe. http://www.sample.com/cgi-bin/auth/env.sh

AUTH_TYPE

A document may be protected on the HTTP server. If it is, this variable is set to the type of authentication used to verify the remote user.

HTTP_*

The header data received from the client is bound to environment variables that begin with HTTP. For instance, if the client sends the header User-agent: Mosaic, an environment variable named HTTP_USER_AGENT will be defined with the value Mosaic.

B.3 HTTP Server Variables

SERVER_NAME

The hostname or IP address of the HTTP server machine (The machine where the CGI program is running on.)

SERVER_PORT

The TCP/IP port number the server has open and is accepting connections on.

SERVER_PROTOCOL

The name and version of the protocol being used by the client to communicate with the server.

SERVER_SOFTWARE

The name and version of the HTTP server software that the CGI program is running under.

GATEWAY_INTERFACE

The version of the CGI specification supported by the HTTP server software.

SCRIPT_NAME

The virtual path to the CGI program. Useful for scripts that must reference themselves in generated URLs.

Appendix C

Table Definitions of WIZ Database

WIZ System MiniSQL Database Table Definitions

The Student Table

```
*********************************************************
create table student (
        userID          char(16) primary key,
        passwd          char(16),
        fullname        char(32),
        email           char(32),
        age             int,
        sex             char(6),
        address         char(128),
        city            char(16),
        country         char(16)
)
*********************************************************
```

The student table holds various information about the student. A new record is inserted into this table whenever a student registers to the system. Every student has a unique user-ID and therefore the *userID* is the primary key in this table. Trivially, the *passwd* field stores the password of the student in the WIZ system, the *fullname* field stores the name and the surname of the student separated with a '_' character, the *email* field stores the email address of the student, and the other fields hold the age, sex, address, city and the country information.

The Course Table

```
*********************************************************
create table course (
        cname           char(32) primary key,
        descript        char(400),
        creator         char(32),
        email           char(32),
        news            int,
        newsadr         char(64)
)
*********************************************************
```

The course table holds information about the courses in the system. Every course has a unique record in this table. Each course has a unique name that is stored in the *cname* field. The *cname* field is the primary key for this table. The *descript* field stores the course description. The course description information is directly taken from the first line of the course template by the *Course Registration Subsystem* during the registration of a course to the WIZ system. Each course has a template in the course template library, see figure 5.1 in chapter 5, and the first line of a course template holds the course description of that course. The *creator* field stores the name and the surname of the course creator separated with a '_' character. The *email* field stores the email address of the lecturer (course creator). The *news* field either stores a '0' value meaning that the course does not have a newsgroup, or a '1' value meaning the course

has a newsgroup. The *newsadr* field stores the newsgroup address of the course and the information that it contains is ignored if the *news* field stores a '0' value.

Other than the course description information, all the other information about a course is entered directly by the course creator when registering a course to the WIZ system by the *Course Registration Subsystem*, and the course name entered for the *cname* field should be the same with the name of the course template.

The Module Table

```
*********************************************************
create table module (
        mname           char(32) primary key,
        descript        char(400),
        creator         char(32)
)
*********************************************************
```

The module table holds information about the modules in the system. Modules are automatically inserted into the module table when a course is being registered to the system. When the lecturer enters the course name during course registration, the *Course Registration Subsystem* reads the course template getting the module names comprising the course, and meanwhile reads the first line of every module template (getting the module description) and inserts a new record into the module table. The *mname* fields stores the name of the module obtained from the module template, the *descript* field stores the module description and the *creator* field stores the name of the course creator.

The Student-Course Table

```
*********************************************************
create table stucour (
        userID          char(16) not null,
```

```
        cname           char(32) not null,
        times           int,
        cl              int,
        ml              int,
        lev1            char(8),
        stat            int,
        cl2             int,
        ml2             int,
        lev2            char(8),
        normcl          int,
        normml          int,
        flag            int,
        lev             char(8),
        private         int
)
**************************************************
```

A new record is added to the student course table whenever a student registers to a course, and vice versa, a record is deleted when a student unregisters from a course. The *userID* and *cname* (course name) pair is unique for every record in this table. The *times* field stores the number of times a student has started a course by pressing on the *Course Start* button in the WIZ main page (see figure 6.5 in chapter 6). When the student registers to a course, this field is set to '0'. When a student presses the *Course Start* button and starts a course, this field is increased by one. This field is important in the sense that when it stores a value equal to '1', new records are created in the student model table. In other words, when the student starts a course for the first time, new records are created only once in student model table for that student.

As can be seen in the table definition, there are two pairs of fields; the *cl, ml, lev1* and the *cl2, ml2, lev2* pairs seperated by the *stat* field. First of all, let's note that *cl* stands for course line number and *ml* stands for module line number. An *cl* and *ml* pair indirectly gives the name of a web page in the web page library or a test template name in the test template library. The name of a web page or a test template is determined by a *cl, ml* pair as follows: The

Course Presentation Subsystem reads the *cl*th line in the course template and gets a module name, the *ml*th line of the module read. gives the name of a web page or a test template name. The *stat* field stores a value of either '0' or '1'. It stores a value of '0' during a *default* flow and it stores a value of '1' during a *dynamic* flow (see course flow types in chapter 6). When the *stat* field holds '0' (default flow), the *cl2, ml2, lev2* fields are ignored. When the *Stop at this point* button is pressed (see control panel section in chapter 6), the *cl,ml* pair holds the web page name of the point where the student has stopped and bookmarked the course presentation during *default flow*, and the *lev* field holds the level of student in that module. Later, when the student wants to continue the course presentation, he/she continues from this page. In a dynamic flow situation, (i.e. when a module is being reviewed by the student) if the student presses on the *Stop at this point* button, the *cl, ml* pair holds the web page that is currently being displayed with the *lev1* field storing the level of the student in the module that is being reviewed, and the *cl2, ml2* pair hold the web page where the dynamic flow was started with the *lev2* field storing the level of the student in that module. As a result, during dynamic flow when the current page is bookmarked and the course is restarted later, the course presentation starts from the bookmarked page using the information stored in the *cl, ml, lev1* fields. When the dynamic flow ends (that is the module is reviewed), the *Course Presentation Subsystem* uses the information stored in the *cl2, ml2, lev2* fields to return back to the page in the default flow.

In the control panel section of chapter 6, it was mentioned that the student had the freedom to go to any web page in the course using the *Goto a specific page* button, thus changing the default flow of the course presentation. It was also mentioned that when the student used this facility, there appeared a new button, namely the *Default Flow* button in the control panel, which is used to take back the student to the web page where he was in the *default* course flow. The *flag* field controls this. A '1' value means that the student is in a web page which he/she accessed manually and the *Default Flow* button exists in the control panel. When the *Default Flow* button is pressed the *Course Presentation Subsystem* uses the information stored in the *normcl* amd *normml* fields to take the student back to the web page in the default flow. The *lev* field stores the level of the student in the page referenced with the *normcl,*

normml pair. If a '0' value is stored in the *flag* field, the control panel does not have a *Default Flow* button and therefore the information stored in the *normcl*, *normml* and *lev* are not used. Finally, the *private* field stores the information of whether the student's email address will be listed in the class communication list or not (see figure 6.12 in chapter 6).

The Student Model Table

```
*********************************************************
create table smodel (
        userID          char(16) not null,
        cname           char(32) not null,
        mname           char(32) not null,
        posval          real,
        negval          real,
        unans           int,
        ans             int,
        correct         int,
        skipped         int,
        unskipped       int,
        level           char(8),
        review          int,
        revskip         int
)
*********************************************************
```

Each record in this table, holds the student's progress and his/her score in a particular module of a particular course. Records for a course are created when a student starts a course for the first time. These records, the student's scores and statistics in every module of a course, are maintained as long as the student is registered to the course. Modifications on these records are made during course presentation.

The data stored in the *userID, cname, mname* fields, (user-ID, course name and module name together) specify a unique record in this table. The *posval*

and *negval* fields store the positive and the negative scores of a student in a particular module. The positive and negative scores are used by the *Course Presentation Subsystem* to figure out if the student needs to review this module or not. When a student reviews a module, the *posval* and the *negval* fields of the reviewed module are initialized. The *unans* field stores the number of unanswered questions in the test(s) which are not skipped in a module. The *ans* field stores the number of answered questions in a module. The *correct* field stores the number of correct answers given in a module. The *skipped* field holds the number of skipped tests in a module, likewise the *unskipped* field holds the number of unskipped tests, in other words the number of solved tests in a module. The *level* field stores the level of the student in a module. The *review* field stores the number of times this module has been reviewed. The *revskip* field stores how many times the student was figured out that he/she had to review this module but the student did not want to review and skipped the reviewing this module. In the initial creation of records all the *posval, negval, unans, ans, correct, skipped, unskipped, review revskip* fields are set to zero, and during course presentation, these fields are modified.

Appendix D

A Sample Database Query Operation

A Database Insert Operation

```
#include "msql.h" /* MiniSQL library */

/* Sample insert query */
#define INS_STUDENT "insert into student (userID,passwd,fullname,
     email,age,sex,address,city,country) values
     ('%s','%s','%s','%s', %d, '%s','%s','%s','%s')"
int sock;
char qbuf[500]; /* Query buffer */
char *dbname = "WIZ"; /* Database name */

void main()
{
   /* Connecting to the MiniSQL database server */
   if ((sock = msqlConnect(NULL)) < 0)
   {
      printf("Couldn't connect to engine!\n%s\n\n", msqlErrMsg);
```

```
        perror("");
        exit(1);
    }

    /* Specifying the database name */
    if (msqlSelectDB(sock,dbname) < 0)
    {
        printf("Couldn't select database %s!\n%s\n", dbname, msqlErrMsg);
        msqlClose(sock);
        exit(1);
    }

    /* Forming the query string in the 'qbuf' variable */
    sprintf(qbuf,INS_STUDENT, 'ozhan', 'password123', 'Ozan_Ozhan',
      'ozhan@bilkent.edu.tr', 23, 'male',
      'Bilkent Univ. CS Dept', 'Ankara, 'Turkey');

    /* Performing the insertion operation on the database */
    if (msqlQuery(sock,qbuf) < 0)
    {
        printf("Query failed (%s)\n", msqlErrMsg);
        msqlClose(sock);
        exit(1);
    }
    printf("Insertion has been successfully performed\n");
    msqlClose(sock); /* Closing the database server connection */
}
```

Bibliography

[BLCM95] T. Berners-Lee, D. Connolly, and MIT/W3C. Hypertext Markup Language - HTML 2.0. Available at http://www.w3.org/pub/WWW/MarkUp/html-spec/, September 1995.

[BP96] John M. Barrie and David E. Presti. The World Wide Web as an Instructional Tool. *Science*, 274:371–372, October 1996.

[Byt] Sid Bytheway. Introduction to CGI Programming Tutorial. Available at http://www.usi.utah.edu/cgi-programming/.

[Cos92] Ernesto Costa. *New Directions for Intelligent Tutoring Systems.* Springer-Verlag, 1992.

[Dav90a] D. Davenport. An Extended User Interface for CAL Systems. *Computers and Education*, 14(4):335–342, 1990.

[Dav90b] D. Davenport. HyperCAI: CAI mets Hypertext. In M. de Blasi, editor, *Proceedings of Education and Application of Computer Technology*, pages 377–388, Barcelona, Spain, September 1990.

[DGB89] D. Davenport, A. Guvenir, and F. Buyukkokten. Tutor: An Experiment in CAL. In *Proceedings of the fourth International Symposium on Computer and Information Sciences*, Cesme, Izmir, Turkey, November 1989.

[Doc96] NCSA's CGI Documentation. The Common Gateway Interface. Available at http://hoohoo.ncsa.uiuc.edu/docs/cgi/, March 1996.

[fSA96] NCSA: The National Center for Supercomputing Applications. A Beginner's Guide to HTML. Available at http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html, December 1996.

[Hug95] David J. Hughes. Mini SQL: A Lightweight Database Engine. In *QAUUG Summer Technical Conference*. QAUUG, April 1995.

[HWV96] M. Hamalainen, A. B. Whinston, and S. Vishik. Electronic Markets for Learning: Education Brokerages on the Internet. *Communications of the ACM*, 39(6):51–58, 1996.

[Ibr94] Bertrand Ibrahim. Distance Learning with the World Wide Web. In Erlangen, editor, *International Conference on Open and Distance Learning - Critical Success Factors*, pages 123–126, Geneva, Switzerland, October 1994. FIM.

[ICD97] ICDE. International Council for Distance Education. Available at http://www.cde.psu.edu/ICDE/, June 1997.

[IF95] Bertrand Ibrahim and Stephen D. Franklin. Advanced Educational Uses of the World-Wide Web. *Computer Networks and ISDN Systems*, 27(6), 1995.

[Man96] Mini SQL Manual. Mini SQL Version 1.0.11. Technical report, Hughes Technologies Pty Ltd., January 1996.

[Man97] IEIS Manual. CGI For UNIX WWW Servers, May 1997.

[McC92] Gordon I. McCalla. The Central Importance of Student Modelling to Intelligent Tutoring. *Computer and Systems Sciences*, F91:107–131, 1992.

[OD96] O. Ozhan and D. Davenport. WIZ: An Intelligent Dynamic Web-based Hypermedia System. In *Proceedings of the first International Symposium on Distance Education*, Ankara, Turkey, November 1996. FRTEB.

[Syn96] Dan Synder. The World Wide Web as a Teaching Tool. Available at http://www.wayne.edu/wtt/wtt.html, September 1996.

[Wen87] Etienne Wenger. *Artificial Intelligence and Tutoring Systems.* Morgan Kaufmann Publishers, Los Altos, CA, USA, 1987.

[Wer96] Kevin Werbach. The Bare Bones Guide to HTML. Available at http://werbach.com/barebones/barebone.html, July 1996.