5-1997

NetTest: An Integrated Web-Based Test Tools

Tzy-Tao Yang
Utah State University

NETTEST: AN INTEGRATED WEB-BASED TEST TOOL

by

Tzy-Tao Yang

A report submitted in partial fulfillment
of the requirement for the degree

of

MASTER OF SCIENCE

in

Computer Science

Approved:

Dr. Jianping Zhang Major Professor	Dr. Donald H. Cooley Committee Member

Dr. Larre Egbert
Committee Member

UTAH STATE UNIVERSITY
Logan, Utah
1997

ABSTRACT

NetTest: An Integrated Web-Based Test Tools

by

Tzy-Tao Yang

Utah State University, 1997

Major Professor: Dr. Jianping Zhang
Department: Computer Science

This report presents the design and implementation of *NetTest*; a Java program that can be run within a Java enabled Web browser. *NetTest* is a Web-based test tool, which allows instructors to create or edit tests and students to take tests using a Java-enabled Web-browser. It also allows managers to perform their tasks using similar tools.

NetTest is a server/client program, which is located on the server. While using the *NetTest* system, the Web browser will automatically download the program and execute it on the client machine. In other words, *NetTest* can be run on any kind of network environment as long as the network supports the TCP/IP protocol. It can be run on the Internet, an Intranet, a WAN, or a LAN.

NetTest is a cross-platform program, for which the servers and clients can be any kind of platforms and the source code needs not be modified or recompiled.

This report examines the structure of the *NetTest* with several Object-Oriented models. The structure of the *NetTest* database is illustrated by using data flow diagrams and table definitions. A preliminary user manual is also included.

ACKNOWLEDGEMENTS

I would like to thank my Advisor, Dr. Jianping Zhang, for his guidance and assistance throughout this project. I am especially grateful to my committee members, Drs. Donald Cooley, and Larre Egbert, for their support and assistance throughout the entire project. I also would like to thank my teammate Yun Ni, for completing the student module and giving positive suggestions.

Special thanks go to my friends, and colleagues for their encouragement and support. Finally, I wish to thank my beautiful wife, Shao-Jan, for her tremendous support and patience.

Tzy-Tao Yang

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

As computer technologies evolved, the use of computer-aided training and education has dramatically increased. The biggest advantage of using computer-based programs in training and education is their flexibility and uniformity. Once the educational material is prepared, without being tied to a schedule, students can take lessons at any time when they have access to a computer. Besides, training and education should always be accompanied with assessments or tests for evaluation of the quality of learning. The time and efforts spent on creating and grading a test are considerable burdens for instructors. By using a computer program, instructors can easily evaluate how well the students have learned. The students can know their grades right after the test. At the same time, the computer can store the results for each student that used the system and create comparison or statistics reports. The instructors can adjust the course content according to the results of students' performance. In this way, the student can learn better, the instructor can spent more time on improving the quality of the course contents, and let computers take care of the tedious work.

WHY NETTEST

The Internet technology has grown rapidly in the last few years. The Internet population reached 20 to 30 million on August 25th, 1995 [1] and projects to reach 15.79 billion in year 2000 [2]. Because of its popularity, connectivity, and easy of use, the Internet is the ideal place to put a testing tool.

There are some testing tools available on the Web. Some of them, such as the Internet Test System [3], use the Java language while others use HTML and the CGI programs to do the work, such as the EDUTEST [4]. But these tools only support limited functions and do not have an authoring tool for creating tests. Therefore, there is a need to build a testing tool that is able to handle various types of questions, which has an authoring tool for instructors to create tests, and which maintains a student test record for tracking or improvement purpose.

GOALS OF NETTEST

The followings are the three goals for building *NetTest*:

1. Build A Web-Based Test Tool

Let the users use the system from anywhere in the world is the first goal of *NetTest*. This world-wide accessibility will make the test more flexible than traditional in-class test. For example, in the case of remote education, both students and instructors can benefit from *NetTest*. Since the students in remote education session have difficulties to come to class for lectures, it can be difficult for them to come to class for taking tests as well. If the *NetTest* is been used, students will be able to take tests from anywhere. The instructors can also work on the tests remotely. They can decide which test can be taken remotely without proctor and put the test on the system.

2. Build A Cross-Platform Test Tool

The second goal of *NetTest* is to build a cross-platform test tool. Figure 1-1 shows this scheme. With this feature, the system can be implemented on any kind of machines or

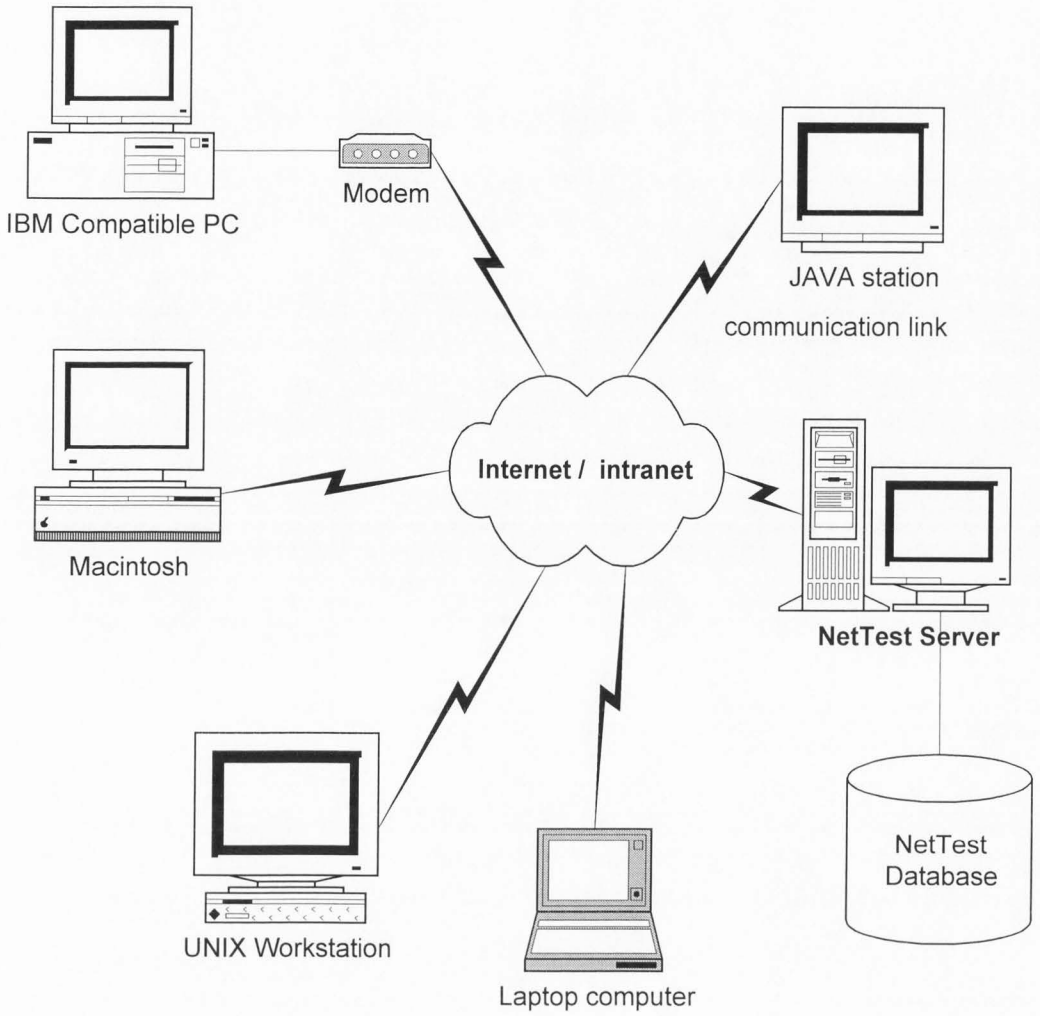


Figure 1-1. The schema of NetTest

operating systems. There are several types of hardware and several different operating systems on the market. Therefore, there are many hardware and operating system combinations. If *NetTest* users can use the program in a computer environment they are already familiar with, it will be easier for the users to adapt themselves to the system. This goal can be achieved by using Java as the programming language, because Java is designed to be a cross-platform programming language. This feature also will greatly reduce the efforts of developing and maintaining the program. The source code of the

NetTest only needs to be written and compiled once and it will be able to run on different platforms.

3. Develop A Server/Client Test Tool

In *NetTest*, all the information related to the users and tests are stored in the database on the server. The users can use a Java enabled Web browser to log on to the system and the server will send information corresponding to the user requests. There is only one single database that needs to be maintained, and all user information is stored in one host computer. In the case of multiple test sites, we do not need to worry about creating and maintaining another database. Therefore, there will not be the problem of maintaining consistency between databases. Meanwhile, a user will not be limited to one test site, he/she can choose any test site that is accessible concerning the place and time.

OVERVIEW OF REPORT

Chapter 2 explores the structure of *NetTest* from an object-oriented point of view. This chapter also describes the database structure of the *NetTest*. Chapter 3 is the preliminary user manual that explains how to use the system from different users' perspectives. Chapter 4 is the conclusions, discussing the current state of the *NetTest* and the pros and cons of using Java. Chapter 5 gives some suggestions for future development. Appendix A lists the definitions of the tables in the *NetTest* database.

CHAPTER 2

THE STRUCTURE OF NETTEST

There are five major components of the *NetTest*: the manager module, the student module, the instructor module, the database, and two daemon programs. The Object-Relationship Model (ORM) between these modules and the users is illustrated in Figure 2-1.

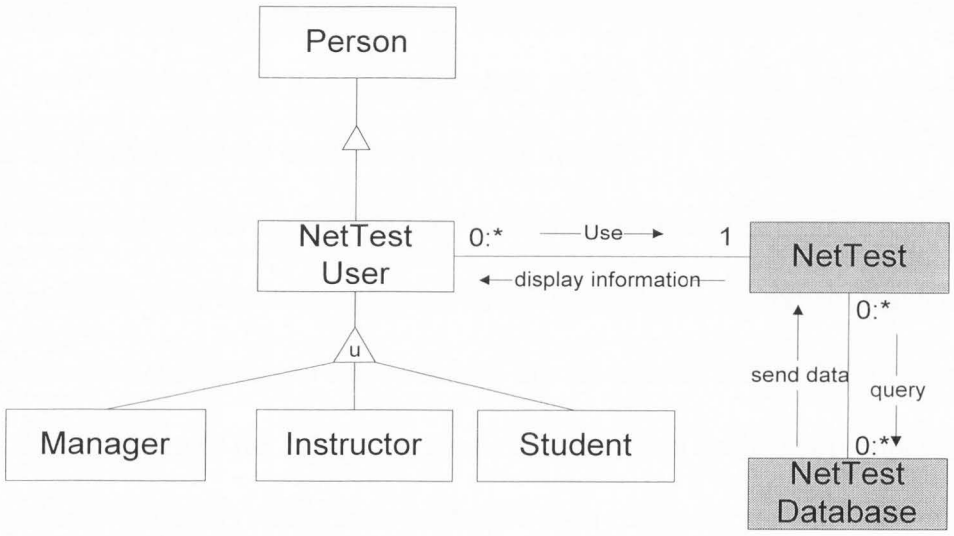


Figure 2-1. The Object-Relationship Model of NetTest

This model employs the Object-Oriented Systems Model (OSM) developed by David W. Embley et al. [5]. The rectangle box represents an object class. The shaded rectangle box represents a high level view of an object class that can be further divided into several subparts. The arc represents the relationship between two object classes. The number over the connected line means the participation constraints. The basic form for a

participation constraint is a pair of min:max value. The participation constraint is the number of times an object in an object class can participate in a connected relationship set. The "*" represents an infinite number. The triangle represents the "is a" relationship, also know as inheritance relationship. The union symbol (∪) inside the triangle means that the inheritance is a union set. The union means that an object in the super class could be at least fit in one of the subclass.

From the ORM in figure 2-1, the *NetTest User* has three sub-types: student, manager, and instructor. The *NetTest* users use the *NetTest*. The *NetTest* sends queries to the database, and the database sends data back to the *NetTest*. Then the *NetTest* will display the information back to the users. In later section, we will see more detailed views of the *NetTest* and the database within this system.

For this project, I am responsible for developing the instructor module and the manager module, while the student module is developed by my teammate Yun Ni. I constantly receive feedback and opinions from my supervisors during the development process. The structure of the database is a collective decision made by my supervisors, Yun Ni, and me. Basically, these three modules do not communicate with each other, but they share some information of the database. The detailed structure of these three modules and the database are introduced in the following sections.

MANAGER MODULE

The ORM of manager module is shown in figure 2-2. This ORM is derived from figure 2-1 and expands one level down to present more details of the manager module. The solid black triangle represents the "is part of" relationship between objects. For

example, in the graph, the user information is part of the *NetTest* database and username is part of the user information. The remaining notations were introduced in the previous section. The ORM of the manager module illustrates how those object classes are involved in the manager module and how they are related to one another.

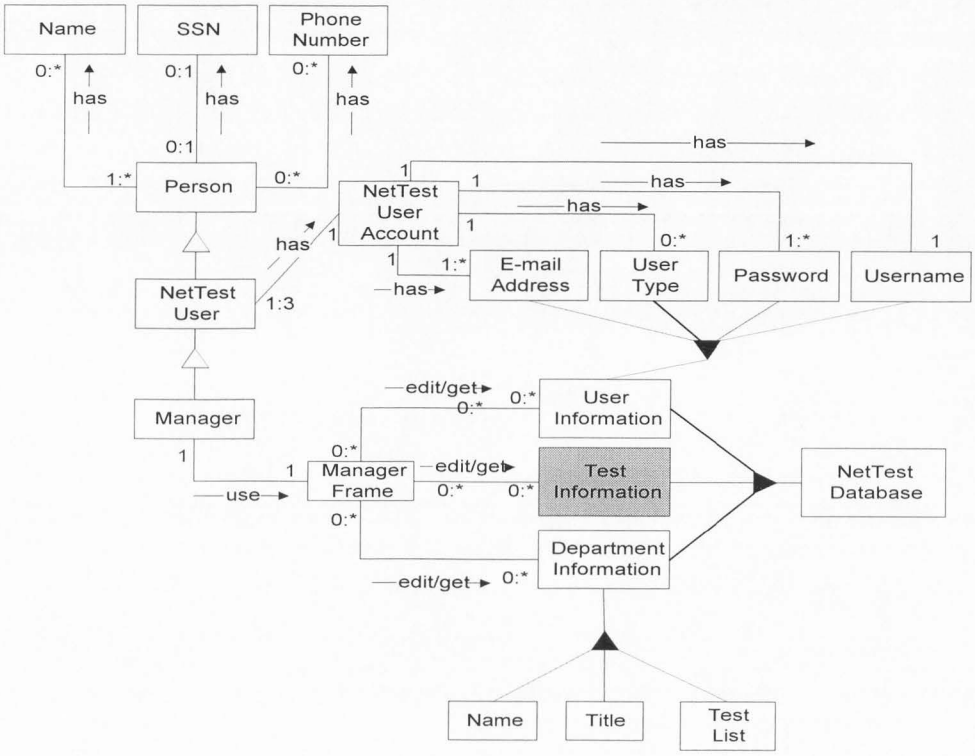


Figure 2-2. The Object-Relationship Model of the manager module

This ORM shows that every person can have zero to many phone numbers and each phone number can belong to many persons. A person can have only one social security number and each social security number belongs to one person at most. A person has at least one name and every name can belong to different persons.

To distinguish *NetTest* users from the general public, we give each *NetTest* user a unique account belonging to only one user. However, for security reasons, every account

has a user type that allows the user to perform certain jobs correspond to the user's privilege. In the case that one person has multiple roles in the *NetTest*, the system manager needs to create an account for each role the person plays. The system identifies different users by the username, therefore, the username is unique to the system and it can only belong to one user account. Every user account has a password for the user to logon. Besides, each user account has an e-mail address for the system and the manager to contact a user electronically.

The tasks that the managers perform in the *NetTest* are: maintaining the user information, test information, and department information. Management tasks consist in creating and editing the different kinds of information. The managers use the manager frame to perform these tasks. The department information includes the name, the title, and the test list of the department. The test list is the tests that are required for that department. The test information in this model is a high level view. The details of test information are revealed in the ORM of the instructor module.

INSTRUCTOR MODULE

Figure 2-3 shows the ORM of the instructor module. The diamond shape and the statement beside the diamond shape in the graph shows the general constraint. The general constraint is used for restricting the membership of one or more classes. There are two general constraints in this model: the instructor uses instructor frame to change password, and the instructor uses authoring frame to create or edit test. The test contains sections and each section has one or more questions.

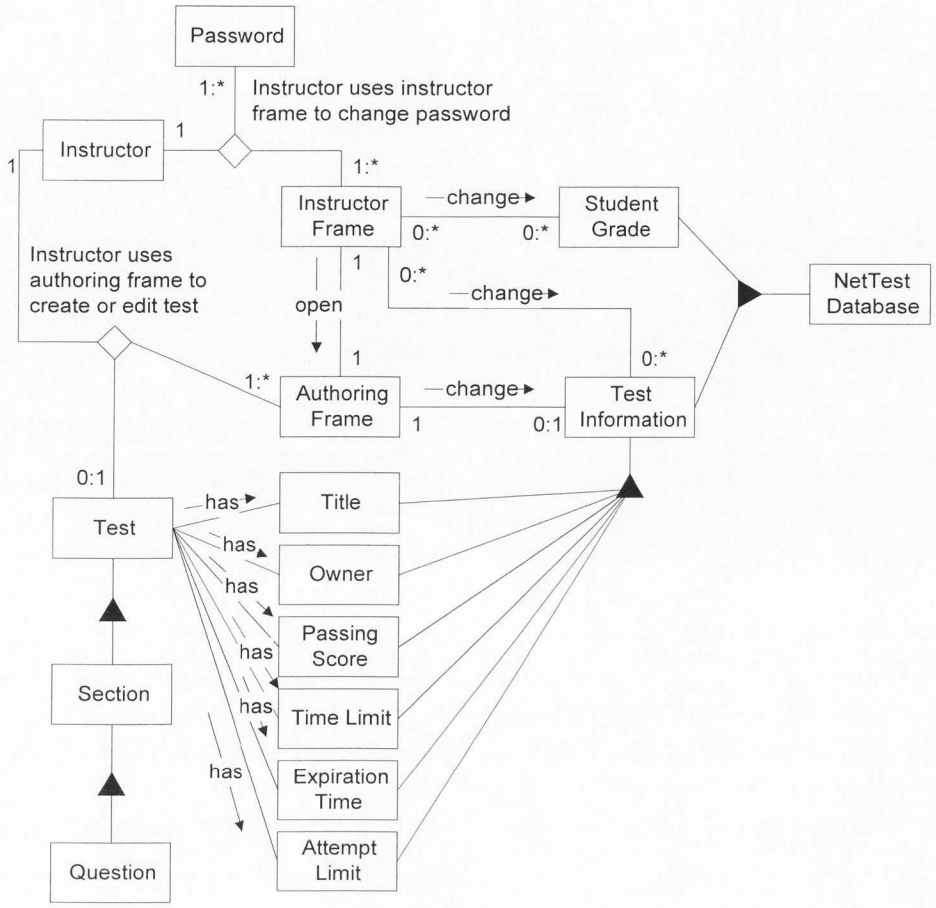


Figure 2-3. The Object-Relationship Model of the instructor module

The test has its title, owner, passing score, test-taking-time limit, expiration time, and maximum number of attempts. All the information belongs to the test information in the database. The major tasks of an instructor in the *NetTest* are: creating tests, editing tests, and change students' grades for a certain test. In the instructor frame, the instructors can choose to change the students' grade of any test or change the information of any test, so that the participation constraint is 0:*. The instructors can also decide to open an existing test or create a new test. In this case, the instructors will open the authoring frame to let

the instructors do so. Once inside the authoring frame, the instructors can change the information of the opened test, add new sections, add new questions, or edit questions. When a test is opened, it is locked, so that no other instructors can access that test. Therefore, every authoring frame relates to one test; a test and its information relate to one authoring frame only.

STUDENT MODULE

The ORM of the student module is shown in figure 2-4. There is no new notation introduced in this figure. The student module was built by my teammate, Yun Ni.

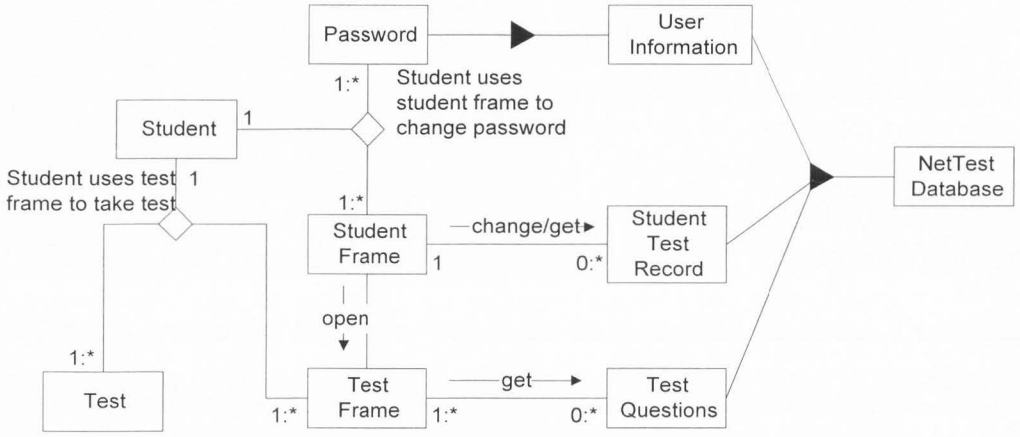


Figure 2-4. The Object-Relationship Model of the student module

Once logged on to the *NetTest*, the student frame is presented to the student. In the student frame, a student can change the password, view the test record, or take a test. Because the student can only change or view his/her own records, the participation constraint is 1. When the student decides to take a test, the student frame opens the test frame for the student to take a test. The student can take one test at a time. The test is

presented to the student with questions not grouped by the section. Therefore, only test questions are related to the test in the student module.

STRUCTURE OF NETTEST DATABASE

All the information related to the *NetTest* is stored in the database. In previous sections, I introduced the structure of the *NetTest*. Before looking into the database, let's examine the data flow diagram (DFD) of each module.

A typical data flow diagram contains data flows, processes, databases, and sources or sinks of data. The notation used in these data flow diagrams is extracted from "Software Engineering" by Dr. Gregory W. Jones [6], where the rectangles represent sources and sinks, circles represent processes, arcs represent data flows, arc labels represent data objects, and parallel lines represent databases.

Data Flow Diagram Of Manager Module

The data flow diagram of manager module is shown in figure 2-5. When the manager logs on the system, he/she needs to enter the username and password. The system looks up the user information database to find a match and verify the password. When the manager wants to create a new user account or edit the existing user account, the manager sends a request to the system, the system sends the query and gets the information from the user information database. When updating the test, the system checks the number of the logon user by sending the test ID received from the manager to the log database and gets the user count. If the user count equals to zero, the system updates the test immediately; otherwise, the test is scheduled for updating at a later time.

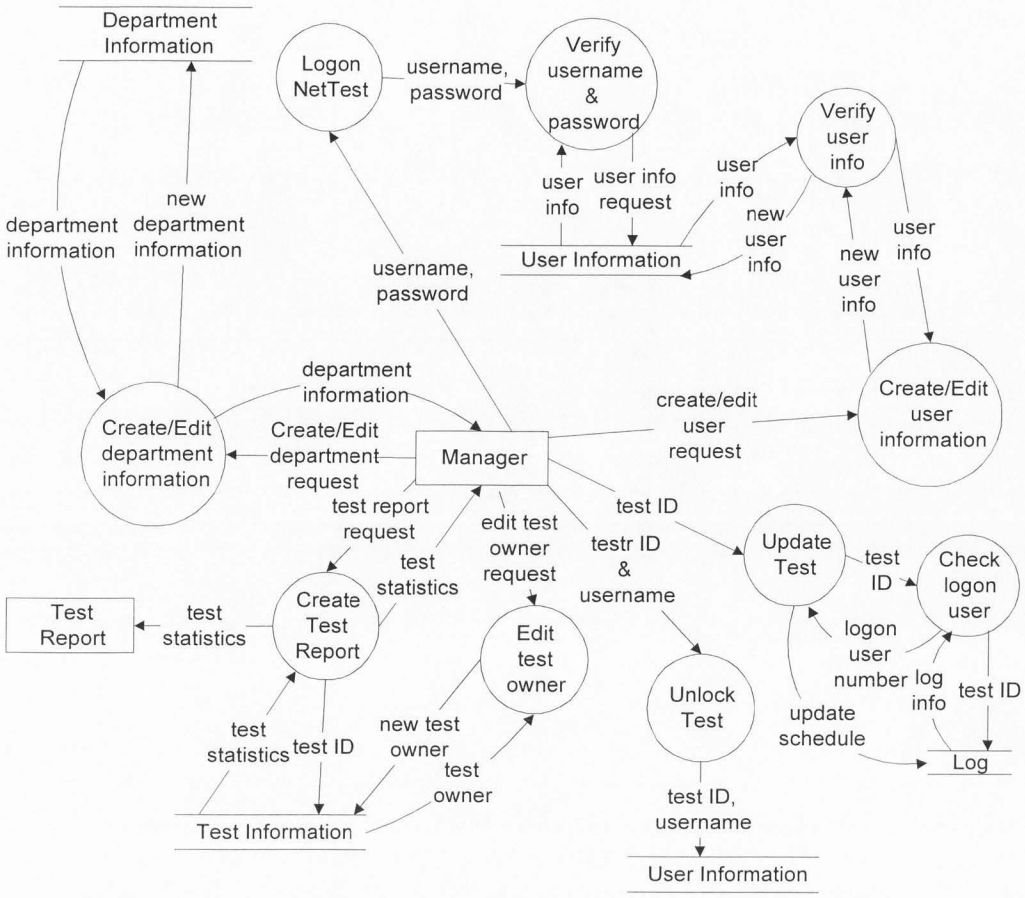


Figure 2-5. The data flow diagram of the manager module

Some tests need to be unlocked before a student can take that test. Upon receiving the request, the manager sends the test ID and the student's username to the system. The system then sends this information to the user information database. When a student logs on to the system, he/she will be able to take the test.

When an instructor creates a test, he/she is the default owner of the test. But there are cases in which a test needs to be edited by more than one instructor. In this case, the manager needs to change the ownership of a test to those instructors who need access to the test. For this end, the manager sends a request to the system. The system gets the

current owner from the database, then the manager changes it and sends it back to the system. Finally, the system changes the ownership in the database. When creating test reports, the manager sends a request. The system will let the manager choose from different tests and reports. Then, the system will get the corresponding information from the database, format the report, and send it back to the manager. When creating a new department or editing an existing department, the managers need to send a request. The system saves the updated information in the database.

Data Flow Diagram Of Instructor Module

Figure 2-6 shows the data flow diagram of the instructor module. It shows the tasks performed by the instructor and the data flows. When changing the password, the instructor sends the new password and the system saves it in the user information database. When adding or editing a section, the instructor sends the new section information to the system, then the system saves the information into the instructor test database. The same occurs when the instructor adds or edits test questions. The instructor sends a test ID and new test information to the system when editing the test information, and the system saves the changes to the instructor test database. When deleting or opening a test, the instructor sends the test ID to the system, then the system performs the required action. When creating a new test, the system will request for the new test information and saves it to the instructor test database.

After a test information or its contents have been changed, the instructor needs to decide whether to update the test, so that the student can use the new test. When this happens, the instructor sends the test ID to the system, and the system gets the test

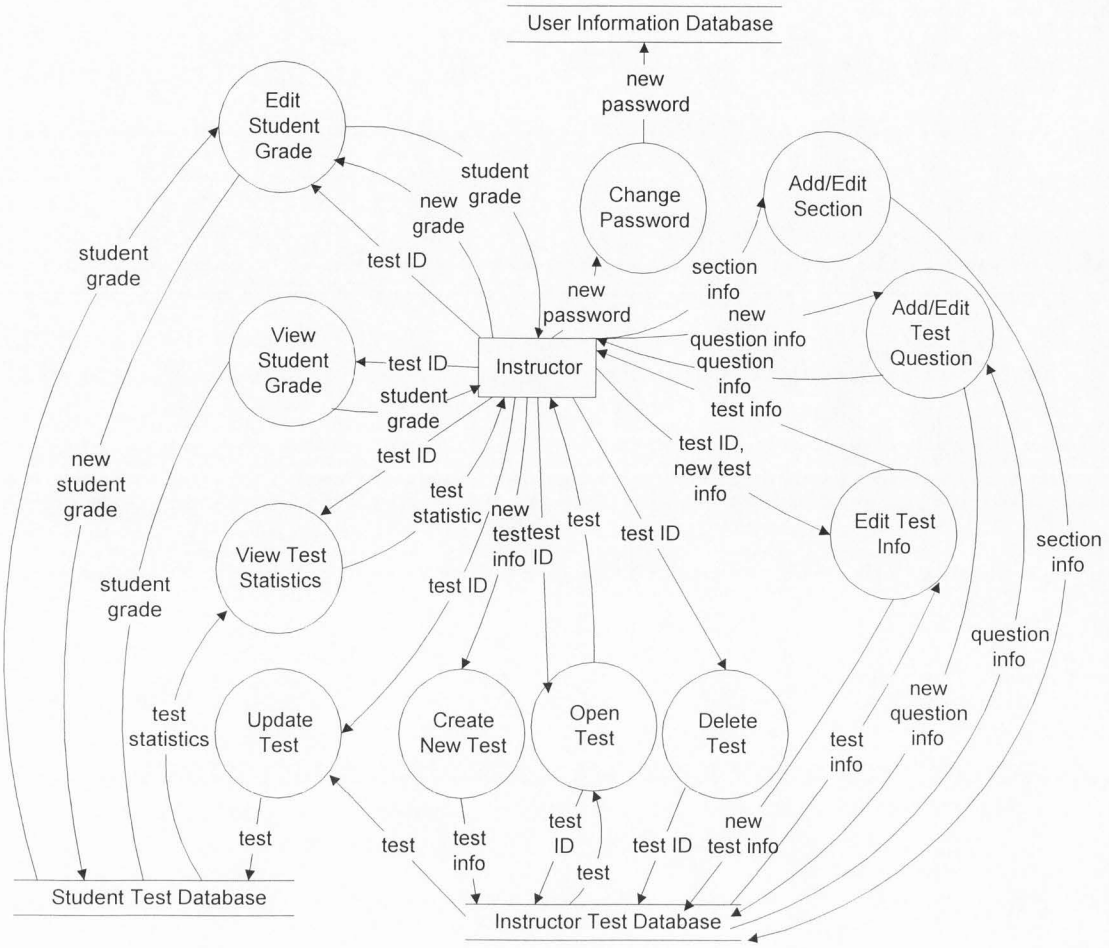


Figure 2-6. The data flow diagram of the instructor module

information from the instructor test database, writing the information to the student database. The reasons to keep separate databases for instructor and student are that there are cases in which the instructor is not finishing his/her work on the test or when there are students taking the test. The new information cannot appear on the test underway, so that the system saves the test information in a temporary storage and writes it to the student database when it is not under lock.

The instructor can also view the test statistics by specifying the test ID to the system,

which will retrieve the test statistics from the student test database. A similar process happens when the instructor wants to view the student grade. When editing the student grade, the instructor gets the current grades from the system by supplying the test ID and sends the new grade back to the system, which saves the new student grade to the student test database.

Data Flow Diagram Of Student Module

Figure 2-7 shows the data flow diagram of the student module. The student sends username to the system and the system gets the grade of this particular student from the student grade database.

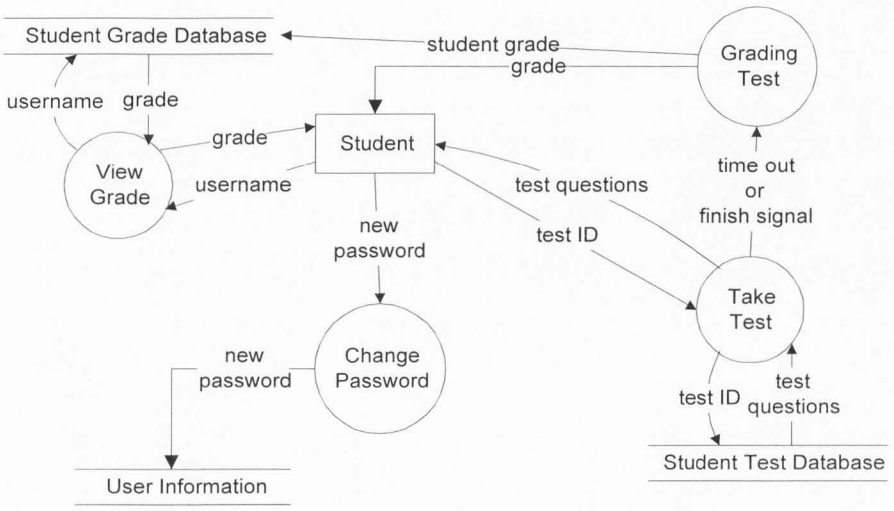


Figure 2-7. The data flow diagram of the student module

When taking a test, the student sends the test ID to the system, which sends back the test questions. When the student has finished the test or the test duration is elapsed, the system grades the test and sends the grade both to the student and to the student grade database. In order to change the password, the student needs to give the system the old

and the new password and the system saves it in the user information database.

DAEMON PROGRAMS

The two daemon programs are named *log server* and *update daemon*. Both programs are continuously running on the server as Java applications.

Log Server

The responsibility of the *log server* is to log the user logon and logout activities. When a user logs on to the system the *log server* records the user name, logon time, and the IP address of the machine from which the user connected. This log will be kept in a file to monitor the traffic of the system. The *log server* will prevent the same user to log on to the system with multiple simultaneous connections. It will check whether the test the user is trying to take is scheduled to be updated, in this case the user is not allowed to log on to the system. It will likewise prevent two instructors editing the same test at the same time.

Update Daemon

The *update daemon* runs on the server, sleeps for the period of time specified during the set up, then wakes up to do its job again. The primary job of the *update daemon* is to update tests when they are so scheduled. Another of its jobs is to check whether there is any entry in the log table to which user's program does has not responded for a long time. In this case, the *update daemon* will consider that the user's program has terminated abnormally or crashed. Therefore, the system will remove this entry from the log table so that the user will be able to log on again.

QUESTION TYPES

There are seven types of question available in the *NetTest* namely: multiple choice, filling the blank, sequencing, matching, multiple answers, true or false, and essay questions.

Multiple Choice

In this type of questions, there has to be at least two choices from which the instructors ask the students to choose, of which there is only one correct answer.

Fill-In-Blank

The fill-in-blank questions have a missing word or phrase in the text. The students need to complete the whole text by determining what is missing, then type in their answer in the spaces provided.

Sequencing

This type of questions has a list of items that have a specific order in their nature; the students need to determine the correct order from the items listed in the question.

Matching

This type of questions contains two sets of items, which maintains a fixed relationship from one side to the other. The students need to determine the correct match of one item on the left side and another item on the right side, according to the fixed relationship.

Multiple Answers

This type of questions is similar to the multiple choice, but it is possible that there is more than one correct answer.

True Or False

In this type of questions, the students need to answer whether the statement in the question is true or false.

Essay

The answer of this question is open-ended, students need to write their opinion based on the question asked. The answer could be more than one sentence or even extended to several paragraphs. Since the answer is not fixed, this type of questions cannot be graded by the system. The instructor needs to grade this type of questions personally.

THE DATABASE ENGINE AND TABLES

The *NetTest* uses Microsoft SQL Server® as the database server. The decision of using Microsoft SQL server® stems from the Windows NT Server® gains popularity in the server operating system market. The installed base of Windows NT Server® is growing larger while the hardware cost of running a Windows NT Server® is relatively lower than that of a UNIX box. On the other hand, the Microsoft SQL Server supports the standard query language (SQL) and it is robust for relational databases. These features keep the cost of this project low and make the design of the database very easy. However, it is fairly easy to switch to another database server, because the application interfaces to communicate with database servers have been standardized. These Application Programming Interfaces (API) are called Java Database Connectivity (JDBC). To change the database server to another one, we only need to change the JDBC driver to the proper

driver that is capable to communicate with the target database server.

The database of the *NetTest* has 24 different tables. These tables can be grouped into three categories: user information database, student test database, and instructor test database. I will discuss these tables in the following sections by category. The table definitions of all the tables are listed in appendix A.

User Information Database

There are four tables in this category: Password, Major, Log, and Std_Info. The Password table is used to store the basic user information, such as username, password etc. The Major table stores departmental information. The Log table stores information of current logon user that is accessing any test. The Std_Info stores the student test records and the result of each attempt.

Student Test Database

There are 10 tables in this category. These tables are named Test, Section, Qindex, and Qtype1 through Qtype7. These tables store the information and questions of tests that the student will take. The test table stores test information such as owner, title, passing score, and time limit. The Section table stores the section information of all tests. The Qindex table stores the reference and statistics of questions of all tests. The system will first look up this table then look into the corresponding Qtype table according its question type. Qtype1 through Qtype7 are used to store different types of questions of all tests. Qtype1 stores the information of matching questions. Qtype2 stores the information of sequencing questions. Qtype3 stores the information of true or false questions. Qtype4 stores the information of multiple choice questions. Qtype5 stores the information of fill-in-blank questions. Qtype6 stores the information of multiple answer questions. Qtype7

stores the information of essay questions.

Instructor Test Database

The remaining 10 tables are in the instructor test database. The tables in this category are a mirror set of the student test database, and their names have "Update_" in front of those in student test database. All the table definition is identical to those in the student test database except Update_Test and Update_Qindex.

In the Update_Test table there are three columns added to the Test table in the student test database. These columns are named updateStatus, graphSeq, and IdSeq. The updateStatus column is for keeping track of the test update status. The possible values of this column are 'i', 'd', 'u', and 'n'. The value 'i' indicates that the test is incomplete, so that the system will not allow update on this test. The 'd' indicates that the test has been deleted, the system will delete the corresponding test in the student test database when no student is taking the test. The 'u' means that the test needs to be updated, the system will update the test upon receiving the request from the manager or instructor. The 'n' means that the test has been updated previously, and no change has been made after that. In this case, the system will not update the test unless the instructor or manager elects to do that.

The Update_Qindex table has four columns fewer than the Qindex table. These columns are timesCorrect, timesIncorrect, timesNotAnswered, and timesSelected, which exist in the Qindex table for the purpose of statistics. They only need to be kept in one place, and the student test database is the best choice.

NETTEST SECURITY POLICIES

Since the *NetTest* is running on the Web, there are some security issues. In the following sections, I will discuss the security policies implemented in the *NetTest*.

General Security Policies

The *NetTest* only allows people that have an account to access it. The users need to log on to the system by entering their usernames and passwords. This piece of information is stored in the database. The password must be at least six characters up to 15 characters long. The password could be composed by any characters. As long as the users keep their password secret, no other person can use their accounts. When changing the password, the system will ask the old password and reconfirm the new password. Because the *NetTest* is running within the browser as an applet, it can not access the resources on the client side.

The security on accessing the database is controlled by the database server. The database is a black box to the user. The user neither knows the name of the database nor the name of the tables. Therefore, the chances that someone will corrupt the database are small. To connect to the Microsoft SQL Server, we need to supply the username and password. Each user in the Microsoft SQL Server has its own permission for reading or writing to the database. The Microsoft SQL Server handles this kind of security very well, therefore, we did not have to add security features. Currently, the username and password for accessing the database is hard-coded in the program. It is possible that someone can hack into the system and get the passwords, however, to get the username and password is difficult for normal users. In order to illegally obtain the username and password, the data-thief needs to know which class files holds the information then try to

save the class file and disassemble the Java byte code. Another way is to capture the packet transmitting between the server and client and unscramble the packet. Both ways would require a lot of work.

When a test is opened, the opened test is locked for the write operation. However, there are cases in which the user terminates the program abnormally or the computer crashes. The lock will not be released, which will cause indefinite waiting or starvation. To fix this problem, there is a monitoring agent running on the background. Whenever the user opens a test, the clock starts to tick. If there is no user action (i.e. mouse movement, keyboard event) for more than 12 minutes, the system terminates the session. If there is any user action before the time running out, the clock is reset to zero. When a student is taking a test and there is no user action for 12 minutes, the system will close the session. The test will not be graded and the test is marked as incomplete. When an instructor is editing a test and there is no user action for 12 minutes, the system will save the test to the instructor test database automatically before closing the session.

Manager Module Policies

There are two types of managers in the *NetTest*: system managers and department managers. The system managers can access system-wide information. The department managers can access the departmental information only. For example, a system manager can create a new user account in any department but a department manager can only create new users for his/her department.

Instructor Module Policies

When accessing the test, the instructor can only read or write those tests that he/she owns. When an instructor creates a new test, he/she is the default owner of the test. For

sharing the test, the instructor needs to ask the manager to add more owners to the test. The same rule applies to viewing the test statistics and viewing or changing the student grades.

Student Module Policies

The major security issues in the student module are related to how the student is taking the test. Basically there are two kind of tests: one needs to be proctored, while the other needs not. For those tests that need to be proctored, the instructor has to specify the tests that need to be unlocked by the manager. When a student wants to take this kind of test, he/she has to ask the manager to unlock the test first. When unlocking a test, the system will ask the manager to specify the student username, IP address and the valid time. The test will be unlocked for one specific student on a particular machine for a certain period of time. For those tests that do not need to be unlocked, the student can take the test at any time and any place.

The test questions presented to the student are randomly selected from the test database. The order of the questions will be different in each instance. The sequence of possible answers is also randomly ordered. Therefore, it is useless for the student to defeat the system by memorizing the order of the answers when taking a test.

THE PROGRAMMING LANGUAGE AND TOOL USED

The programming language chosen is Java because of its portability and built-in networking capability. The developing tool used initially was Microsoft Visual J++, but we switched to Symantec Visual Café two months ago. The JDBC driver for connecting the *NetTest* and Microsoft SQL Server is Fast Forward, purchased from Connect Software Inc.

CHAPTER 3

USER MANUAL

The *NetTest* is a Web-based test tool. With this system, the students can take tests, the instructors can create or edit tests, and the managers can perform management tasks on the Internet. There are three modules in the *NetTest*. Each module is designed for specific types of users. The users who utilize the *NetTest* are categorized into three different types: manager, instructor, and student. Each type of users have different access privileges. There are certain actions allowed for each type of users, while some are inhibited. Therefore, each type of users have their own interfaces with menu choices that allow them to perform a restricted class of tasks. This user manual is organized into five sections:

1. Explains the steps to install the *NetTest* on the server.
2. Lists the system requirements for the server and clients.
3. Explores the functions available for managers and explains how to use them.
4. Tells the students on how to take tests.
5. Explores the functions available for instructors and gives detailed explanation on how to use the system.

INSTALLATION

There are five steps to install the *NetTest* on the server machine.

1. Copy all files to the hard drive

   The files include all the class files, image files, HTML files, and the JDBC driver

files.

2. Create database

Create a database in the server called "nettest" and a logon account for this database. This logon account must have read and write privileges on all tables. The procedures for creating database, logon account, and tables vary among different database servers.

3. Create tables in the database and initialize the tables

Create all tables in the database. The password table must have the account information for the system manager, so that the system manager can logon and create new users or new departments.

4. Set up the environment variables

An environment variable that must be set is "CLASSPATH". The procedure to set up environment variables is different for different platforms, but normally it should be "Set CLASSPATH=%CLASSPATH%;[*NetTest* directory]." The CLASSPATH variable should point to the directory where the *NetTest* system is installed.

5. Set up the Web server

Make the Web browser point to the *NetTest* directory, so that the public can access *NetTest* using the Web browser via the HTTP protocol. Again, the procedure varies on different Web servers.

Currently, there is no setup program written, thus these five steps need to be done manually.

SYSTEM REQUIREMENT

The system requirements on the server are:

- HTTP server or Web server
- Database Server (i.e. Microsoft SQL Server, Oracle)
- JDBC driver for the database server
- TCP/IP network

The system requirements on the client are:

- Java enabled Web browser that supports Java Development Kit (JDK) 1.1 or later, for example, Netscape Communicator 4.04.
- TCP/IP connection to the server

MANAGERS' USER MANUAL

The major component of the manager module is the manager interface. This interface is designed for management tasks; only users with management privilege can access this interface. In this interface, a manager can perform user management tasks, change the departmental information, manage tests, maintain the test database, and generate statistic reports. There are two types of managers: system managers and department managers. A department manager can only access part of the users and part of the tests, while a system manager is allowed to access all the information related to all users, departments, and tests. First, the managers need to log on to the system, by entering their username and password. Figure 3-1 shows the logon box. After a successful logon, the tasks that managers can perform are as listed below:

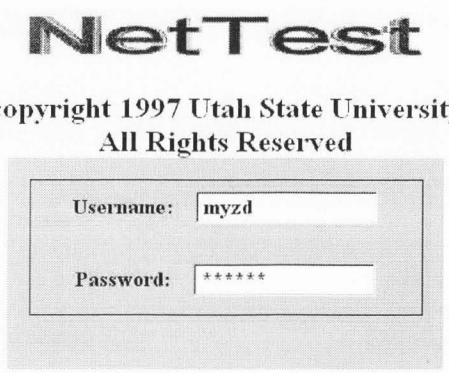


Figure 3-1. The logon screen of NetTest

User Management:

The tasks that managers need to do are to create new user account, edit existing user information, list current users, disable user account, activate user account, and remove user from the database.

a. Create New User Account

The manager needs to create the account for each new user and assign a unique user name to the user, so that the users can logon to the system. For adding new users, the manager needs to select the proper user type from the menu bar, then select "add" from the pull down menu. If the manager is a department manager, then he/she can only create users for his/her own department. Only the system manager is allowed to create users for different departments. Figure 3-2 shows the interface for adding new managers. The interface for adding new students and instructors looks exactly the same as figure 3-2. The only difference is the title. If the manager selects the wrong user type, he/she can rectify this error by choosing the correct user type choice button. The manager needs to input the password twice for verification then select the proper department for

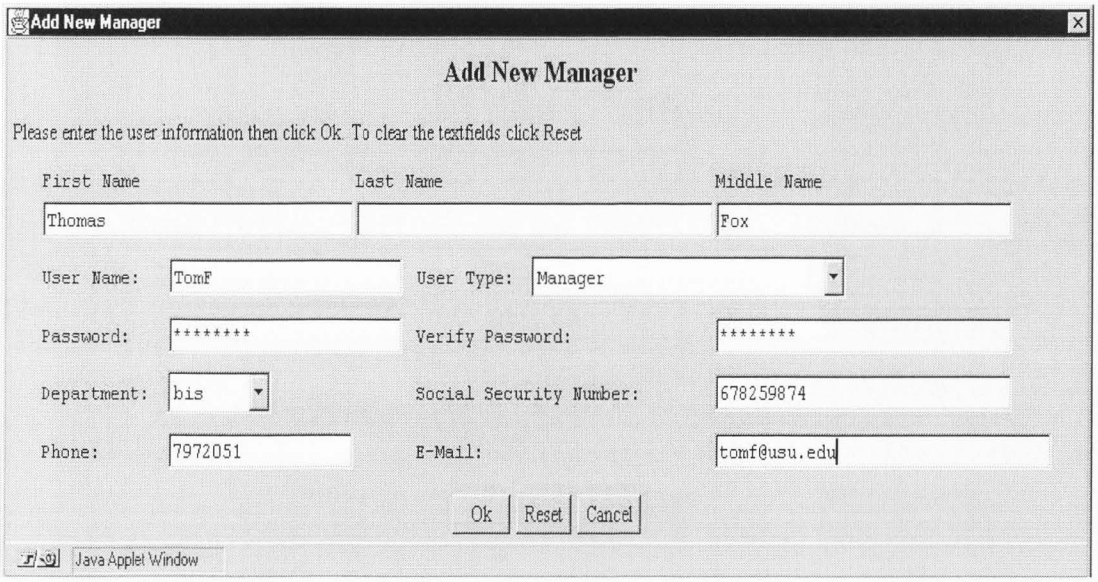
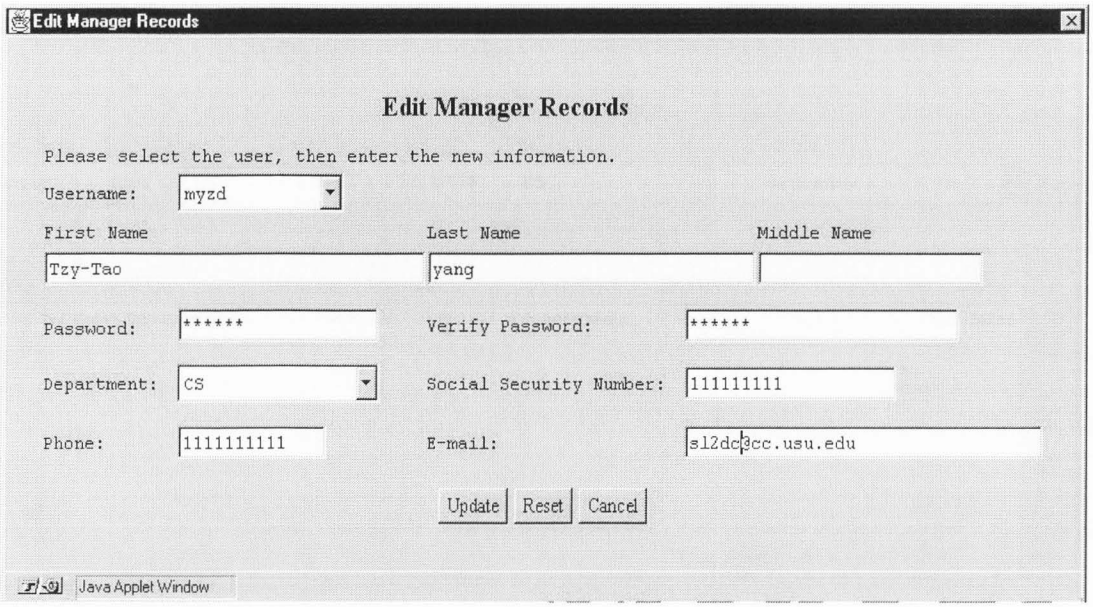


Figure 3-2. The dialog box for adding a new manager

the new user. After all information is entered, the manager needs to click on the *Ok* button to complete the job. The *Reset* button is for clearing all the information in the text fields. The manager can quit the operation by clicking on the *Cancel* button, which will bring the manager back to the manager interface.

b. Edit Existing User Information

If the manager pressed a wrong key during the setup of the user account or the user information has changed, the manager can change the user's personal information by clicking the proper user type from the menu bar, then select *Edit* menu. The interface for editing managers' information is shown in figure 3-3. The box for editing students' or instructors' information is almost identical with figure 3-3. If the manager chooses the user from the *Username* choice button, the current information will show up in the text fields. After the manager has changed the information and clicks on the *Update* button,



the

Figure 3- 3. The dialog box of editing a manager account

new information is saved into the database. The *Reset* button will reload the current information from the database, in case of errors. The *Cancel* button will close this interface and abort the operation when clicked. The department manager can edit the user information in his/her own department, but not the users in other departments.

c. List Current Users

The interface for listing users is similar to the one for editing user information, the differences are that the managers can not change the user information in this interface, and there are no *Update* and *Reset* buttons. If the manager wants to change the user information, he/she needs to use the editing interface. If the manager chooses to list the student information, the students' test records are also displayed. Figure 3-4 shows a sample of a student list. The manager can choose the student from the choice button, or click on the *Previous* or *Next* button to browse through all student records.

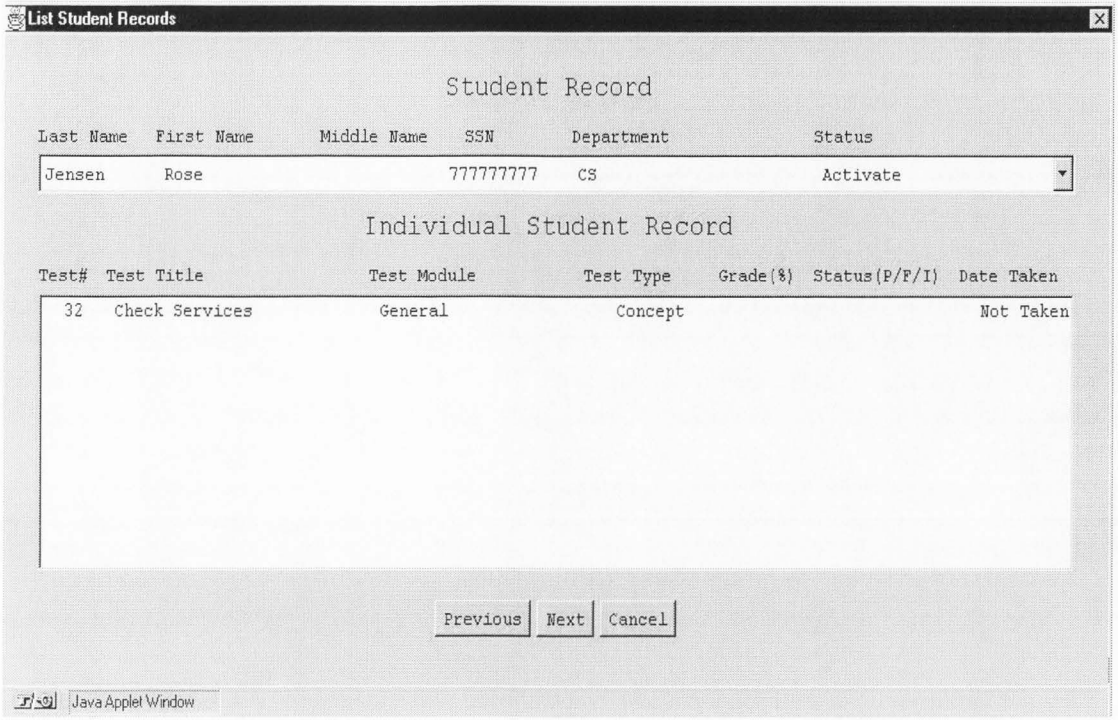


Figure 3- 4. A sample of student record list

d. Disable User Account

This function is designed for cases when a particular user is on vacation or left the organization for a short period of time. For security considerations, the manager might want to disable that user's account. When an account is disabled, the user can not log on to the system. To open the disable interface, simply select the user type from the menu bar then choose *Disable*. Figure 3-5 shows the disable interface for managers. The disable interfaces for instructors and students are similar to the one in figure 3-5. To disable user accounts, select the users from the list then click on the *Disable* button. If no account is available for disabling, the list will be empty. To quit the operation, click on the *Cancel* button. Again, the department manager can only disable users in his/her own department.

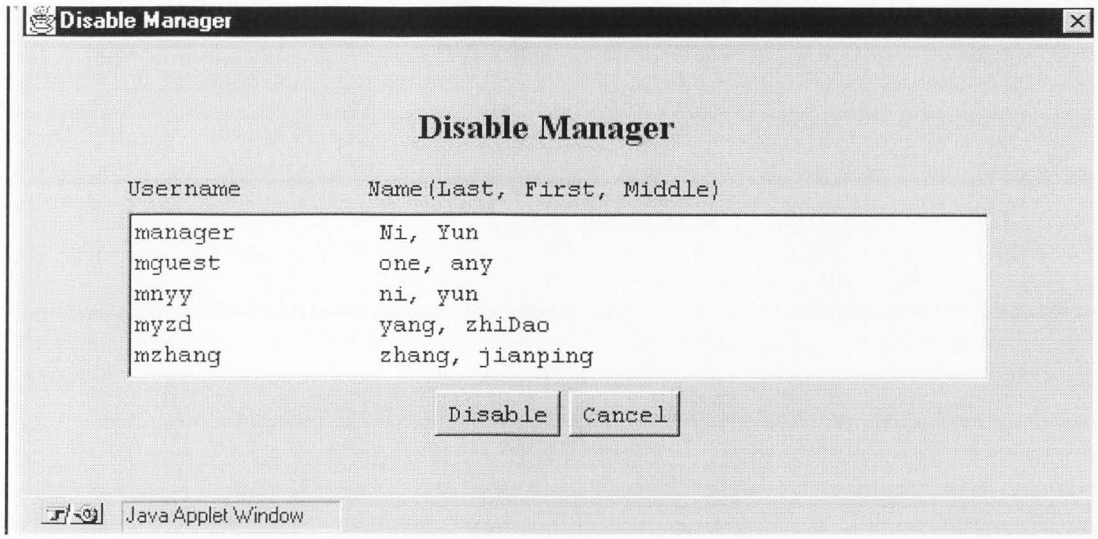


Figure 3- 5. The dialog box for disabling a manager account

e. Enable User Account

This function is opposite from the previous function. When the reason of disabling a user account has ceased, the manager needs to re-enable the user account, so that the user can utilize the system again. This job could be done through the activate-user-account interface. To open the activate-user-account interface, select the proper user type from the menu bar, then click on the *Activate*. This interface for enabling a manager account is shown in figure 3-6. The interface for activating an instructor or student account is similar to the one shown in figure 3-6. To activate a user account, simply select the user in the list then click on the *Activate* button. If no account has been disabled previously, the list will be empty. To close the interface or cancel the job, click on the *Cancel* button.

f. Remove User Account

When a user no longer uses the system, the manager might want to remove the user from the system and remove all the information related to that user from the database.

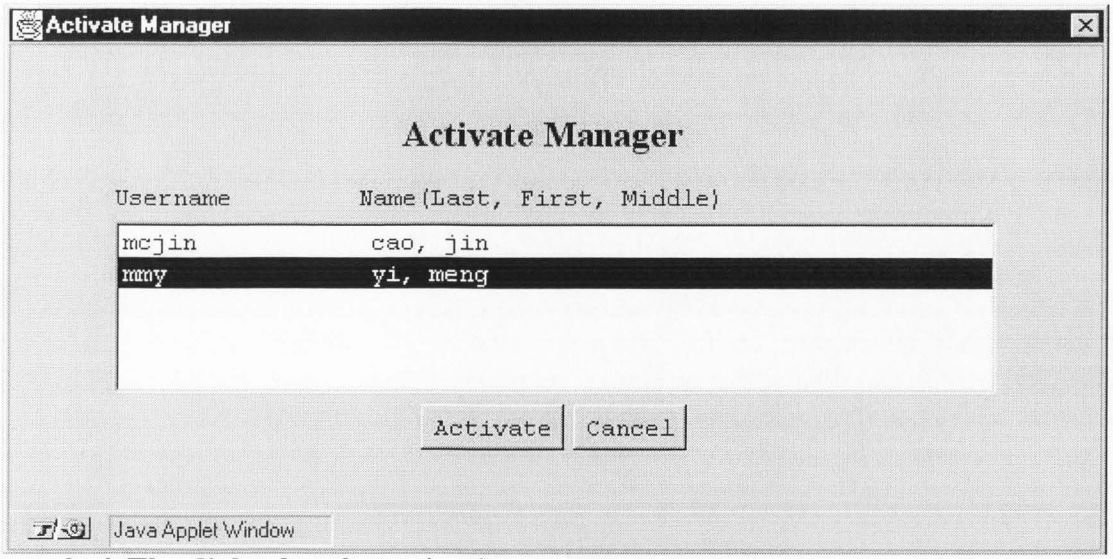


Figure 3- 6. The dialog box for activating a manager account

If the user is a student then all his/her test record will be deleted along with his/her personal information. If the user is an instructor then the instructor's personal information will be deleted but the test(s) that the instructor created will not be deleted. The system will only remove the test ownership from the test entries. Please be cautious, for this operation is not reversible. Once the user account is deleted, to restore the information one must again add the user. From the menu bar select the proper user type and choose the *Remove* item and the remove user interface will be opened. The sample interface for removing a manager account is shown in figure 3-7. The interface for removing a student or instructor account is similar to that in figure 3-7. To remove a user, first select the user from the list then click on the *Remove* button. If no account exists in the database of this particular user type, the list will be empty. The *Cancel* button will close this interface when one clicked on it.

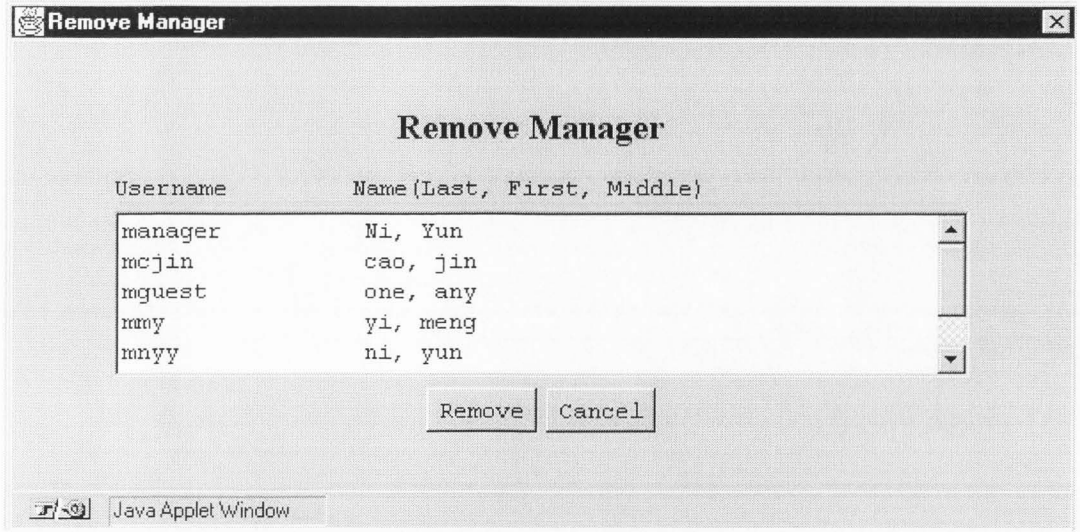


Figure 3- 7. The dialog box for removing a manager account

Test Management

After a test has been created, it is possible that the state of a test be changed occasionally. In this situation, the manager needs to apply the changes to the test. The tasks designed for maintaining the tests include: edit test ownership, unlock test, and update test.

a. Edit Test Ownership

When a test is created, the creator is the default owner. However, there are cases in which a test has more than one owner or a test owner should be removed from the owner list. In these cases, the manager needs to add an existing instructor to the owner list or remove an instructor from the owner list. Figure 3-8 shows the interface to add or remove instructors from a test. This interface shows the test title at the top, its current owner on the left pane, and the remaining instructors on the right pane.

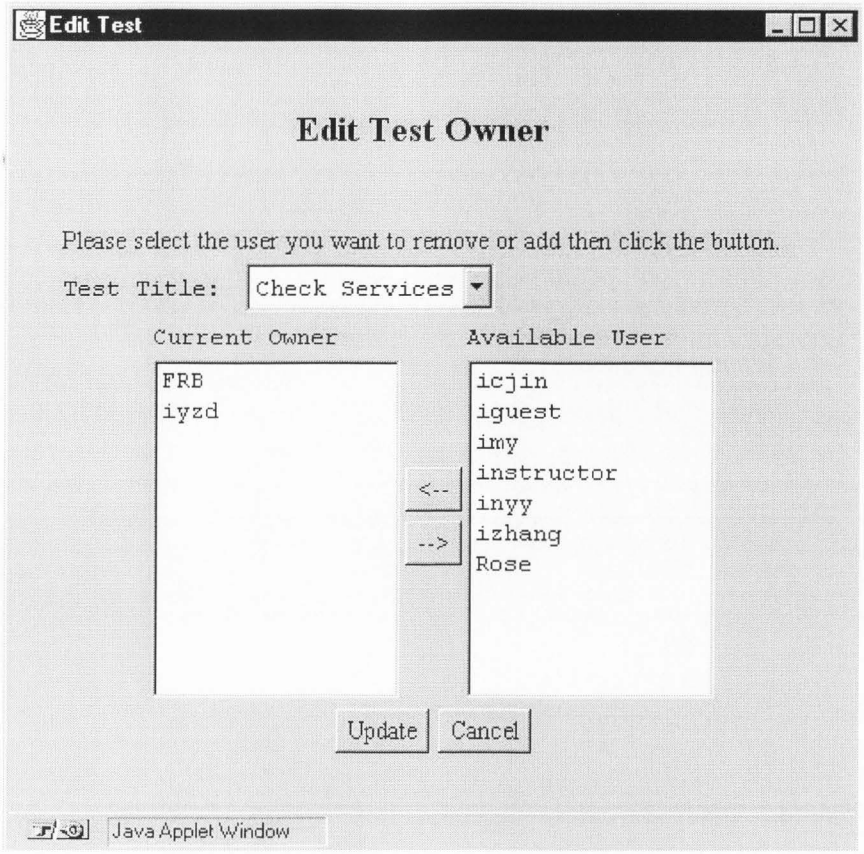


Figure 3- 8. The dialog box for editing a test owner

To add instructors to the owner list simply select instructor(s) from the right pane and click on the left arrow-headed (←) button. To remove instructor(s) from the owner list select the instructor(s) from the left pane and click on the right arrow-headed (→) button. After changes have been made, click on the *Update* button at the bottom will save the changes to the database. Clicking the *Cancel* button will close this interface. The department manager can only modify those tests in his/her department.

b. Unlock Test

Some tests need to be proctored, depending on instructor's requirements. When a student wants to take this kind of tests, he/she needs to go to the manager on the test site

and request the manager to unlock the test. When unlocking a test, the manager needs to specify the IP address and valid time for the test. In this way, the test can only be taken on the machine with the specified IP address within the valid time frame by a particular student (for example machine 1 in lab 407 for student A). Then, the student can only take the test on one particular machine. Figure 3-9 shows the interface to unlock a test.

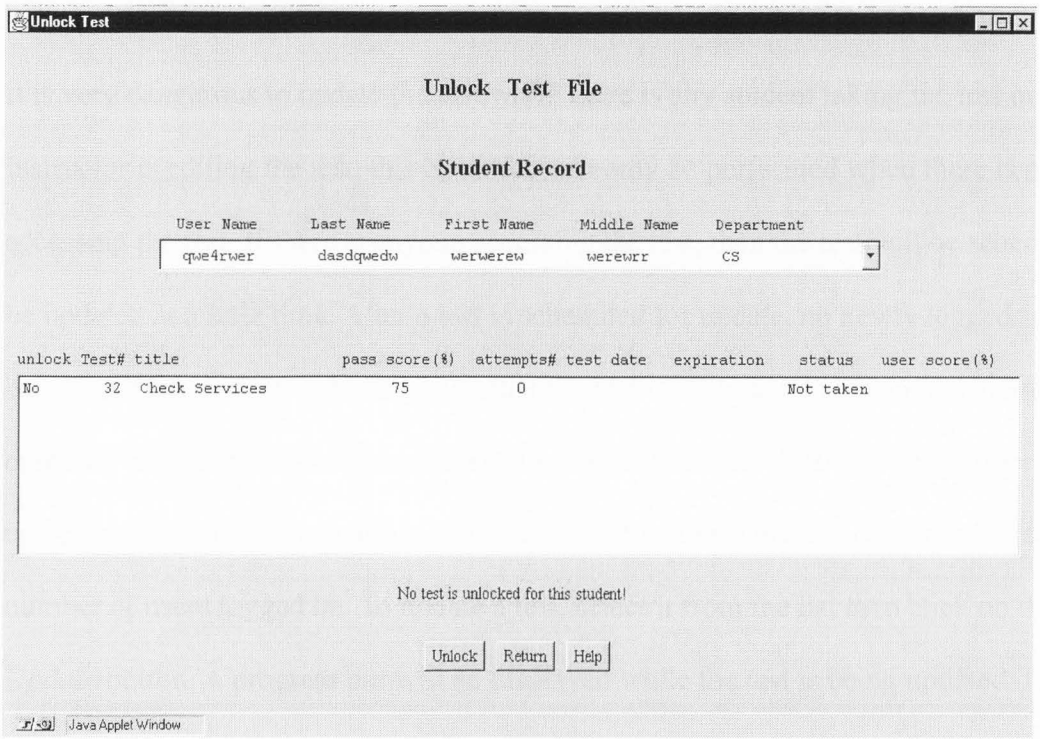


Figure 3- 9. The dialog box for unlocking a test

To unlock a test, first the manager needs to select the student by means of the student record choice button. Double clicking the test or selecting the test from the list then clicking on the *Unlock* button will unlock the test. Another dialog box will appear asking for the IP address and valid time. After entering the information and clicking on the *Ok* button the test is unlocked. The student then needs to go to the specified machine

to take the test within the valid time frame specified. If the student did not take the test within the valid time frame, the test is locked again. To effectively take the test, the student needs to go through the same process again.

c. Update Test

When a test is created or modified, it is saved in the instructors' database. The student will not be able to see the new test or the newer version until the test is updated. Because it is very dangerous to update the test while there is any student taking the test or any instructor is editing the test, this operation can only be performed when there is no user accessing the test. If there is anyone accessing the test, then the test will be scheduled to be updated at a later time. After a test is scheduled for update, no newly logged in user can access that test, in order to assure that the test will be updated when those users currently accessing the test have finished their work. Figure 3-10 shows the interface for updating a test. The information shown in the list includes the test title and the current number of users logged on. To update a test, select it from the list then click on the *Update* button. A progress bar will be displayed while the test is being updated. If there is any test that can not be updated at the moment, the *NetTest* will schedule the update process and inform the manager. To close, simply click on the *Cancel* button.

Department Management

The department management tasks include list, add, edit, and remove department. The system manager can perform all these functions, while department managers can only list and edit his/her own department.

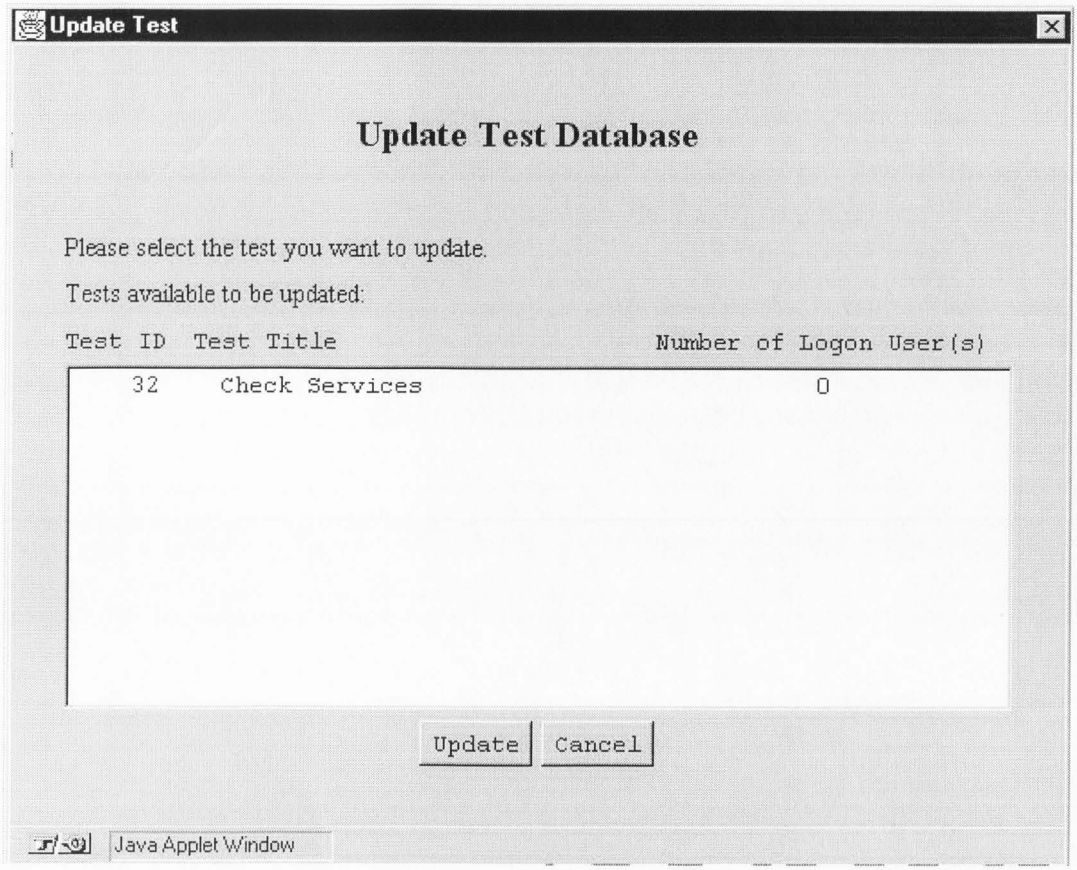


Figure 3- 10. The dialog box for updating a test

a. List Department

This function allows the manager to list all departments already created after the *NetTest* is installed. The manager will not be able to change the information listed in this interface. To change the information, the manager needs to use the *Edit Department* function.

b. Add Department

Only system managers can use this function. It allows the system manager to add a new department to the system and assign test(s) to the new department from existing tests. Figure 3-11 shows the interface for adding a new department to the system.

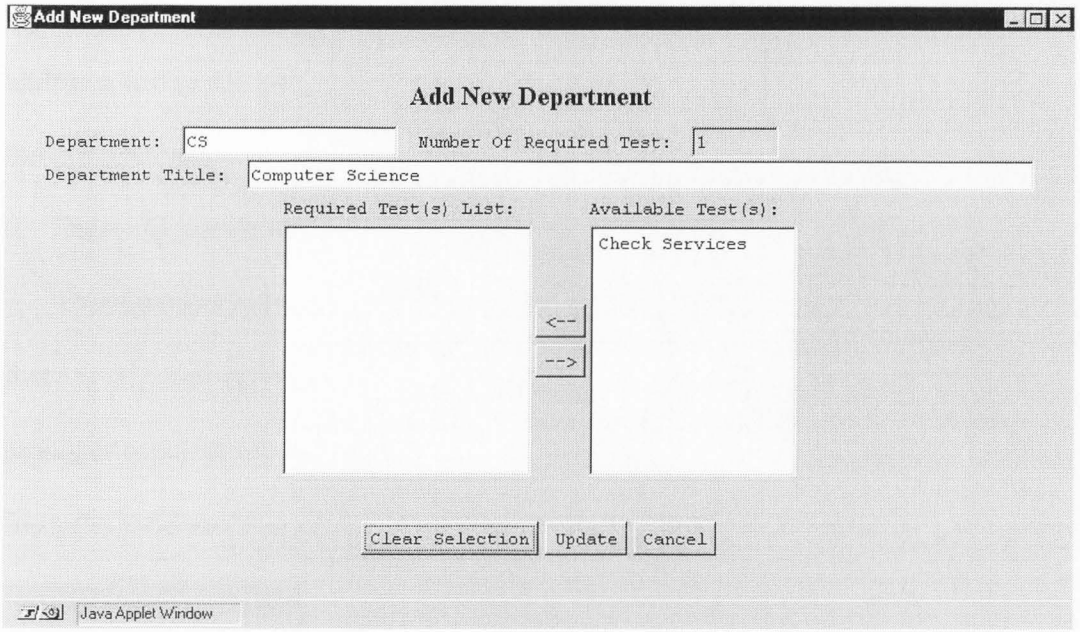


Figure 3- 11. The dialog box for adding a new department

The *Department* field at the top is for its abbreviation. The *Department Title* field is for the full name of the department. The left pane showing the test list is initially empty. The right pane lists all tests available in the system. To add available test(s) to the test list, select test(s) from the right pane then click on the left arrow-headed (←) button between the two panes. To remove any test from the test list, select the test from the left pane then click on the right arrow-headed (→) button. To clear the test selection, click on the *Clear Selection* button. To save the new department information into the database, click on the *Update* button. To close the interface, click on the *Cancel* button.

c. Edit Department

Through this function, the system manager can change the test lists of an existing department. Notice that changes made here will affect the department immediately. The interface for editing the department is similar to the interface shown in figure 3-11. The

only difference is that the department field is a choice and is not editable. The method of adding a test to the test list or removing a test from the test list is the same as described in the previous section.

d. Delete Department

This function is for deleting an existing department from the database. However, if there is any user belonging to this department, the system will not allow the deletion. The manager needs to remove all users in this department first, then to delete the department from the database. Figure 3-12 shows the interface for deleting a department.

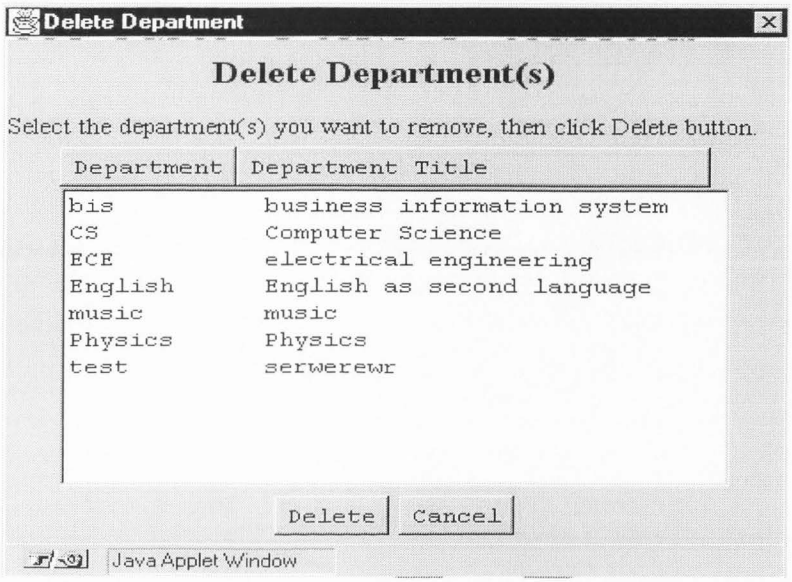


Figure 3- 12. The dialog box for deleting a department

The department abbreviation and its title are shown in the list. To delete any of the listed departments, select the department from the list, then click on the *Delete* button. To close the interface, click on the *Cancel* button.

Generate Test Reports

Currently, there are three different types of reports that can be created within the system, they are attempt reports, statistics reports, and summary reports.

a. Attempt Reports

In this type of reports the manager can see the results of tests taken from each department. The listed information includes the test title, the number of times that the test is passed, failed, and incomplete. This report will provide the information of how each test is taken by the students. A sample of attempt report is shown in figure 3-13.

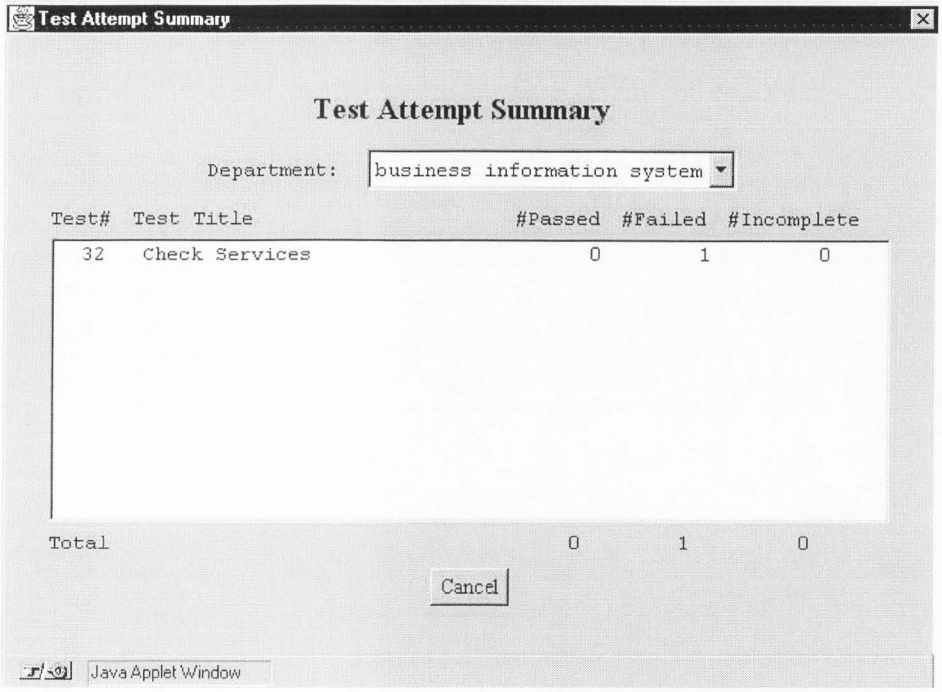


Figure 3- 13. A sample of test attempt report

b. Statistics Report:

The test statistics can be listed (1) *by all tests*, (2) *by tests taken from each department*, or (3) *by questions* in a test.

1. By All Tests

The report lists the statistics of *incomplete*, *passed*, and *failed* numbers of each test on the first, second, and third attempt. This report also shows how well the students have done on each test. If the passed number on the first attempt is high then the test might be too easy for the students or the students have learned very well. If the passed number become higher in the later attempt then the students have improved during the learning process. A sample statistics by all tests is shown in figure 3-14.

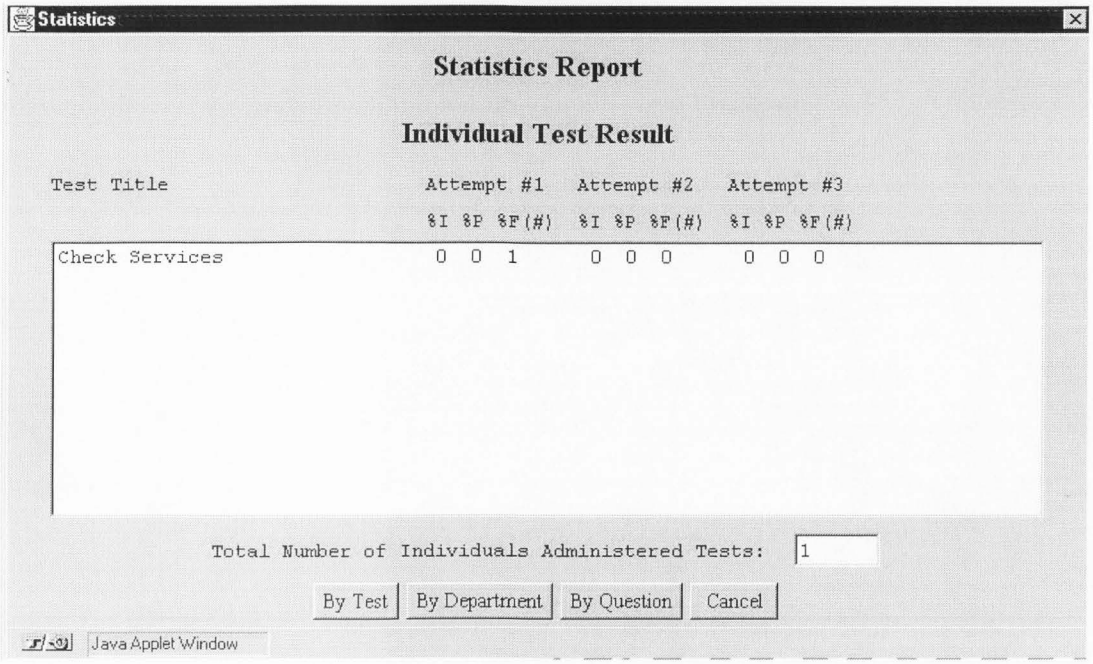


Figure 3- 14. A sample test statistics report listed by test

2. By Department

The report is similar to the previous report, but it filters out those tests not required by the department that the manager has chosen to display. To execute the function, select the department from the choices, and the statistics of tests belonging to the department will

be listed. A sample test statistics listed by the department is shown in figure 3-15.

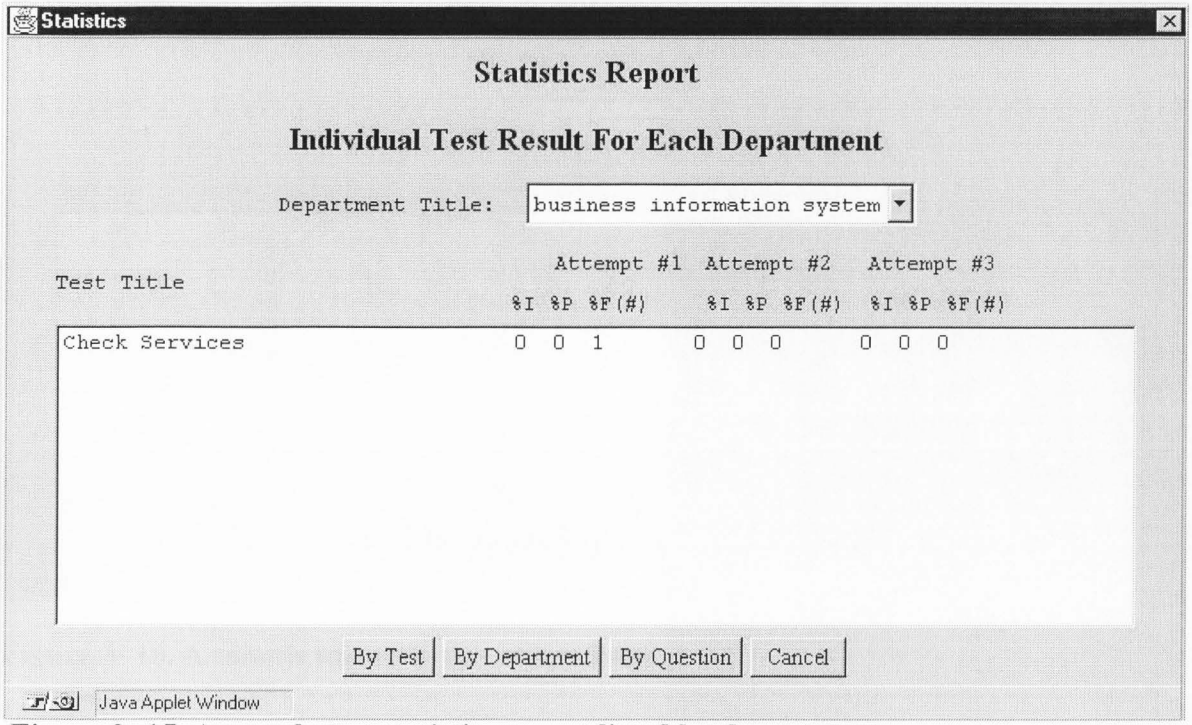


Figure 3- 15. A sample test statistics report listed by department

3. By Questions In A Test

This report displays how well each question was answered by the students. It lists all the questions of the test, and the number of times each question is answered correctly and incorrectly. Note that because of the random manner in which questions are selected individual question will not have the same attempt total. This report allows the manager to go into more detail to see the performance of students on the basis of individual questions. If in most instances a question is answered incorrectly, the instructor might want to change the question or disable the question. A sample test statistics report listed by test is shown in figure 3-16. To view the statistics of another test, select the test from the choice at the top.

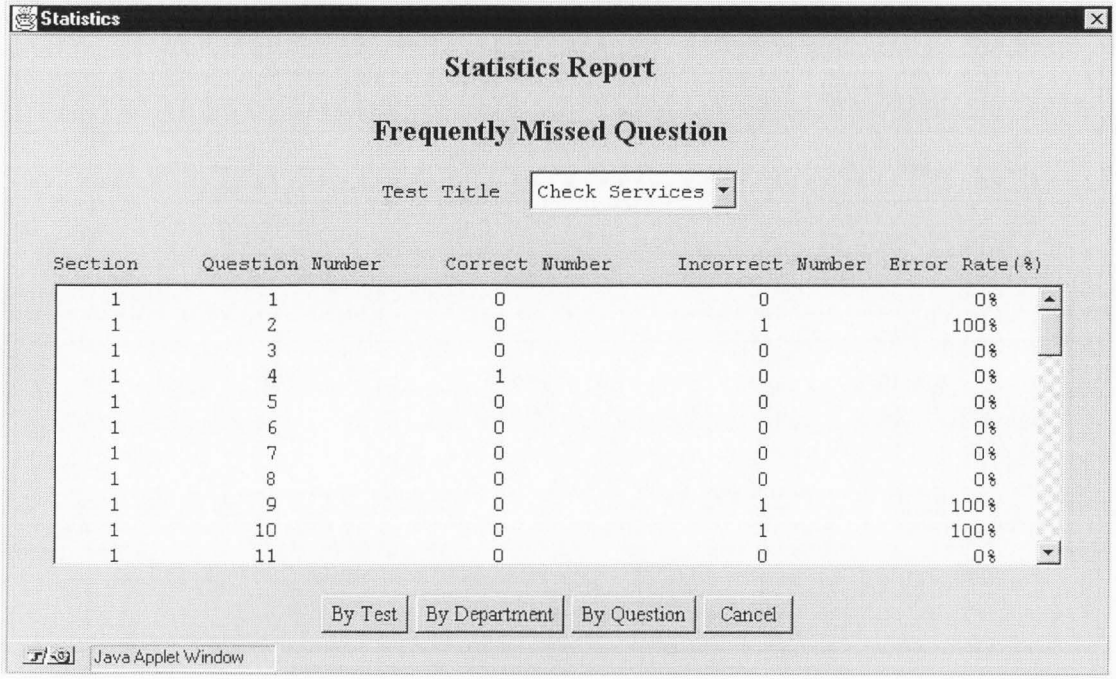


Figure 3- 16. A sample test statistics report listed by questions

c. Summary

The summary report lists the departments and all the tests required, along with the number of failed tests on the first and second attempt and the total of attempts. It also lists the total passed number and the required passing score. A sample is listed in figure 3-17. To view the statistics of another department, select the department from the choice at the top.

Other Tasks

Other miscellaneous functions performed by the manager are: clear database, clear student record, and view log files.

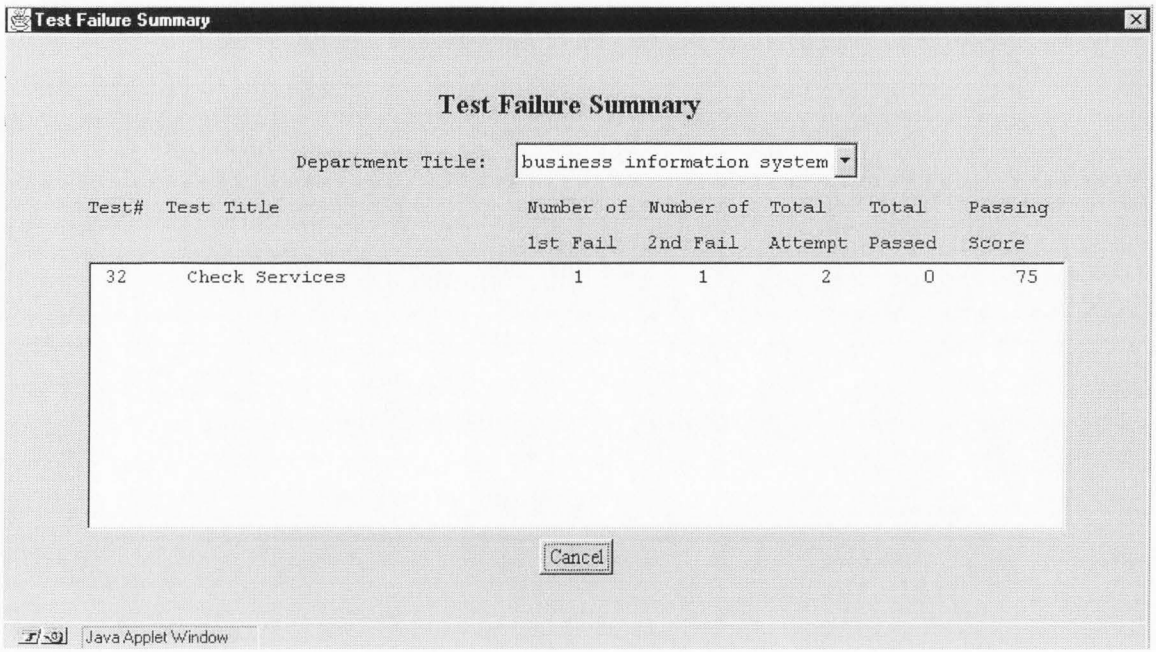


Figure 3- 17. A sample test summary

a. Clear Database

There are times that the tables in the database is corrupted or all the records in the table need to be cleared but we still want to maintain the structure of the table. This function will let the manager clear all the records in the selected table but maintain the table definition intact. Only a system manager can perform this operation.

b. Clear Student Record

The system will keep a record every time the student takes a test, until that test expires. In a large organization, the size of the database could increase dramatically and occupy the precious disk space. For this reason, the manager can choose whether to delete all test records of a selected student or keep the most recent test records. The department manager can only delete those students within his/her department.

c. View Log Files

There are two log files maintained by the system. One is the user logon log; the other is the update log. The *log server* maintains the user logon log, while the update log is maintained by the *update daemon*. These two programs were discussed in chapter 2 section 5 page 16. This function will pop up a window and let the managers choose which log they want to see.

STUDENTS' USER MANUAL

The primary objective of students in *NetTest* is to take tests. Also, students can view their test records and change their password. The questions in a generated test are randomly selected from the test database. The order of the questions that appear in the test is also randomly generated. Furthermore, the order of the choices is also randomly generated. After the test is done, the system will grade the test automatically, except the essay questions. At this point, students have the chance to see the results immediately and the correct answer for each question that appeared in the test. The tasks that a student can perform are discussed below:

Change Password

The students can change their password after logging on to the system. When a student decide to do so, the system will ask him/her to type in the old password, the new password, and verify the new password, so that the student will remember the password in the future and to assure that the new password is what he/she wants. The interface is shown in figure 3-18. To save the new password to the database, click on the *Change*

button; otherwise, click on the *Cancel* button to close the interface.

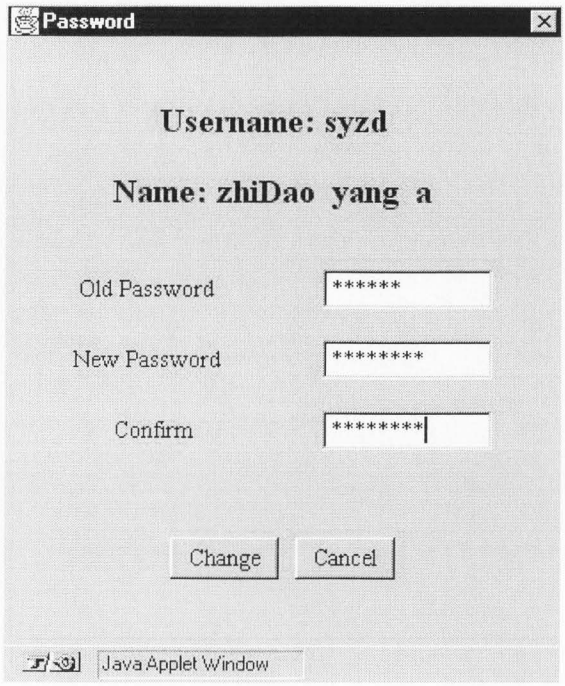


Figure 3- 18. The dialog box for changing a student password

View Personal Record

This function displays all the tests that a student is required to take, the dates those tests were taken, the expiration dates of the tests, and the results of the tests. In this manner, a student can keep track of his/her performance in the tests, and to become aware of the tests they need to take in the future. The interface for *view personal record* is similar to that for taking a test (figure 3-19).

Take A Test

Students are able to see those tests required within their department, then decide which test they want to take. For those tests that need to be proctored, the students must request the manager to unlock the test they want to take. For those tests that do not need

unlocking, the students can take the test anywhere and at any time they want by simply selecting it. The interface for choosing a test to take is shown in figure 3-19.

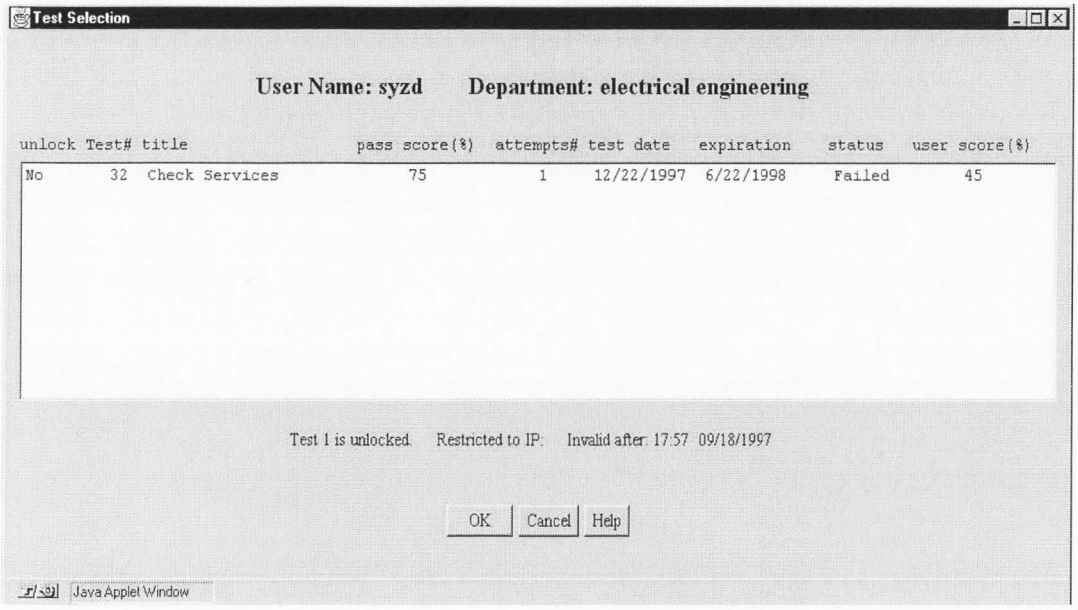


Figure 3- 19. The dialog box for choosing a test to take

After a test has been chosen and unlocked another window will appear, called *test frame*. A sample of *test frame* is shown in figure 3-20. During the test the student does not need to answer the question in a particular order. The student can browse through the questions or jump to a question by clicking on the *Previous*, *Next*, *First*, *Last*, or *Goto* button. They can skip a question, and answer it later. They can also browse through the answered or unanswered questions by clicking the *Answered* or *Unanswered* button at the bottom. They can also view the status (shown in figure 3-21) of all questions to see whether the question is answered or not, and they can directly go to any chosen question. Some sample questions are shown from figure 3-22 through 3-24.

All the test sessions have their time duration measured. However, some tests may not

have a time limit so that the students can take as long as they need. Some tests have a

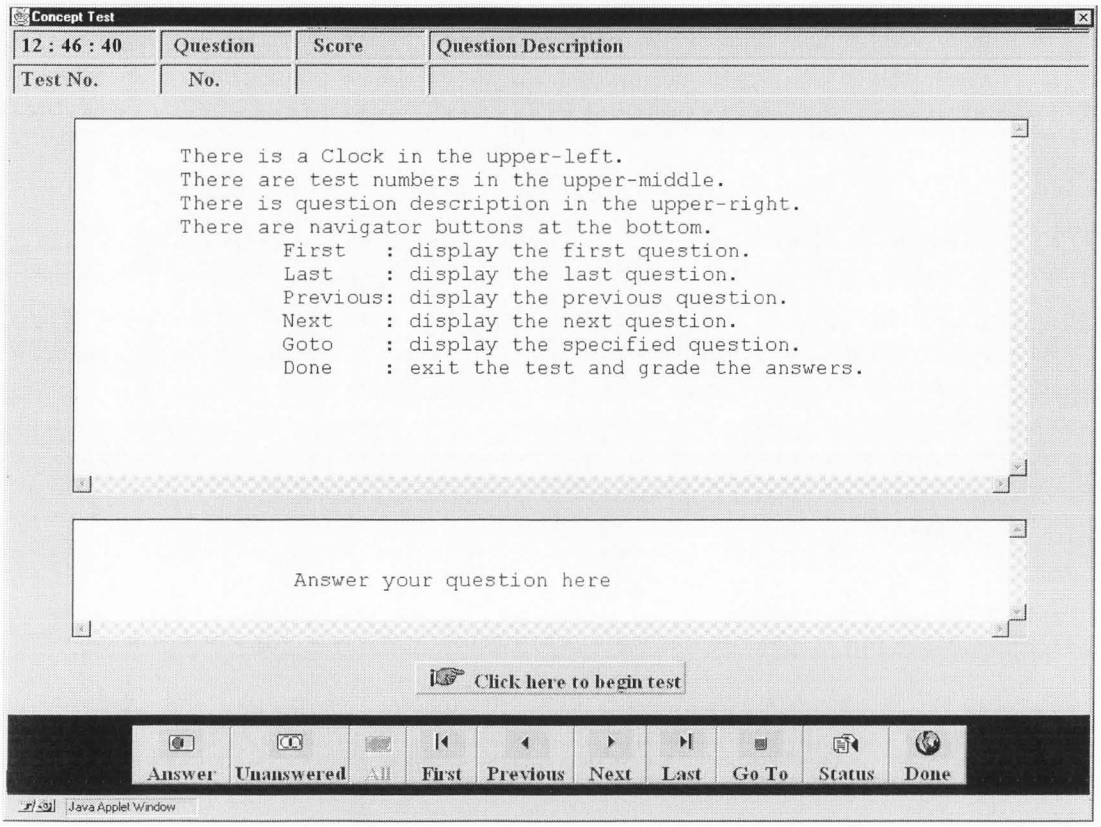


Figure 3- 20. The student test frame

time limit, so that when the specified time has elapsed or the student has finished the test, the system will stop the test session immediately and start grading the test. This procedure assures that every student will have the exact same amount of time to answer the questions. After the test has been graded, the system will display the grade (shown in figure 3-25) and provide the chance for the student to review the questions and their correct answers. Two samples of reviewing sessions are shown in figure 3-26 and figure 3-27. The test results are kept in the database every time the student takes a test.

Once again, all the questions in the test are randomly chosen, and the choices in the

questions are also randomly ordered. Even if more than one student is taking the same test at the same time and place, the questions they get will appear differently.

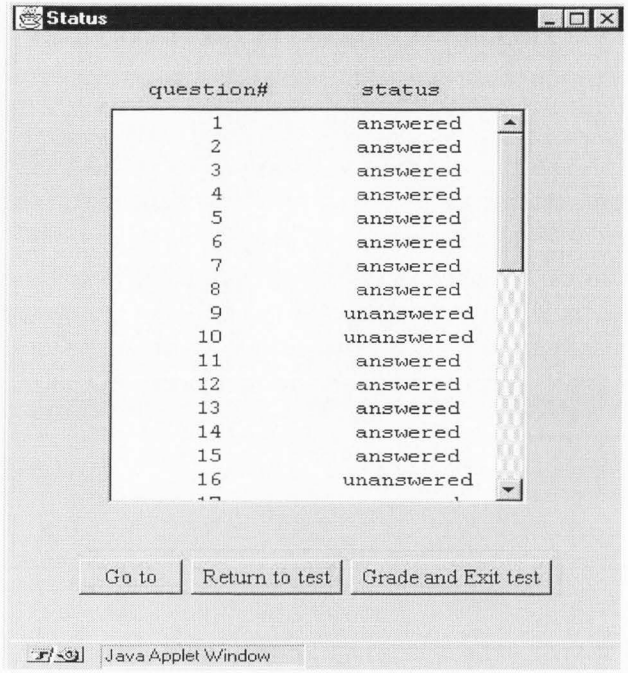


Figure 3- 21. A sample of test status when taking a test

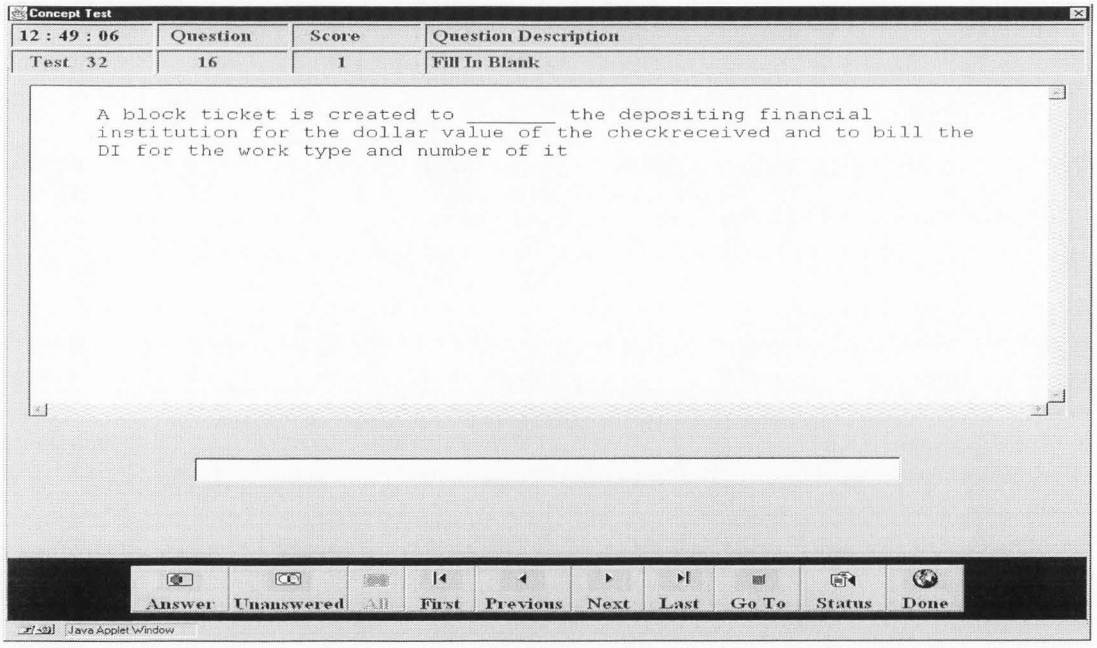


Figure 3- 22. A sample of fill-in-blank question

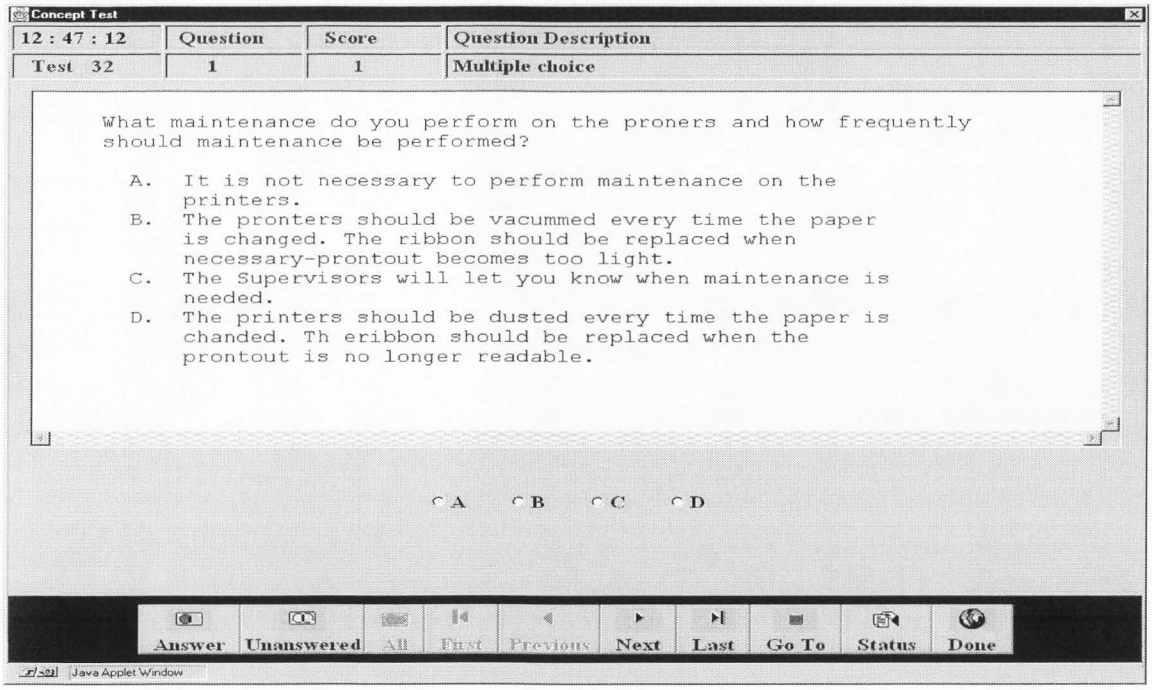


Figure 3- 23. A sample of multiple choice question

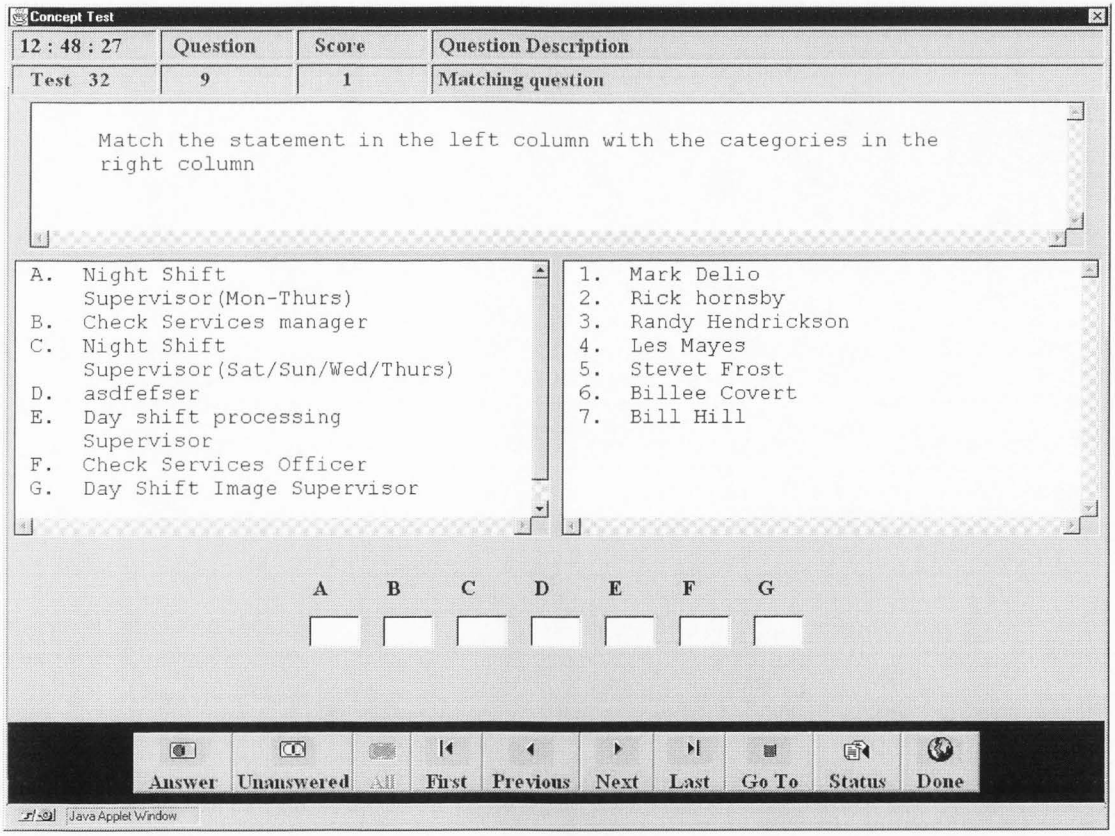


Figure 3- 24. A sample of matching question

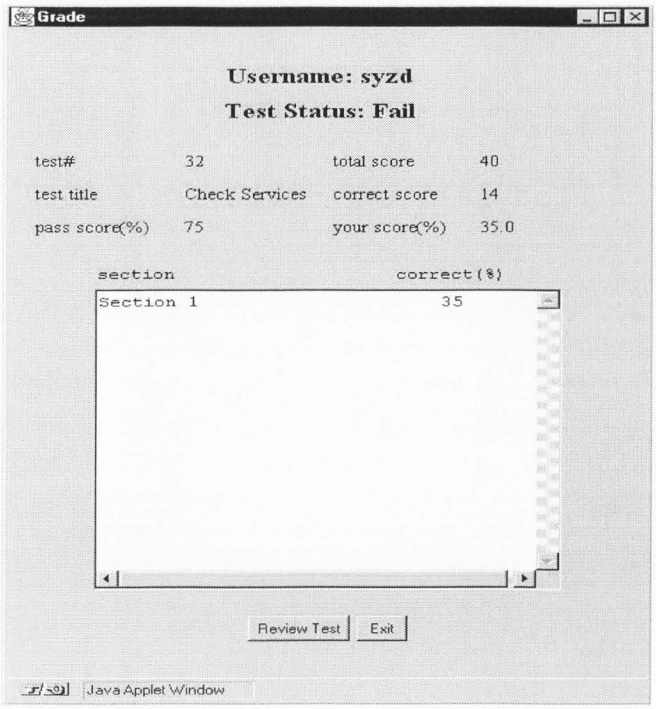


Figure 3- 25. A sample of test grade dialog box

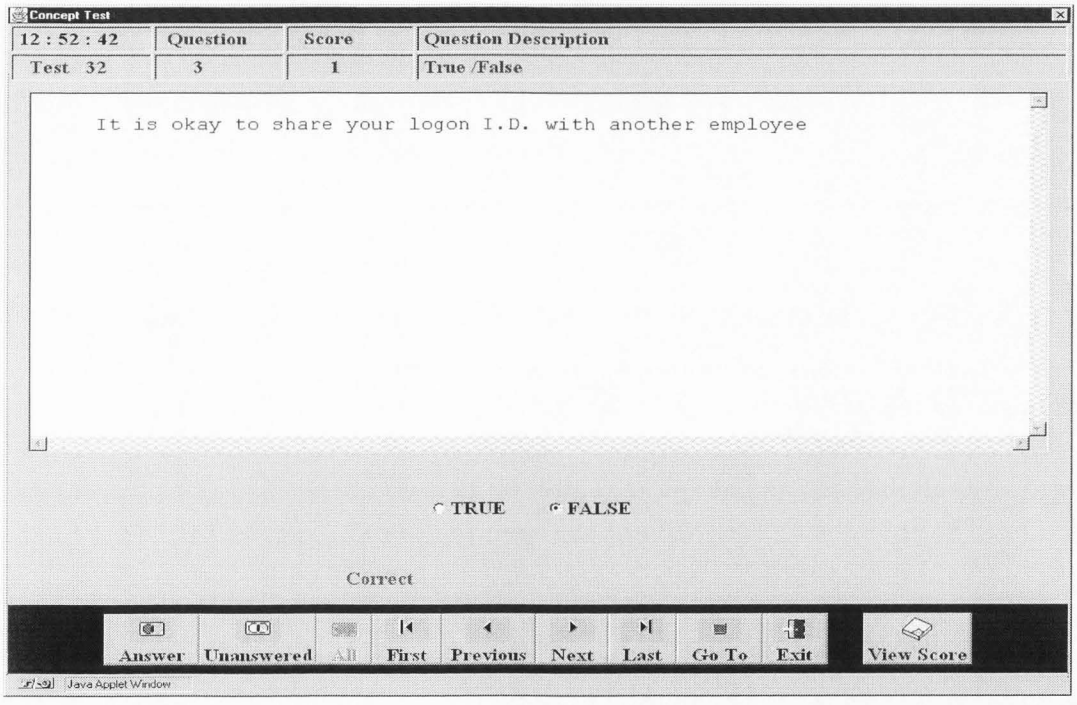


Figure 3- 26. A sample of question answered correctly in a review session

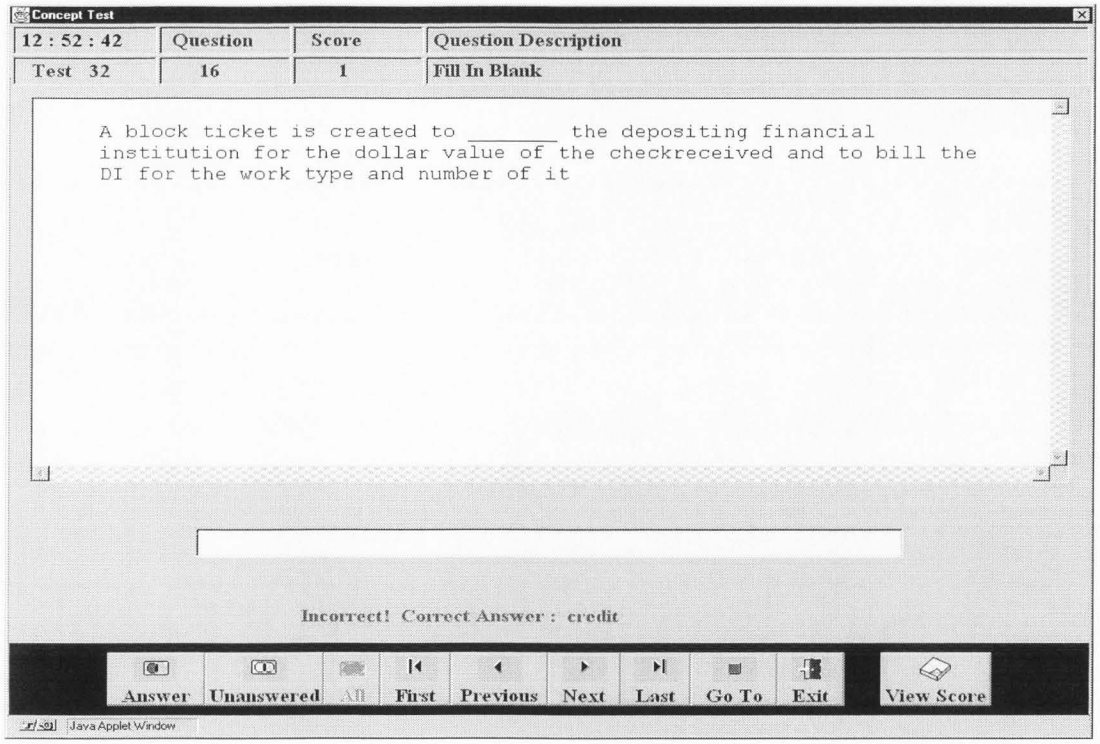


Figure 3- 27. A sample of question answered incorrectly in a review session

INSTRUCTORS' USER MANUAL

All the tasks performed by the instructors are related to the tests. The instructor can only access those tests of which he/she has the ownership. When an instructor create a test, he/she is the only owner by default. However the manager could change the ownership later. It is possible that one test have more than one owner. For security reasons, only one instructor can open the test at any given time.

The primary entrance for the instructor of the system is the instructor frame. Inside this interface the instructor is able to create a new test, open an existing test, update a test, delete a test, grade a test manually, view the test statistics, view the student grade, change the student grade, and edit the information related to a test. If the instructor chooses to create a new test or open an existing test, the system will pop up another window, which

is called *authoring interface*. I will discuss the authoring interface in a later section. Let me discuss the functions inside an *instructor interface*.

INSTRUCTOR INTERFACE

Create A New Test

When creating a new test the instructor needs to tell the system the name of the test, the test *time limit*, and the time validity of the test by typing the information into the proper fields. Other information is optional and can be left blank. By default, the time limit is 50 minutes. If there is no time limit on the test, the instructor should type in a 0. The *test expiration time* means the effective time period of a test, in other words, the test results will expire after that time has elapsed upon taking the test. The validity period is of 6 months by default. A sample interface for creating a new test is shown in figure 3-28.

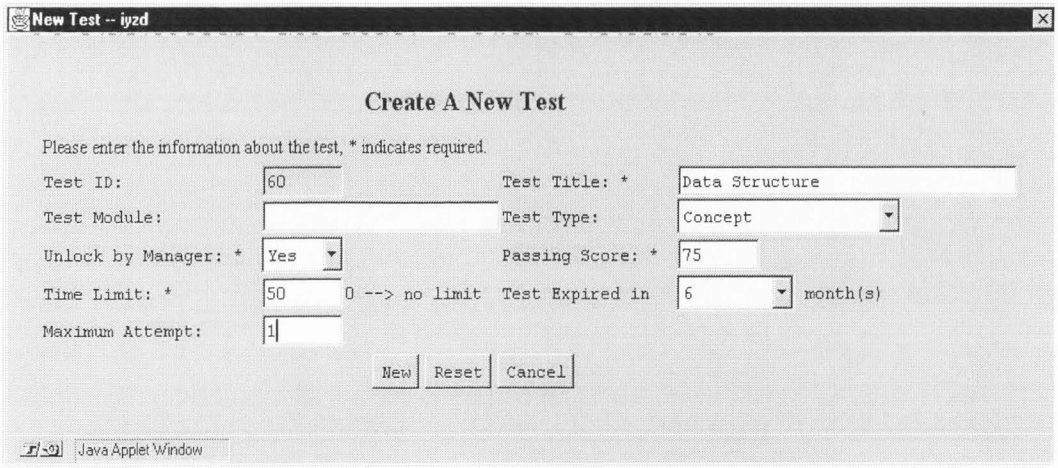


Figure 3- 28. The dialog box for creating a new test

After the instructor click on the *New* button, the test will be created. But the test will not be saved to the database until the instructor saves the test. After the test is created, the

instructor is free to create new sections or questions.

Open A Test

When the instructor is trying to open a test, the system will only show those tests of which the instructor has ownership. This will prevent other instructors, who have no privileges on the test, from modifying it. When the test is opened, the instructor can add new sections, or questions, delete questions, modify existing questions, or change properties of the test, sections, or questions. To open a test, select the test from the list, then click on *Open* button or simply double-click on the test.

Update A Test

Whenever a test information is changed, the test database that the student used will not show the changes. Only when the instructor or manager updates the test will the students be able to use the new version. The instructor can only direct the system to update those tests of which the instructor has ownership. To update test(s), select the test(s) the instructor wants to update from the list, then click on the *Update* button. If there is any other user accessing the test, the test will be scheduled for later updating and the system will display a message to the instructor.

Delete A Test

When a test is no longer valid, the instructor can delete it. After the test is deleted, all information related to the test is removed from the database, including the sections and the questions, the student records related to it, and the test list of every department. This operation can not be undone to restore the test. To delete a test, select the test in the list then click on the *Delete* button.

View Test Statistics

The statistics shown in this interface is the statistics of each question in every test. The information includes section number, question number, question type, the number of correct answers correctly, the number of incorrect answers, and the number of times not answered by students. A sample is shown in figure 3-29. With this information, the instructor can see how each question was answered by students. If the number of incorrect answers is much higher than the correct answers, it might indicate that the question is too difficult or the question is confusing to the students.

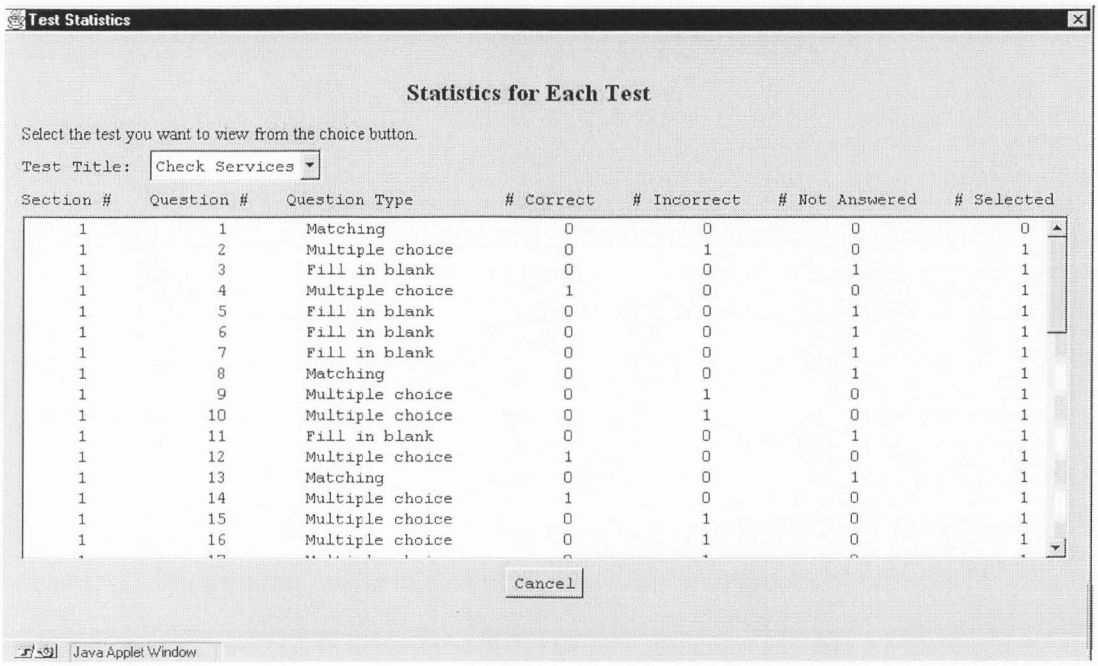


Figure 3- 29. A sample of test statistics

View Student Grade

The instructor can see the grade of all the students who take the test. The instructor can only see those test results of which he/she has ownership. A sample is shown in

figure 3-30. The information available in this interface includes the student name, the number of attempts, and the score. If the student has taken the test more than once, then there are multiple entries in the list. When one clicks on one of the buttons on top of the list, the system will sort the list in ascending order corresponding to the button chosen. For example, if one clicks on the *Grade* button, the list will be presented from the lowest to the highest grade. By default, the list is sorted by student name in ascending order.

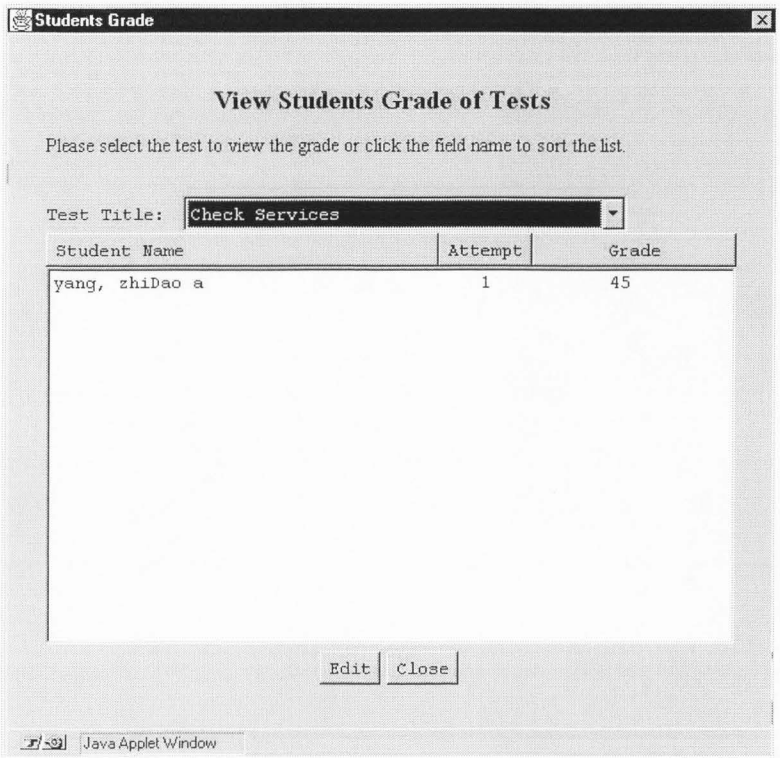


Figure 3- 30. A sample of student grade

Edit Student Grade

When the grade of a student is changed, the instructor needs to use this function to record the changes. This interface is similar to the previous one. However, if the instructor double-clicks on the student grade, the system will display another dialog box

(shown in figure 3-31) to let the instructor change that student's grade. To change the grade, simply enter the new grade then click on the *Ok* button.

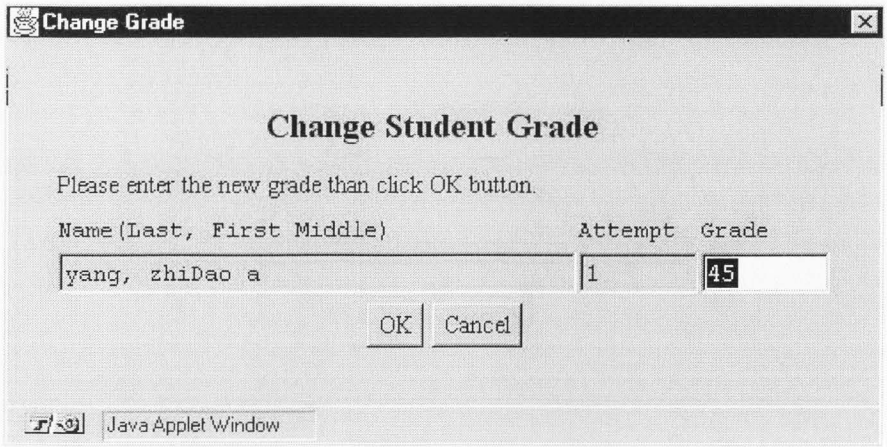


Figure 3- 31. The dialog box for changing a student grade

Edit Test Information

When the instructor wants to change the information of a test, such as the test title, time limit, and section information, he/she does not need to open the test. However, if the instructor wants to edit the question, the instructor needs to open the test first. The interface for changing test information is shown in figure 3-32. To change the section information, double-click on the section in the list, and another dialog box will appear. To save the changes, click on the *Change* button, and the changes will be saved into the database.

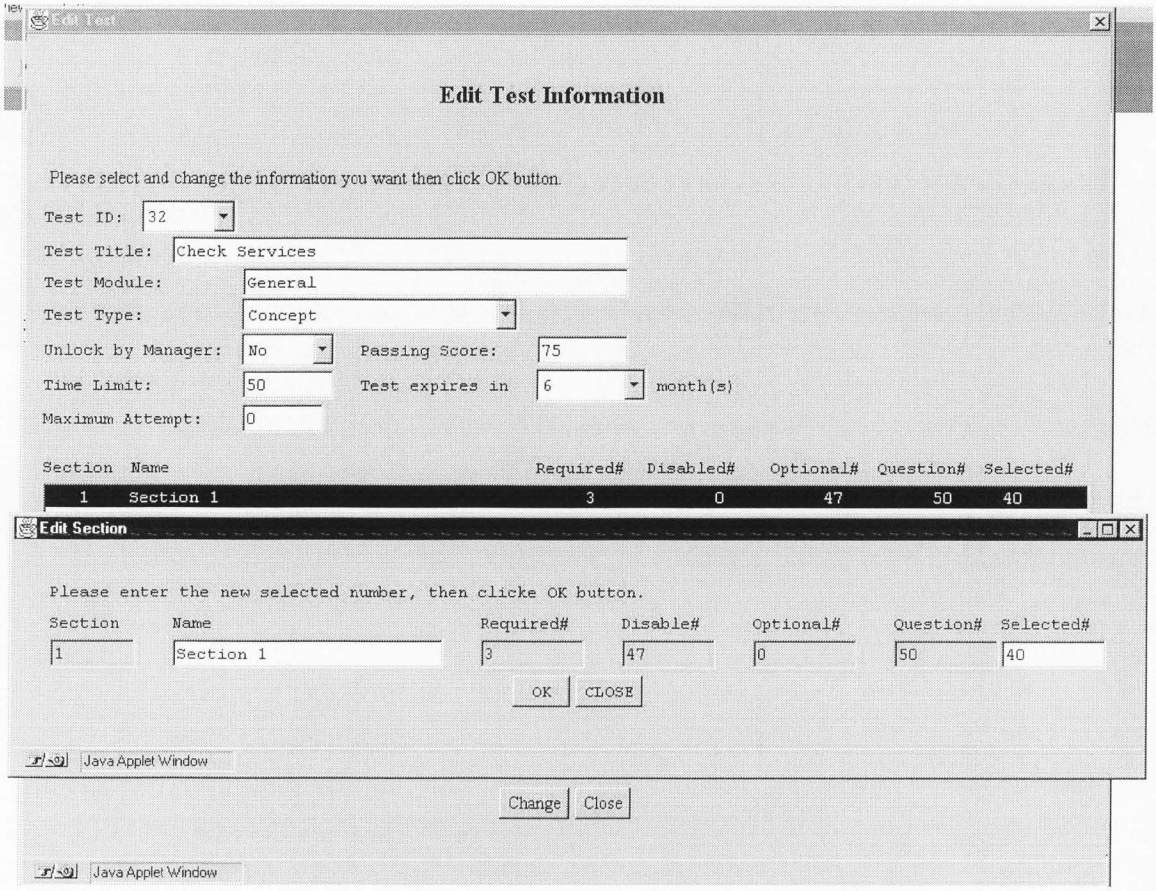


Figure 3- 32. A sample of editing test information in an instructor interface

AUTHORING INTERFACE

The authoring interface will be displayed only when the instructor is creating a new test or opening an existing test. The tasks within the authoring interface are specifically related to the test opened or created. These tasks are sub-categorized into tasks related to the opened test, the sections, and the questions. The tasks related to the opened test include: edit test, save test, rename test, delete test, update test, and edit preference. The tasks related to sections include: edit section, create new section, and rename section. The

tasks related to the questions include: question navigation (i.e. next, previous, first, last, and go to), edit question score, create new question, and delete question.

Edit Test

The instructor will be able to change the information of an opened test by choosing *Edit* from the *Test* menu. An example is shown in figure 3-33. The information includes: test title, test module, test type, unlock status, passing score, time limit, expiration time period, and attempt limit. Clicking on the *Reset* button will reload the test information prior to the change. To save the changes, click on the *Change* button. The information changed will not be written to the database until the test is saved.

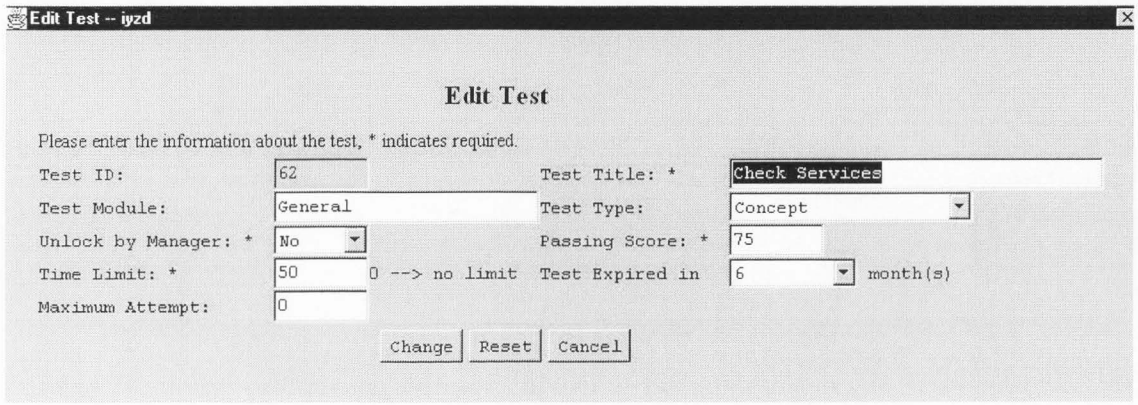


Figure 3- 33. An example of editing test information in an authoring interface

Save Test

During the authoring session, the instructor can choose the *Save* menu item from the *Test* menu to save the test to the database. The instructor can save the test as frequently as he/she wants to assure that the changes are saved in the database, in case of system lock up or crash. Another alternative is to enable the auto-save feature in the system. When

saving a test, the system will check whether the selected number of all sections is greater the total number of questions in that section. If it is not true, the system will display another dialog box to let the instructor change the selected number, because it is impossible to select 10 questions from 8 questions, for example. If the selected number is still greater than the total number of question, the test will be saved as *incomplete*. When a test status is incomplete, the system will not update the test.

Rename Test

Renaming the opened test means changing the title of the test. Two text fields appear in the dialog. The top one labeled *From* is the original test title, the bottom one labeled *To* is for the new test title. The new test title will not be saved to the database until the test is saved.

Delete Test

Deleting a test here is the same as deleting a test in the instructor interface. All the information related to the test would be removed from the database as soon as the instructor clicked on the *Delete* button and this operation is irreversible.

Update Test

The student will not see the new test or the changes when a test is created or changed until it is updated. When updating a test, the system will first check whether there are other users accessing the test. If there is nobody accessing the test, it will be updated immediately. Otherwise, it will be scheduled for updating as soon as there is nobody accessing the test. When a test is scheduled for updating, the test is locked, but the current users can continue their tasks until finished. This will assure that the test will be updated when all current users finish their jobs.

Edit Preference

Currently, the instructor can set the preference of *auto-saving*. More features can be added to the system. The default setting of *auto-save* is enabled and set to five minutes intervals. The example is show in figure 3-34. The instructor can set the intervals from one to ten minutes. When *auto-save* is enabled, the system will write the opened test to the database in case of system lock up or crash.

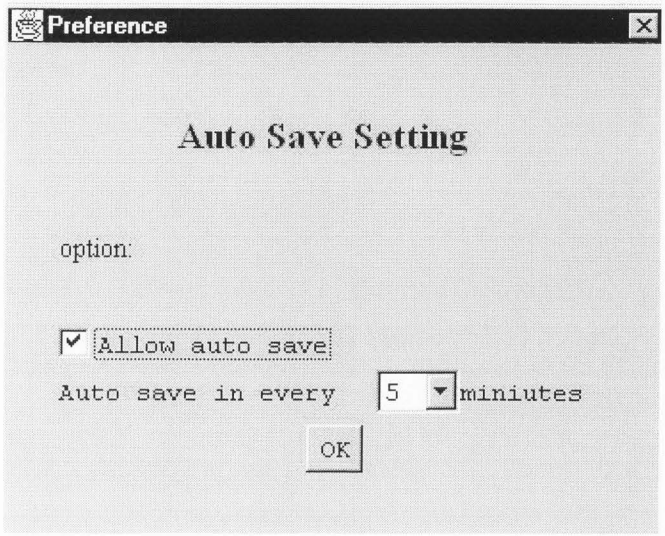


Figure 3- 34. The dialog box of auto save settings

Edit Section

When the instructor clicks on the *Edit* menu item in the *Section* menu, the *edit section* dialog box (shown in figure 3-35) will be displayed. In this dialog box, the section number, section title, the number of required, optional, disabled, and selected question are shown. The instructor can only change the section title, or the selected number. For changing the information of a different section, the instructor simply selects

the different section number from the *Section Number* choice button on the top.

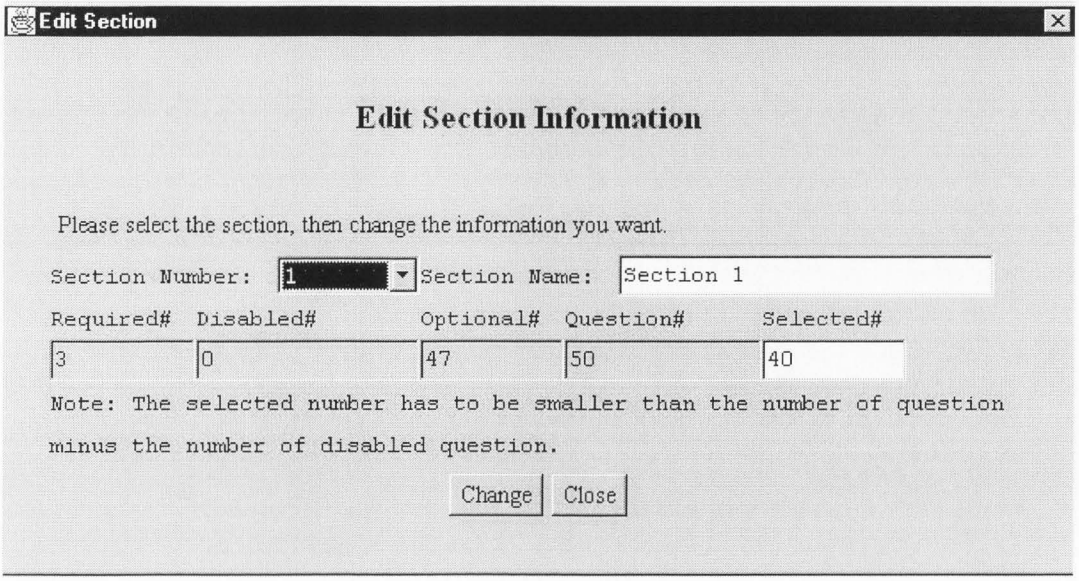


Figure 3- 35. An example of editing section information

Create New Section

When a new section is needed, the instructor can create a new section to host a different group of questions. However, the student will not notice the existence of a different section when taking the test. When creating a new section, its section number is generated by the system as the current section number incremented by one. If this section is inserted, all the later sections will be automatically changed. The instructor needs to type in the section title and selected number, which is the number of questions to be selected when generating the test, in order to create the new section. The section title is optional but the selected number is required. After the instructor clicked on the *New* button, a N*ew Question* dialog will be displayed for adding new questions to the new section. For adding a new question, please refer to a later section.

Rename Section

This function allows the instructor to change the section title(s) of the test. All the sections are listed in a scrolled list. To change the section title, simply double-click the section, type in the new name in the text field, and click on the *Change* button.

Question Navigation

The instructor can navigate through the questions in different sections by choosing from the *Question* menu or click on the navigation buttons (shown in figure 3-36).

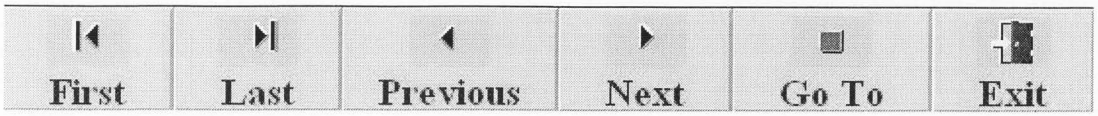


Figure 3- 36. The navigation buttons of an authoring frame

The instructor could either go to the previous, next, first, or last question; or go to any question in the test. When the current question is the first one, the *Previous* and *First* option is disabled. On the other hand, when the current question is the last one, the *Next* and *Last* option is disabled. When the instructor chooses the *Go-To* option, a dialog (shown in figure 3-37) with all the questions will be displayed along with the section number, question number, question type, graphic status, and completion status. This dialog box will let the instructor jump to the question he/she desired, and will show the completion status of each question. The question is not completed when the question text is empty. Other requirement depends on the question type, i.e., there must have at least two choices to constitute a multiple choice question. If any of the questions is not completed, the status of the test is *incomplete*. Thus, it can not be updated.

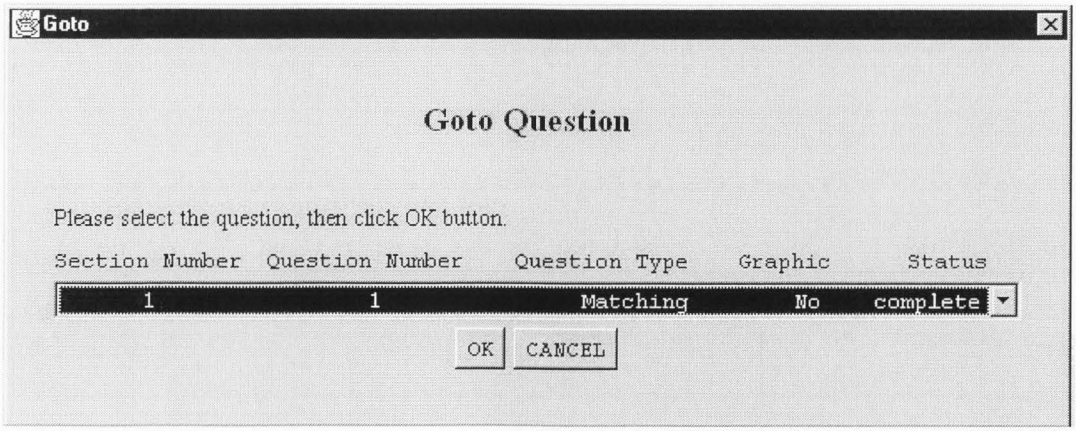


Figure 3- 37. The goto dialog box with question information

Edit Question Score

Each question is assigned a score for grading purposes. The instructor can change the score of the current question by selecting the *Score* menu item from the *Question* pull down menu.

Create New Question

There are seven types of questions: multiple choice, matching, sequencing, fill-in-blank, multiple answer, true or false, and essay. The interface for creating a new question is shown in figure 3-38. When creating a test, the instructor needs to choose the question type from the list and assign a scoring value to the new question. The default scoring value is one. If the new question is a graphic question, the instructor also needs to check the *isGraph* check box. When a new question is created, it is inserted right after the current question in the current section. For example, if the current question is in section 2 and the question number is 5, the new question created will be section 2, question 6. If originally there was question 6 in section 2, that question will be question 7, etc.

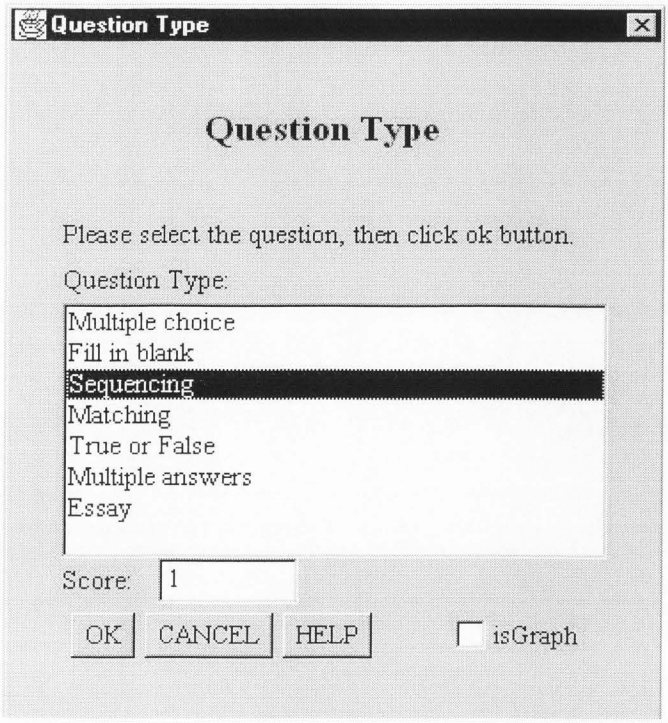


Figure 3- 38. The dialog box for adding a new question

Delete Question

The system will reconfirm the deletion, when *Delete* in the *Question* menu is selected. After the instructor clicked on the *Ok* button, the current question will be removed. All the question numbers of the following questions will be decremented by one. If the current question is the only question in the current section, this section is deleted and all the section numbers of the following questions will be decrement by one.

Once again, any changes made will not be saved to the database until the test has been saved. This could be accomplished by choosing *Save* from the *Test* menu, enabling the *auto-save* feature, or clicking on the *Yes* button when *asked "Do you want to save the changes?*" when exiting the authoring interface.

CHAPTER 4

CONCLUSION

The *NetTest* project was started in March 1997, and has been developed for 9 months until now. The first version was released in the beginning of December 1997, having been installed on the Federal Reserve Bank Salt Lake office as a beta test site. This is the first time we have had the chance to put the *NetTest* into a real environment and we are awaiting their feedback. The Computer Science department in Utah State University plans to use *NetTest* beginning from the Spring quarter of 1998 for several courses. The History department also has shown great interest on *NetTest* and the team is now working on adding more functions to *NetTest* to suit its needs.

THE ADVANTAGES OF USING JAVA

1. Java in its nature is an object-oriented programming language. Every object has its own properties and behaviors and communications between objects is mostly through message passing. Every object is seen as a black box by the others. There is no multiple inheritance in Java. Therefore, with all these features and a good object design a bug is isolated in the module if it exists, so that it will not corrupt the whole system in other modules.
2. Java is running on a Java virtual machine (JVM) and different platforms have their own JVM. Sun Microsystems published the standard for building the JVM to the public. This feature makes Java programs to be cross-platform. As the Sun's slogan reads "*write once, run everywhere*", a Java program can run on various platforms

without recompiling or modifying the code.

3. Java has built-in network capabilities, so that, Java programs are ideal for developing server/client applications over a networked environment.

4. Java has a garbage collection mechanism, does not have pointers, and does not support multiple inheritance. This eliminates a lot of pitfalls and makes using Java much easier than C/C++. Therefore, the learning curve is lower, everyone can learn programming in Java in a short period of time.

THE DISADVANTAGES OF USING JAVA

1. During the development, the biggest problem that we have had is the compatibility of Java between different platforms and different Web browsers. This problem is mostly because Java is in its early stage of introduction to the market. Java itself is still evolving to a more mature language. Every several months Sun releases a newer version of the Java Development Kit (JDK) and the new JDK version tends to have dramatic changes. This causes the companies that build Java-compatible Web browsers difficulties to update their product. To make things worse, different companies decided on proprietary ways to implement JVM. Therefore, the Java applets looks different on different Web browsers, and sometimes even behave differently.

2. A Java program is not a binary executable module, rather it is interpreted by the JVM. Every interpreted program suffers a performance problem, namely, it runs slowly. A Java applet suffers another factor that slows it down: because it resides on the server, the client needs to download the program first, then run it locally. The network bandwidth is critical

for Java applets. When the available network bandwidth is high, there is no significant effect on the Java applets performance, but when the available network bandwidth is low, the Java applets become dramatically slow.

3. There are strict security policies on what Java applet can do and which local resources it can get. I totally agree on that these security policies are necessary. However, they limit the abilities of Java applets and cause the programmers some difficulties, because they need to find ways to work around these restrictions.

4. Finally, another drawback that I found in Java is managing the class file. It is no easy task when the project is large. When compiling a Java program, the compiler generates one class file for each object class. In the *NetTest* project, there are more than 27,000 lines of Java code and 124 different classes. Therefore, there are 124 different class files of which we need to take care. This fact does not pose any serious problem until installation. For a larger project, assuming that the project has thousands of class files, it is almost impossible to handle them properly. Even though Sun has a solution for this, the new version of JDK supports a compressed file format called Java archive (JAR) file. The JAR is similar to zipped files in that a programmer can put several class files into a single file. This feature removes the installation problem describe previously and decreases the download time for Java applets. However, most current available Web browsers do not support the JAR file format. Therefore, it is not a good solution for *NetTest*.

In conclusion, Java is still a good choice on developing server/client programs because of its versatility and ease of learning. I believe that as time goes by, the compatibility problem will be solved and Java will evolve to a mature programming language. After all, *NetTest* does benefit tremendous from the Java features.

CHAPTER 5

FUTURE DEVELOPMENT

This chapter will describe what future work could be done on *NetTest* to make it evolve to a better product. Some of these suggestions are from the client and my supervisors.

TRAINING SESSION

The training should always be accompanied by assessment. In this way, the instructor could know how much the student has learned during the training process. *NetTest* already has the framework and graphic user interface (GUI) for testing. We can use this GUI to design the training session, so that the product can do the training and testing together and still maintain the same portability and flexibility.

ARTIFICIAL INTELLIGENCE ABILITY

After a student has taken tests, the test results are stored in the database. Base on the results, the system is able to determine the level of expertise the student has on the subject. If the student learns well in that subject, the system can provide more advance information during the training session or provide more difficult test question in the testing session on the next time the student uses the system. Or conversely, the system could suggest a re-training or provide easier questions to help the learning process.

TITLE PAGE

Currently, at the beginning of a test, *NetTest* only provides a hard coded brief message to the student. The *NetTest* could provide the opportunity to allow instructors to give specific instructions on taking the test or specific instructions for individual sections by adding a title page for each section and test. The instructor could also decide not to add the title page.

BUNDLED QUESTIONS

The test questions in *NetTest* are selected from the test database in a random order. In the current version, the instructor can make certain questions to be always selected but the order of the questions is not in any particular order. In other words, questions could not be bundled together in the current version. *NetTest* should provide a mechanism to enable the instructor to force several questions to appear in a certain order in the test, and when these are selected, all these questions should be all selected.

RANDOMIZATION

In the current version of *NetTest*, the questions are randomly selected from the test database and the order of multiple answers question is also randomized. The system should allow the instructor to choose whether to randomize the answers, questions, or sections.

REFERENCES

[1] Matrix Information And Directory Services, http://www.mids.org/press957.html.

[2] Cowles/Simba Information, http://www.simbanet.com/.

[3] Internet Distance Education, *Internet Testing System*, http://www.studio-ide.com/index.html.

[4] Internet Educational Testing And Assessment, *EDUTEST*, http://www.edutest.com.

[5] Embley, D., Kurtz, B., and Woodfield, S., *Object-Oriented System Analysis: A Model-Driven Approach*, Prentice Hall , Englewood Cliffs, NJ, 1992.

[6] Jones, G., *Software Engineering*, John Wiley & Sons Inc., NY, 1990.

Appendix A Table Definitions Of NetTest Database

1. Password Table

Key	Column Name	Data Type	Size	Null	Default
Yes	username	varchar	15	No	
No	Password	varchar	15	No	
No	fName	varchar	15	No	(‘’)
No	lName	varchar	15	No	(‘’)
No	mName	varchar	15	No	(‘’)
No	userType	varchar	1	No	
No	major	varchar	10	No	(‘’)
No	ssn	varchar	9	No	(‘’)
No	telNumber	varchar	10	No	(‘’)
No	email	varchar	30	No	(‘’)
No	status	varchar	1	No	(‘a’)
No	testId_unlock	int	4	No	(0)
No	validIP_unlock	varchar	255	No	(‘’)
No	validTime_unlock	varchar	20	No	(‘’)

2. Major Table

Key	Column Name	Data Type	Size	Null	Default
Yes	major	varchar	10	No	

No	title	varchar	70	No	
No	testList	varchar	255	No	('')

3. Log Table

Key	Column Name	Data Type	Size	Null	Default
Yes	username	varchar	15	No	
No	userType	varchar	1	No	
No	testId	varchar	4	No	
No	ip	varchar	15	No	('')
No	logTime	varchar	20	No	('')
No	resetTime	varchar	20	No	('')

4. Std_Info Table

Key	Column Name	Data Type	Size	Null	Default
Yes	username	varchar	15	No	
Yes	testId	Int	4	No	
Yes	attemptNr	Int	4	No	
No	testDate	varchar	10	No	('')
No	expireDate	varchar	10	No	('')
No	status	varchar	1	No	
No	grade	int	4	No	

5. Test Table

Key	Column Name	Data Type	Size	Null	Default
Yes	testId	int	4	No	
No	owner	varchar	255	No	('')
No	subjectTitle	varchar	20	No	
No	testModule	varchar	20	No	('')
No	testType	varchar	20	No	('')
No	unlock	varchar	1	No	
No	passScore	int	4	No	(0)
No	timeLimit	int	4	No	(50)
No	timeValid	int	4	No	(6)
No	attemptLimit	int	4	No	(0)

6. Section Table

Key	Column Name	Data Type	Size	Null	Default
Yes	testId	int	4	No	
Yes	sectionId	int	4	No	
No	sectionTitle	varchar	25	No	
No	questionN	int	4	No	
No	selectedNr	int	4	No	

7. Qindex Table

Key	Column Name	Data Type	Size	Null	Default
Yes	testId	int	4	No	
Yes	indexNr	int	4	No	
No	questionId	int	4	No	
no	sectionNr	int	4	No	
No	questionNr	int	4	No	
No	questionType	int	4	No	
No	selectStatus	char	1	No	
No	score	int	4	No	
No	timesCorrect	int	4	No	
No	timesIncorrect	int	4	No	
No	TimesNotAnswer	int	4	No	
No	timesSelected	int	4	No	

8. Qtype1 Table

Key	Column Name	Data Type	Size	Null	Default
Yes	testId	int	4	No	
Yes	indexNr	int	4	No	
No	questionText	varchar	255	No	('')
No	isGraph	char	1	No	('f')

No	graph	varchar	30	No	('')
No	answer1	varchar	255	No	
No	match1	varchar	255	No	
No	answer2	varchar	255	No	
No	match2	varchar	255	No	
No	answer3	varchar	255	No	
No	match3	varchar	255	No	
No	answer4	varchar	255	No	
No	match4	varchar	255	No	
No	answer5	varchar	255	No	
No	match5	varchar	255	No	
No	answer6	varchar	255	No	
No	match6	varchar	255	No	
No	answer7	varchar	255	No	
No	match7	varchar	255	No	

9. Qtype2 Table

Key	Column Name	Data Type	Size	Null	Default
Yes	testId	int	4	No	
Yes	indexNr	int	4	No	
No	questionText	varchar	255	No	('')

No	isGraph	char	1	No	('f')
No	graph	varchar	30	No	('')
No	answer1	varchar	255	No	('')
No	answer2	varchar	255	No	('')
No	answer3	varchar	255	No	('')
No	answer4	varchar	255	No	('')
No	answer5	varchar	255	No	('')
No	answer6	varchar	255	No	('')
No	answer7	varchar	255	No	('')

10. Qtype3 Table

Key	Column Name	Data Type	Size	Null	Default
Yes	testId	int	4	No	
Yes	indexNr	int	4	No	
No	questionText	varchar	255	No	('')
No	isGraph	char	1	No	('f')
No	graph	varchar	30	No	('')
No	isCorrect	char	1	No	

11. Qtype4 Table

Key	Column Name	Data Type	Size	Null	Default

Yes	testId	int	4	No	
Yes	indexNr	int	4	No	
No	questionText	varchar	255	No	('')
No	isGraph	char	1	No	('f')
No	graph	varchar	30	No	('')
No	answer1	varchar	255	No	('')
No	answer2	varchar	255	No	('')
No	answer3	varchar	255	No	('')
No	answer4	varchar	255	No	('')
No	answer5	varchar	255	No	('')

12. Qtype5 Table

Key	Column Name	Data Type	Size	Null	Default
Yes	testId	int	4	No	
Yes	indexNr	int	4	No	
No	questionText	varchar	255	No	('')
No	isGraph	char	1	No	('f')
No	graph	varchar	30	No	('')
No	answer	vrachar	255	No	('')

13. Qtype6 Table

Key	Column Name	Data Type	Size	Null	Default
Yes	testId	int	4	No	
Yes	indexNr	int	4	No	
No	questionText	varchar	255	No	(‘’)
No	isGraph	char	1	No	(‘f’)
No	graph	varchar	255	No	(‘’)
No	answer1	varchar	255	No	(‘’)
No	answer2	varchar	255	No	(‘’)
No	answer3	varchar	255	No	(‘’)
No	answer4	varchar	255	No	(‘’)
No	answer5	varchar	255	No	(‘’)
No	correct1	varchar	255	No	(‘’)
No	correct2	varchar	255	No	(‘’)
No	correct3	varchar	255	No	(‘’)
No	correct4	varchar	255	No	(‘’)
No	correct5	varchar	255	No	(‘’)

14. Qtype7 Table

Key	Column Name	Data Type	Size	Null	Default
Yes	testId	int	4	No	
Yes	indexNr	int	4	No	
No	question1	varchar	255	No	(‘’)

No	question2	varchar	255	No	(‘’)
No	isGraph	char	1	No	(‘f’)
No	graph	varchar	30	No	(‘’)
No	answe1	varchar	255	No	(‘’)
No	answer2	varchar	255	No	(‘’)

15. Update_Test Table

Key	Column Name	Data Type	Size	Null	Default
Yes	testId	int	4	No	
No	owner	varchar	255	No	(‘’)
No	subjectTitle	varchar	20	No	
No	testModule	varchar	20	No	(‘’)
No	testType	varchar	20	No	(‘’)
No	unlock	varchar	1	No	
No	passScore	int	4	No	(0)
No	timeLimit	int	4	No	(50)
No	timeValid	int	4	No	(6)
No	updateStatus	char	1	No	(‘u’)
No	graphSeq	int	4	No	(0)
No	attemptLimit	int	4	No	(0)

16. Update_Section Table

Same as Section table.

17. Update_Qindex Table

Key	Column Name	Data Type	Size	Null	Default
Yes	testId	int	4	No	
Yes	indexNr	int	4	No	
No	questionId	int	4	No	
no	sectionNr	int	4	No	
No	questionNr	int	4	No	
No	questionType	int	4	No	
No	selectStatus	char	1	No	
No	score	int	4	No	

18. Update_Qtype1 Table

Same as Qtype1 table.

19. Update_Qtype2 Table

Same as Qtype2 table.

20. Update_Qtype3 Table

Same as Qtype3 table.

21. Update_Qtype4 Table

Same as Qtype4 table.

22. Update_Qtype5 Table

Same as Qtype5 table.

23. Update_Qtype6 Table

Same as Qtype6 table.

24. Update_Qtype7 Table

Same as Qtype7 table.