Web-based interactive self-evaluation system for computer science in generic tutorial system for the sciences project

Supachai Praritsantik

Follow this and additional works at: https://scholarworks.lib.csusb.edu/etd-project

Recommended Citation
Praritsantik, Supachai, "Web-based interactive self-evaluation system for computer science in generic tutorial system for the sciences project" (2002). Theses Digitization Project. 2273. https://scholarworks.lib.csusb.edu/etd-project/2273
WEB-BASED INTERACTIVE SELF-EVALUATION SYSTEM FOR COMPUTER
SCIENCE IN GENERIC TUTORIAL SYSTEM FOR THE
SCIENCES PROJECT

A Project
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Supachai Praritsantik
March 2002
WEB-BASED INTERACTIVE SELF-EVALUATION SYSTEM FOR COMPUTER

SCIENCE IN GENERIC TUTORIAL SYSTEM FOR THE

SCIENCES PROJECT

A Project
Presented to the
Faculty of
California State University,
San Bernardino

by
Supachai Praritsantik
March 2002
Approved by:

Arturo Concepcion, Chair, Computer Science 16 Jan 2002

Kay Zemmel

George M. Georgiou
ABSTRACT

The goal of this master project, Web-based Interactive Self-evaluated System (WISE), is to promote and facilitate the use of new Java-based and Web-based technologies in the development of teaching materials for Computer Science in particular, analysis of sorting algorithms. This project is built as part of the Generic Tutorial System for the Sciences (GTSS) project. WISE promotes Web-based interactive exercises by providing educators with the tools to create interactive exercise materials.

As part of GTSS project, WISE provides both tutorials and exercises. Students can study the interactive tutorials and test the self-evaluation exercises. Self-evaluation result indicates students’ understanding.

WISE provides tools such as EZ question creator for creating question database and automatic exercises, and knowledge base creator for creating new intelligent java programs filled with how to create automatic answer with prevented cheating policy.

WISE uses object oriented and unified modeling language to design the system. To implement WISE system, Java Server Pages (JSP) and MySQL are used. JSP technology makes WISE system possible to run over the Internet.
ACKNOWLEDGMENTS

I would like to thank Dr. Concepcion, my advisor, for guiding me in the steps I needed to take to get this master project done. Also thanks to Dr. Zemoudeh and Dr. Georgiou for the time they invested guiding me to create the question database.

And thanks to the Associated Studies, Incorporated (ASI), for their grant to provide partial funding support for this project.

Supachai Praritsantik
TABLE OF CONTENTS

ABSTRACT ................................................................. iii
ACKNOWLEDGMENTS .................................................... iv
LIST OF TABLES .......................................................... viii
LIST OF FIGURES ......................................................... x

CHAPTER ONE: SOFTWARE REQUIREMENT SPECIFICATION

1.1 Introduction ....................................................... 1
  1.1.1 Purpose of Project ........................................... 1
  1.1.2 Scope of Project .............................................. 2
1.2 Overall Project Description ......................... 2
  1.2.1 Project Product .............................................. 2
  1.2.2 Functionality of Project ................................... 5
  1.2.3 System Analysis and Requirements ................... 8
  1.2.4 Hardware and Software Requirement ................ 10
  1.2.5 User Characteristics .................................. 13
1.3 Software Specific Requirements ................. 13
  1.3.1 External Interface ..................................... 13

CHAPTER TWO: DESIGN .................................................. 36

2.1 Architecture Design ......................................... 37
  2.2.1 Sorting Algorithm Tutorials Design .................. 37
  2.1.2 Interactive Self-Evaluation System Design ........ 47
  2.1.3 Database Design ..................................... 53
2.2 Detailed Design ................................................ 59
  2.2.1 Connect Database Class ................................. 60
2.2.2 Query Bean Class ......................... 61
2.2.3 Data Manipulation Class .................. 63
2.2.4 Connect File Class ........................ 65
2.2.5 File Bean Class ............................ 66
2.2.6 Read File Bean Class ...................... 67
2.2.7 Score Bean Class ............................ 68
2.2.8 Create Choice Class ...................... 69
2.2.9 Shuffle Class ............................... 70
2.2.10 Random Number Class .................... 71
2.2.11 Add Question General Class ............. 72
2.2.12 Add Question Java Class ................. 74
2.2.13 Question Retrieval Manager Class ...... 76
2.2.14 Question General Class ................... 78
2.2.15 Question Java Class ...................... 79
2.2.16 Table of Content Check Class .......... 83
2.2.17 Knowledge Base Check Class .......... 85
2.2.18 Knowledge Base Deletion Class .......... 88
2.2.19 User Information Class ................... 89

CHAPTER THREE: TESTING

3.1 The Testing Process ......................... 95
3.2 Unit Testing ................................. 95
3.3 Subsystem Testing ............................ 100
3.4 System Testing ............................... 101
3.5 Sample Session .............................. 102
  3.5.1 Demonstration of Tutorial ............... 102
LIST OF TABLES

Table 1.1. Hardware Requirements for Development Phase ........................ 11
Table 1.2. Software Requirements for Development Phase ........................ 11
Table 1.3. Hardware Requirements for Implement Phase .......................... 12
Table 1.4. Software Requirements for Implement Phase .......................... 12
Table 2.1. Core Objects ............................................. 38
Table 2.2. Computer Science Engine Objects ................................. 39
Table 2.3. New Application Objects .................................. 41
Table 2.4. Core Classes .............................................. 47
Table 2.5. Question Manager Objects ................................ 48
Table 2.6. Test Taking Objects ..................................... 48
Table 2.7. Table of Content Manager ................................. 49
Table 2.8. Knowledge Base Manager ................................ 49
Table 2.9. User Manager ............................................. 49
Table 2.10. Miscellaneous Objects ................................... 50
Table 2.11. User Information ....................................... 53
Table 2.12. Tutorial .................................................. 53
Table 2.13. Table of Content ....................................... 54
Table 2.14. Exam Control ............................................ 54
Table 2.15. Knowledge Base ....................................... 55
Table 2.16. Question ................................................. 56
Table 2.17. Question General ....................................... 57
Table 2.18. Question Java .......................................... 57
Table 2.19. Parameter ................................................. 58
Table 2.20. Score Table ................................................. 59
Table 3.1. Unit Test Results ............................................ 96
Table 3.2. Subsystem Test Results ................................. 101
Table 3.3. System Test Results ....................................... 102
Table 4.1. Files in Main Directory ................................. 107
LIST OF FIGURES

Figure 1.1. Project Integration ........................................ 2
Figure 1.2. Project Overview ........................................... 3
Figure 1.3. Overview of Project ....................................... 4
Figure 1.4. Use Case Diagram .......................................... 6
Figure 1.5. Authorization Page ........................................ 14
Figure 1.6. Student Signup Page ...................................... 15
Figure 1.7. Forget Password Page ...................................... 16
Figure 1.8. Student Main Menu ......................................... 17
Figure 1.9. Test Selection Page ........................................ 18
Figure 1.10. Test Page ................................................... 19
Figure 1.11. User Modification Page ................................... 20
Figure 1.12. Instructor Main Menu .................................... 21
Figure 1.13. Add Table of Content .................................... 22
Figure 1.14. Table of Content Management Page ................. 23
Figure 1.15. Knowledge Base Creator Page ....................... 24
Figure 1.16. Knowledge Base Management ......................... 25
Figure 1.17. Adding General Question Form ....................... 26
Figure 1.18. Easy Question Creator Step One .................... 27
Figure 1.19. Easy Question Creator Step Two .................... 28
Figure 1.20. Easy Question Creator Step Three .................. 29
Figure 1.21. Easy Question Creator Step Four ................... 30
Figure 1.22. Question Control Page ................................ 31
Figure 1.23. Administrator Sign Up Page ........................... 32
Figure 1.24. Tutorial Page ............................................. 33
Figure 1.25. Insertion Sort Animation ..................... 34
Figure 1.26. Heap Sort Animation .......................... 35
Figure 2.1. Tutorial Applet Structure ....................... 37
Figure 2.2. Insertion Sort Class Diagram .................... 43
Figure 2.3. Selection Sort Class Diagram ..................... 44
Figure 2.4. Heap Sort Class Diagram ........................ 45
Figure 2.5. Quick Sort Class Diagram ........................ 46
Figure 2.6. Project Class Diagram ............................ 52
Figure 2.7. Connect Database Class ........................... 60
Figure 2.8. Query Bean Class .................................. 61
Figure 2.9. Data Manipulation Class .......................... 63
Figure 2.10. Connect File Class ............................... 65
Figure 2.11. File Bean Class ................................... 66
Figure 2.12. Read File Bean Class .............................. 67
Figure 2.13. Score Bean Class .................................. 68
Figure 2.14. Create Choice Class ............................. 69
Figure 2.15. Shuffle Class ..................................... 70
Figure 2.16. Random Number Class ........................... 71
Figure 2.17. Add Question General Class ...................... 72
Figure 2.18. Add Question Java Class ........................ 74
Figure 2.19. Question Retrieval Class ......................... 76
Figure 2.20. Question General Class ........................... 78
Figure 2.21. Question Java Class .............................. 79
Figure 2.22. Table of Content Check Class .................... 83
Figure 2.23. Knowledge Base Check Class ..................... 85
Figure 2.24. Knowledge Base Deletion Class ............. 88
Figure 2.25. User Information Class ................. 89
Figure 3.1. Sorting Algorithms Tutorial ............... 102
Figure 3.2. General Question Creator ................. 103
Figure 3.3. Testing Knowledge Base Creator .......... 104
Figure 3.4. Easy Question Creator ..................... 105
CHAPTER ONE
SOFTWARE REQUIREMENT SPECIFICATION

1.1 Introduction

1.1.1 Purpose of Project

There are many Web sites that provide interactive tutorials for students such as CBT, GTSS, and etc. There is no evaluation of students' understanding and learning in these web sites. In order to solve this problem, the idea of creating evaluation tool was proposed in Fall 2000 as a master project called "Web-based Interactive Self-Evaluation System".

The goal of this master project is to promote and facilitate the use of new Web-based and Java-based technologies in the development of self-evaluation system for computer science in particular, analysis of sorting algorithms. This project is built as part of the GTSS system (see figure 1.1). WISE will promote Web-based interactive exercises by providing educators with the tools to create interactive exercise materials. Students are able to evaluate their understanding though exercises.
1.1.2 Scope of Project

WISE project will provide tools to create exercises and to evaluate students' performance. Evaluation topics are limited to sorting algorithms that include Selection Sort, Insertion Sort, Bubble Sort, Quick Sort and Heap Sort. These are important topics in computer science. Students who understand sorting algorithms very well can develop efficiently software.

1.2 Overall Project Description

1.2.1 Project Product

WISE system is designed to run on the Internet. Therefore students can access WISE system from anywhere. As an extension from the GTSS project, WISE system makes
the final product "the complete tutorial system". The complete WISE project consists of two main parts, GTSS tutorial and WISE application server (see figure 1.2).

![Project Product with Deployment Diagram](image)

Figure 1.2. Project Overview

1. **Sorting algorithm Tutorials.** These tutorials include the concept of each sorting algorithms, and analysis of sorting algorithms in term of Big Oh notation.

2. **Interactive self-evaluation exercise system.** This system includes interactive exercise creator tool, question management tool, and score analysis. These easy-to-use tools will help instructors create interactive exercises in the area of sorting algorithms. Creating some of the exercises may require the programming to set rules.
for WISE in order to generate multiple choices. This information is kept in knowledge base, which later on can be used to create more interactive questions.

Figure 1.3 shows how WISE works. The instructor creates both tutorials and question exercises. Then the

Figure 1.3. Overview of Project

students are able to study and practice the self-evaluation exercises, which are created from the WISE
system. The system will evaluate the student scores with the passing scores set by the instructors. If students pass, they can start the next chapter otherwise they repeat the current chapter is required.

1.2.2 Functionality of Project

In WISE system, all functions are categorized by level of users, which include student level, instructor level and administrator level. Figure 1.4 shows a Use Case diagram, which describes the functionalities provided by WISE system.

1.2.2.1 Instructor Level. The instructor is able to create, edit and delete sorting algorithm questions by using tools such as EZ question creator to create questions and Question Manager to edit and delete questions. Question Manager engine handles all activities described above. The instructor can also access students' record to view their score and evaluate students' understanding.
Figure 1.4. Use Case Diagram
1.2.2.2 **Student Level.** After the student finished their tutorial applets, The Question Manager engine will generate a set of questions. Most questions automatically generate answers. Then they are shuffled them by the Question Manager engine. The Answer Manager engine is responsible for counting score and recording it in the student database. If the student passes the exam, the student can start the next tutorial, otherwise the Answer Manager engine locks the current chapter until the student passes the exam.

1.2.2.3 **Administrator Level.** System administrator or SA has the major task to maintain all the tables in the WISE database and overall system. SA is able to create instructor and student account. Students and instructors cannot delete their account unless the SA does. The User Manager engine will handle these tasks on the background. User manager engine also checks user logins to ensure that only registered student access the system.

SA has also to maintain the knowledge base. With the knowledge base engine, SA can manage the knowledge base that is used for creating new questions by the instructors.
1.2.3 System Analysis and Requirements

The WISE system consists of several components in order to work properly: a) Java Server Pages (JSP), b) Java Database Connectivity driver (JDBC), and c) MySQL database engine. In this section, each component will be described briefly. In WISE system, all information will be stored in MySQL databases. The front-end applications will be written by JSP. All applications will use JDBC to connect to databases.

(a) Java Server Pages consists of powerful JAVA programming language integrated with the server-side script language based on extra markup language or XML. These make JSP easy for dynamic Web programming. Furthermore, JSP uses graphics from HTML. As a result, JSP applications run faster. Compared to the other server-side language such as, Active Server Pages (ASP), PERL or PHP, JSP has the major advantages of real object-oriented, run faster, and highly reusable. JSP is a language of choice for creating dynamic Web-based interaction.

Since WISE contains many interactive forms to get data in or to display result out, it is a good idea to use JSP in WISE system. Integration
of GTSS and WISE system will not slow the system down because WISE system will optimize GTSS performance.

There are many advantages to use JSP in WISE system:
- WISE system is object-oriented, which is the same as GTSS. WISE can be very modular and readily adapted to new uses.
- As a teaching interactive material tool, a well-written JSP program allows the instructor to focus attention precisely on the point at hand.
- With build-in templates, it will be easy for instructors of the computer sciences to develop additional interactive exercises with minimal requirement for programming.

(b) Java Database Connectivity defines a structured interface to Standard Query Language (SQL) database. The JDBC API provides Java developers with consistent approach to accessing SQL databases that is comparable to existing database development techniques. The JDBC API includes classes for common SQL database constructs such as database connections, SQL statements, and
result sets. JDBC database drivers can either be written entirely in Java or they can be implemented using native methods to bridge Java application to existing database access libraries.

(c) Databases are becoming a big part of applications, and one of the most popular products is MySQL, from MySQL.com. MySQL is a free product under General Public License. MySQL has many advantages over the other database engine products such as reliability, speed, and capacity.

MySQL can run on any platforms such as windows 98 or NT, UNIX, LINUX, and etc. Using MySQL is also easy to use with simple SQL commands.

1.2.4 Hardware and Software Requirement

In order to successful develop WISE system; the hardware and software are a major concern. Minimum requirements are specified here both a) development and b) implement phases.

(a) Development Phase is the phase that involves design, coding and debugging of this project. The following hardware and software specification are
used to develop and test WISE to work properly when it is launched.

Table 1.1. Hardware Requirements for Development Phase

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>AMD Athlon 700 Mhz</td>
</tr>
<tr>
<td>Memory</td>
<td>Micron 128 MB</td>
</tr>
<tr>
<td>Display card</td>
<td>ATI rage fury 32 MB</td>
</tr>
<tr>
<td>BIOS</td>
<td>American Megatrend 2.0</td>
</tr>
</tbody>
</table>

Table 1.2. Software Requirements for Development Phase

<table>
<thead>
<tr>
<th>Software</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation system</td>
<td>Windows ME, Redhat Linux 7.1 server</td>
</tr>
<tr>
<td>Browser</td>
<td>Internet Explorer 5.5</td>
</tr>
<tr>
<td>JSP server</td>
<td>Tomcat 3.2.1</td>
</tr>
<tr>
<td>Java Compiler</td>
<td>JSDK 1.3.1</td>
</tr>
<tr>
<td>Database</td>
<td>MySQL</td>
</tr>
<tr>
<td>Text Editor</td>
<td>Notepad</td>
</tr>
<tr>
<td>Other script language</td>
<td>JavaScript</td>
</tr>
</tbody>
</table>

WISE is mainly developed under windows ME operating system. The main reasons developing it in Windows is that it is easy to design, code and debug since windows ME provides more friendly graphic user interface. WISE is also tested on
Linux operating system to ensure that WISE system will run on any platform.

(b) The final product will be tested and run on Linux Operating System. The GTSS server will include all GTSS works, WISE system, and all future extension projects.

Table 1.3. Hardware Requirements for Implement Phase

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Pentium 1 Gz.</td>
</tr>
<tr>
<td>Memory</td>
<td>Micron 128 MB</td>
</tr>
<tr>
<td>Display card</td>
<td>Gforce 32 MB</td>
</tr>
<tr>
<td>BIOS</td>
<td>Dell BIOS 2.0</td>
</tr>
</tbody>
</table>

Table 1.4. Software Requirements for Implement Phase

<table>
<thead>
<tr>
<th>Software</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation system</td>
<td>Redhat Linux 7.1 server</td>
</tr>
<tr>
<td>Browser</td>
<td>Netscape 4.7 for Linux</td>
</tr>
<tr>
<td>JSP server</td>
<td>Tomcat 3.2.1 for Linux</td>
</tr>
<tr>
<td>Java Compiler</td>
<td>JSDK 1.3.1 for Linux</td>
</tr>
<tr>
<td>Database</td>
<td>MySQL for Linux</td>
</tr>
<tr>
<td>Text Editor</td>
<td>VI</td>
</tr>
</tbody>
</table>
1.2.5 User Characteristics

Since this project is about analysis of sorting algorithms, the intended users include both computer science instructors and students. Instructors need to have some knowledge of Java programming since some intelligent questions require logic solutions on how to solve these questions in the form of fragments of Java program. Students need to know how to use the Internet.

1.3 Software Specific Requirements

1.3.1 External Interface

In this section, we will continue our discussion of WISE's basic interfaces and functionality expectations.
1.3.1.1 Authorization. This is the main login page (see figure 1.5). There are two input text fields, username and password. All users, students, instructors and system administrator, will use this login page. Successful logins will bring the user to their own main page.

Figure 1.5. Authorization Page
1.3.1.2 New Student Sign Up. This sign up page is designed for new students who want to sign up to use WISE system (see figure 1.6). Student who provides all required information will be registered and will be able to access the system.

![New Student Sign Up Page](image)

Figure 1.6. Student Signup Page
1.3.1.3 Forget Password Page. The student or instructor who forget their password can use this page by filling out his identification (see figure 1.7). WISE system will send the password them through e-mail.

Figure 1.7. Forget Password Page
1.3.1.4 Student Main Menu. This is the student main menu (see Figure 1.8). Student can use this menu to use WISE system. Start tutorial is to start all GTSS tutorial. Self-Test exercise will link to the exercise page to test students' understanding. Students can check their scores at the score and statistic page. Students can also change their password, email at preference link. The logoff link will end the student's session.

![Student Main Menu](image-url)

Figure 1.8. Student Main Menu
1.3.1.5 Test Selection Page. The student can choose the chapter that he/she wants to test (see Figure 1.9). By clicking "Create Exercise", the exercise for that chapter will create and show on screen. If the student clicks on the "Back to Main", the student’s main menu page will be brought back.

Figure 1.9. Test Selection Page
1.3.1.6 Testing Page. Shown in Figure 1.10 is the test page created by WISE system. The student can answer it at any time. After the student answers all the questions, they will click "submit Your Answer". WISE will check the answers and print the score report.

Figure 1.10. Test Page
1.3.1.7 Change User Information. Users who want to change their user information or password can do it on this page (see Figure 1.11). There are two separate buttons here: “Change User Info” and “Change Password”. These buttons are self-explanatory. Clicking “Change User Info” action button will update the user name or e-mail. Clicking “Change Password” will update password for the current user.

Figure 1.11. User Modification Page
1.3.1.8 Instructor Main Menu. The instructor main page shows links to perform many tasks (see Figure 1.12). "Table of Content" links to Table Of Content Manager to create table of content of WISE system. "Knowledge Base" links to knowledge base manager to create the knowledge base. EZ question creator links to Java question creator. Display question links to question manager. "Exam Control" links to exam control page to set the number of question per test and percent pass. Statistics links to score manager to monitor student score.

![Instructor Main Menu](image)

Figure 1.12. Instructor Main Menu
1.3.1.9 Add Table of Content Page. This screen helps the instructor handle the table of content in WISE system (see Figure 1.13). After filling out all information and clicks “submit”, the system will add new content to the database. “Reset” button will be used for clearing old information. TOC manager page will bring instructor back to the table of content main menu.

Figure 1.13. Add Table of Content
1.3.1.10 Display Table of Content. Display table of content menu will help the instructor handle table of content in the WISE system (see Figure 1.14). "Edit" button links to edit page to modify content. "Delete" button links to delete page to delete unwanted content from WISE system.

![Table of Content Management Page](image)

*Figure 1.14. Table of Content Management Page*
1.3.1.11 Knowledge Base Creator. This screen will add knowledge base to the database (see Figure 1.15). Knowledge base helps the instructor create questions in the future. After filling out all the information and clicking "submit", the system will write all information to the database.

Figure 1.15. Knowledge Base Creator Page
1.3.1.12 Knowledge Base Display Page. This is a similar screen to the table of content. Knowledge base shows question in the question field (see Figure 1.16). “Edit” button links to modify the knowledge base. “Delete” button links to delete page, which will delete unwanted knowledge.

Figure 1.16. Knowledge Base Management
1.3.1.13 General Question Creator. General question creator shows the way to create non-calculation type of question (see Figure 1.17). After filling out all the information and clicking "submit" button, Wise system will add this question to the database. Reset button will clear all information in all fields.

Figure 1.17. Adding General Question Form
1.3.1.14 Easy Question Creator. Easy (EZ) question creator is a tool to create calculation question by using the information from the knowledge base (see Figure 1.18). EZ question creator consists of 4 easy steps. The first step is show here. After selecting all information and clicking “next” button, Wise will show the second step.

Figure 1.18. Easy Question Creator Step One
1.3.1.15 Easy Question Creator Step 2. The knowledge base that was setup from the last screen will show here (see Figure 1.19). The instructor just chooses the information from the list and click “next” to go to next step. The instructor can click “back” to go back to the previous screen.

![Easy Question Creator for GTSS](image)

**Figure 1.19. Easy Question Creator Step Two**
1.3.1.16 Easy Question Creator Step 3. This screen shows step three of EZ Question Creator (see Figure 1.20). This screen shows the detail of the question the user intends to create based on selected knowledge base. After filling out all information and clicking on “Next” button, WISE will proceed to step four. The instructor clicks “Back” button to go to the previous step page.

Figure 1.20. Easy Question Creator Step Three
1.3.1.17 Easy Question Creator Step 4. This is the last screen for the calculation type of questions (see Figure 1.21). The instructor will choose “Auto Create Answer” or define his own answers. The “Next” button will save all information to database, which later on be ready for system to create tests for the student.

Figure 1.21. Easy Question Creator Step Four
1.3.1.18 Question Control Page. After creating the question, the instructor can control the number of questions per exam per chapter (see Figure 1.22). The instructors can also control how many percent to pass each exam. "Save Change" button will save all information to database. "Back to Main" button will go back to the main menu.

Figure 1.22. Question Control Page
1.3.1.19 Sign Up Page. This sign up page is exclusively for administrator (see Figure 1.23). User classification includes student and instructor. Instructor account can be created here. "Signup" button saves the information to database. "Reset Form" button will clear all information from this form.

![Administrator Sign Up Page](image)

Figure 1.23. Administrator Sign Up Page
1.3.1.20 Tutorial Page. This is an example of tutorial. All tutorials have standard layout. “Next” button links to next tutorial pages.

Figure 1.24. Tutorial Page
1.3.1.21 Insertion Sort Animation. This is an insertion sort animation written in Java. “Forward” button shows next step of algorithm animation. “Back” button shows backward step of algorithm. “Reset” button start over applet. “Keyboard” button accepts user input from keyboard. Time complexity of insertion sort is $O(n^2)$

Figure 1.25. Insertion Sort Animation
1.3.1.21 Heap Sort Animation. This is an example page of Heap Sort. The time complexity of heap sort is $O(n\log n)$.

```java
public static void heapSort(int[] v, int n) {
    for (i = n/2; i >= 1; i--)
        adjust(v, i, n);
    for (i = n-1; i > 0; i--)
        swap(v, 0, i);
    adjust(v, 0, 1);
}

public static void adjust(v[], int position, int heapSize) {
    while (position < heapSize) {
        int childPos = position * 2 + 1;
        if (childPos < heapSize) {
            if ((childPos + 1 < heapSize) && v[childPos + 1] > v[childPos])
                childPos = childPos + 1;
        } else
            swap(v, position, childPos);
        position = childPos;
    }
}
```

Figure 1.26. Heap Sort Animation
CHAPTER TWO
DESIGN

After defining all software specification, the next step in software development life cycle is design. Design is a creative process. Software design quality is depended on understandability, verification and adaptability. To achieve this goal, Unified Modeling Language (UML) is used for software analysis and design. It simplifies the complex process of software design, making a “blueprint” for construction. Use Case is one example of UML that define functionality of software in the last chapter. In this chapter other UML diagrams, such as class diagram, will be used in the following section. From the last chapter (Figure 1.2), project product goals are to create sorting algorithm tutorials and complete interactive tutorial system in the GTSS system. Therefore these two product goals will be discussed.

The GTSS Project is an on-going project in the department of computer science, mathematics, physics, and chemistry at CSUSB. We used the GTSS tools and built new ones to implement WISE.
2.1 Architecture Design

2.2.1 Sorting Algorithm Tutorials Design

The GTSS structure consists of 3 layers: core objects, engine objects and application objects [1]. For Sorting Algorithm GTSS applets, WISE still use the same structure. Furthermore, new applets add more classes to the engine objects and application objects to make new sorting algorithm applets run correctly. Figure 2.1 shows structure of GTSS.

![Diagram of GTSS structure]

Figure 2.1. Tutorial Applet Structure

From Figure 2.1, Core Objects are used for creating primitive graphics such as frames, text, canvases, buttons, menus, etc. The Engine Objects are build on top
of the core objects and though inheritance and polymorphism, this engine supports the generation of windows and graphics for the specified subject. The Application Objects are a set of objects that is used to generate windows and graphics for a specific topic in a subject.

2.1.1.1 Core Objects. There are seven classes in the Core Objects that are used to build new sorting algorithm applet. They are SGApplet, SGFrame, SGControlPanel, SGMenu, SplashPanel, HsplitPanel, and Vspiltpanel. (See Table 2.1)

Table 2.1. Core Objects

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGApplet</td>
<td>This is the superclass of all applet</td>
</tr>
<tr>
<td>SGFrame</td>
<td>Defines a Frame.</td>
</tr>
<tr>
<td>SGControlPanel</td>
<td>Specifies button on panel.</td>
</tr>
<tr>
<td>SGMenu</td>
<td>Specifies menus on menus bar.</td>
</tr>
<tr>
<td>SplashPanel</td>
<td>A panel used for demonstration.</td>
</tr>
<tr>
<td>HsplitPanel</td>
<td>Defines a two horizontally split panels.</td>
</tr>
<tr>
<td>VspiltPanel</td>
<td>Defines a two vertically split panels.</td>
</tr>
</tbody>
</table>

2.1.1.2 Computer Science Engine Objects. The Computer Science Engines that are used in creating new sorting algorithm tutorials contain six classes: ScatterPanel,
SortPanel, ScatterInterface, Pseudocode, SortInterface, and Sortnode. (See Table 2.2)

Table 2.2. Computer Science Engine Objects

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScatterPanel</td>
<td>Panel that displays the scatter graph.</td>
</tr>
<tr>
<td>ScatterInterface</td>
<td>Class interface to ScatterPanel.</td>
</tr>
<tr>
<td>SortPanel</td>
<td>Defines the base panel for sort walkthroughs.</td>
</tr>
<tr>
<td>Pseudocode</td>
<td>Defines the base panel to display algorithm.</td>
</tr>
<tr>
<td>SortInterface</td>
<td>Class interface for SortPanel &amp; Pseudocode.</td>
</tr>
<tr>
<td>Sortnode</td>
<td>Defines graphical boxes for sorting animation.</td>
</tr>
</tbody>
</table>

The Computer Science Engine Objects are built using Core Objects. Computer Science Objects define basic graphics about animations such as sorting pseudocode and rectangular boxes for sorting animation. It rules also contains how to animate them such as swapping the boxes or highlighting on lines in the pseudocode.

2.1.1.3 Application Objects. The current GTSS structure supports four applications: Computer Science, Physics, Mathematics, and Statistics. We will continue our discussion on computer science applets.
The five new applets added to GTSS for computer science are the following:

(a) Selection Sort is a Selection Sort walkthrough of the steps by steps execution of the algorithm.

(b) Insertion Sort is an Insertion Sort walkthrough of the steps by steps execution of the algorithm.

(c) Quick Sort is a Quick Sort walkthrough of the steps by steps execution of the algorithm.

(d) Heap Sort is a Heap Sort walkthrough of the steps by steps execution of the algorithm.

In Application Objects, we define the rule how the algorithms associate with the pseudocode and rectangular boxes by adding actionlistener. Every time the user clicks on “forward” button. We define two more classes for every sorting applet to handle this situation. In insertion sort, for example, they are Iswalk.java and ISControl.java. These two classes will tell the applet what animation should it do. This method is applied to every new sorting algorithm in GTSS. Table 2.3 shows new Application Objects built for new sorting tutorial algorithms.
Table 2.3. New Application Objects

<table>
<thead>
<tr>
<th>Class name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iswalk and ISControl</td>
<td>Integrate all components objects and Control Insertion sort animation applet</td>
</tr>
<tr>
<td>SSWalk and SSControl</td>
<td>Integrate all components objects and Control Selection Sort animation applet</td>
</tr>
<tr>
<td>Hswalk and HSControl</td>
<td>Integrate all components objects and Control Heap sort animation applet</td>
</tr>
<tr>
<td>Qswalk and QSControl</td>
<td>Integrate all components objects and Control Quick sort animation applet</td>
</tr>
</tbody>
</table>

Using GTSS to create new applet is quite easy since there are preprogrammed classes available for instructors to create a new applet. But one major drawback is that instructors must know how to program in Java. Using Java bean visual tools that help to eliminate the need of programming in Java because Java bean allows the user to create applet in a visual programming environment can solve this problem.

Figure 2.2-2.5 shows the class diagrams of Insertion sort, Selection Sort, Quick Sort and Heap Sort. All five sorting algorithm applets have the same class diagram structures. The class diagram shows all the Objects used now integrated into one big picture showing the
relationships of all objects. This class diagram also shows the detail of functions and variables for each class that binds together to run this applet. This is also the standard pattern of all future animation applets unless this rule will be customized by instructors to fit future applet requirements.
How to renew your books from Home

1. Access to Library Homepage.  
   http://www.lib.csusb.edu
2. Select Renew Your Books  
   bottom of page.
3. Click My Account Tab
4. Enter your barcode (on the  
   back of your Coyote OneCard).
5. Click (Login)
6. Click Checkout Out Tab
7. Select item(s) for renewal by  
   clicking in box next to each  
   each title. Then click on bottom  
   highlighted Renew box.
8. After renewing your item(s)  
   make a print out for your receipt.  
   The Library will make no fee  
   adjustments unless you can show  
   this proof of renewal.

If you have any questions  
concerning renewals please  
call Circulation Desk at 880-5090.

Problems/questions with signing  
onto the Library's web site or  
accessing the periodical indexes  
call Help Desk 880-5116 or 5107.

Figure 2.2. Insertion Sort Class Diagram
Figure 2.3. Selection Sort Class Diagram
Figure 2.4. Heap Sort Class Diagram
Figure 2.5. Quick Sort Class Diagram
2.1.2 Interactive Self-Evaluation System Design

As an extension project for GTSS, WISE is added. WISE system design is based on object-oriented design. WISE architecture consists of two object layers: core and application.

The Core Object in WISE system consists of both main classes that connect to database, and file and utility classes that deploys superclasses to perform tasks that help WISE run application object faster and with no error. Table 2.4 shows all classes in Core Objects.

Table 2.4. Core Classes

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Class Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connectsql</td>
<td>A super class for all objects. To define database</td>
</tr>
<tr>
<td>Connectfile</td>
<td>A Super class for file objects.</td>
</tr>
<tr>
<td>QueryBean</td>
<td>Control query statement</td>
</tr>
<tr>
<td>InsDelUpdBean</td>
<td>Control insert, delete or update in database.</td>
</tr>
<tr>
<td>Filebean</td>
<td>Worked with connectfile, contain detail information to save to or read from identified filename.</td>
</tr>
<tr>
<td>Shuffle</td>
<td>Shuffle all choices by random.</td>
</tr>
<tr>
<td>CreateChoice</td>
<td>Base from correct answer, system will create almost the other four right answers before shuffle</td>
</tr>
<tr>
<td>RandomNumber</td>
<td>Random number by specified ranges.</td>
</tr>
<tr>
<td>Scorebean</td>
<td>Control and manage student's score.</td>
</tr>
<tr>
<td>ReadFileBean</td>
<td>Reading information from file and send to display at screen</td>
</tr>
</tbody>
</table>
Application objects are classes that present all functionality provided by WISE system. Functionalities of WISE system was explained in chapter one using case diagram. The main functionalities that show in the application objects can divided into five main groups: question manager, user manager, table of content manager, test taking and miscellaneous. The following tables show all Application Objects categorized by function.

Table 2.5. Question Manager Objects

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Class Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddQuestionG</td>
<td>Add new no-calculation question to the database</td>
</tr>
<tr>
<td>AddQuestionJ</td>
<td>Add new calculate algorithm question to the database</td>
</tr>
</tbody>
</table>

Table 2.6. Test Taking Objects

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Class Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qmanager</td>
<td>Manage and create exercises to show on screen</td>
</tr>
<tr>
<td>QuestionG</td>
<td>Retrieve non-calculation question to Qmanager</td>
</tr>
<tr>
<td>QuestionJ</td>
<td>Retrieve calculation question to Qmanager</td>
</tr>
</tbody>
</table>
Table 2.7. Table of Content Manager

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Class Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOC_check</td>
<td>Check, add, modify and delete table of content</td>
</tr>
</tbody>
</table>

Table 2.8. Knowledge Base Manager

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Class Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>KbaseDelete</td>
<td>Delete Knowledge base</td>
</tr>
<tr>
<td>KnowledgebaseCheck</td>
<td>Check knowledge base parameter before saving to database</td>
</tr>
</tbody>
</table>

Table 2.9. User Manager

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Class Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserInfoBean</td>
<td>Control and manage user login and signup</td>
</tr>
</tbody>
</table>
Table 2.10. Miscellaneous Objects

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Class Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email</td>
<td>Provide forget password mailing service</td>
</tr>
<tr>
<td>Sortgen</td>
<td>Acquire calculation question with generated answers to QuestionJ</td>
</tr>
</tbody>
</table>

Figure 2.7 shows the WISE class diagram. The connectsql class and connectfile are superclasses of all classes in WISE system. Connectsql has the database connection information. Connectfile also has filename connection information, which is useful when the system loads the student’s exercise file.

After the student logs on to WISE system, the student can browse student menu, which includes GTSS, test taking, show score and personal information. Test taking associates with Qmanager, which manages and creates tests for the student. Userupdate UI has dependency with userinfoBean. Instructor can also browse the main menu, which includes table of content manager, score manager, question manager and knowledge base manager. Table of content manager has direct association with TOCadd, TOCupdate and TOCdel. Question manager also associates with addQuestionG and EZ question manager. These two classes create new question to WISE system. Knowledge base manager associates with kbase_creator, kbase_delete and
Knowledge base system is very important for questions that use Java program to create question. Question manager will use knowledge base information to create test. Every class associates with connectsq1 to access the database to update, insert, delete or query.
Figure 2.6. Project Class Diagram
2.1.3 Database Design

Based on the class diagram, WISE database design uses relational database model. Relational database design defines database as a series of related tables. WISE database has been normalized for performance and logical error reduction. The following physical WISE database tables show the category detail entities for this project.

Table 2.11. User Information

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Userid</td>
<td>Varchar(9)</td>
<td>User login id</td>
</tr>
<tr>
<td>Password</td>
<td>Varchar(20)</td>
<td>Password for userlogin</td>
</tr>
<tr>
<td>Firstname</td>
<td>Varchar(30)</td>
<td>User’s firstname</td>
</tr>
<tr>
<td>Lastname</td>
<td>Varchar(30)</td>
<td>User’s lastname</td>
</tr>
<tr>
<td>Email</td>
<td>Varchar(30)</td>
<td>User’s email for contact</td>
</tr>
<tr>
<td>Login type</td>
<td>Enum</td>
<td>Type of user login: “S” for student and “P” for professor</td>
</tr>
</tbody>
</table>

Table 2.12. Tutorial

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tutorial_ch</td>
<td>Integer(2)</td>
<td>Chapter to study</td>
</tr>
<tr>
<td>Exam_ready</td>
<td>Enum of“Y” and “N”</td>
<td>Status for examination creation</td>
</tr>
<tr>
<td>Exam_selected</td>
<td>Integer(2)</td>
<td>Chapter for testing</td>
</tr>
</tbody>
</table>
### Table 2.13. Table of Content

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter_id</td>
<td>Integer(2)</td>
<td>Assign chapter no</td>
</tr>
<tr>
<td>Section_id</td>
<td>Integer(2)</td>
<td>Assign section no</td>
</tr>
<tr>
<td>Content</td>
<td>Varchar(50)</td>
<td>Name of that section</td>
</tr>
<tr>
<td>No_of_question</td>
<td>Integer(3)</td>
<td>Number of question in that section</td>
</tr>
</tbody>
</table>

### Table 2.14. Exam Control

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter_id</td>
<td>Integer(2)</td>
<td>Chapter no</td>
</tr>
<tr>
<td>Total_question</td>
<td>Integer(3)</td>
<td>Total question in that chapter</td>
</tr>
<tr>
<td>Percent_pass</td>
<td>Integer(3)</td>
<td>Percent to pass exam in this chapter</td>
</tr>
</tbody>
</table>
Table 2.15. Knowledge Base

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample_no</td>
<td>Integer(9)</td>
<td>Assign knowledge base number</td>
</tr>
<tr>
<td>Question_sample</td>
<td>Varchar(200)</td>
<td>Knowledge base question</td>
</tr>
<tr>
<td>Param1</td>
<td>Varchar(50)</td>
<td>Parameter no1 that need for calculate result</td>
</tr>
<tr>
<td>Param2</td>
<td>Varchar(50)</td>
<td>Parameter no2 that need for calculate result</td>
</tr>
<tr>
<td>Param3</td>
<td>Varchar(50)</td>
<td>Parameter no3 that need for calculate result</td>
</tr>
<tr>
<td>Param4</td>
<td>Varchar(50)</td>
<td>Parameter no4 that need for calculate result</td>
</tr>
<tr>
<td>Param5</td>
<td>Varchar(50)</td>
<td>Parameter no5 that need for calculate result</td>
</tr>
<tr>
<td>Sample_choice1</td>
<td>Varchar(50)</td>
<td>Sample Choice A. An example for instructor understanding what kind of answer for this question</td>
</tr>
<tr>
<td>Sample_choice2</td>
<td>Varchar(50)</td>
<td>Sample Choice B. An example for instructor understanding what kind of answer for this question</td>
</tr>
<tr>
<td>Sample_choice3</td>
<td>Varchar(50)</td>
<td>Sample Choice C. An example for instructor understanding what kind of answer for this question</td>
</tr>
<tr>
<td>Sample_choice4</td>
<td>Varchar(50)</td>
<td>Sample Choice D. An example for instructor understanding what kind of answer for this question</td>
</tr>
</tbody>
</table>
### Table 2.16. Question

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample_choice5</td>
<td>Varchar(50)</td>
<td>Sample Choice E. An example for instructor understanding what kind of answer for this question</td>
</tr>
<tr>
<td>Function_call</td>
<td>Varchar(50)</td>
<td>Name of function to calculate and create answer</td>
</tr>
<tr>
<td>Question_topic</td>
<td>Varchar(30)</td>
<td>What sorting topic this question belong to</td>
</tr>
<tr>
<td>Chapter_id</td>
<td>Integer(2)</td>
<td>Assign chapter number for this question</td>
</tr>
<tr>
<td>Section_id</td>
<td>Integer(2)</td>
<td>Assign section number for this question</td>
</tr>
<tr>
<td>Question_no</td>
<td>Integer(5)</td>
<td>Assign question number based on how many question in that chapter and section now</td>
</tr>
<tr>
<td>Question_id</td>
<td>Varchar(6)</td>
<td>Combination of the chapter_id plus section_id plus question_id as foreign key.</td>
</tr>
<tr>
<td>Question_type</td>
<td>Enum &quot;G&quot; or &quot;J&quot;</td>
<td>Type of question &quot;G&quot; for general, which means no calculation question or &quot;J&quot; for java question which must calculate answer</td>
</tr>
</tbody>
</table>
### Table 2.17. Question General

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question_id</td>
<td>Integer(3)</td>
<td>Question_id that assign in question table</td>
</tr>
<tr>
<td>Question</td>
<td>Varchar(200)</td>
<td>Question content</td>
</tr>
<tr>
<td>ChoiceA</td>
<td>Varchar(100)</td>
<td>The correct answer</td>
</tr>
<tr>
<td>ChoiceB</td>
<td>Varchar(100)</td>
<td>Multiple choice B</td>
</tr>
<tr>
<td>ChoiceC</td>
<td>Varchar(100)</td>
<td>Multiple choice C</td>
</tr>
<tr>
<td>ChoiceD</td>
<td>Varchar(100)</td>
<td>Multiple choice D</td>
</tr>
<tr>
<td>ChoiceE</td>
<td>Varchar(100)</td>
<td>Multiple choice E</td>
</tr>
</tbody>
</table>

### Table 2.18. Question Java

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question_id</td>
<td>Varchar(6)</td>
<td>Question_id that assigned in question table</td>
</tr>
<tr>
<td>Question</td>
<td>Varchar(200)</td>
<td>Question content</td>
</tr>
<tr>
<td>No_of_parameter</td>
<td>Integer(1)</td>
<td>Number of parameter used for calculate result</td>
</tr>
<tr>
<td>Function_call</td>
<td>Varchar(50)</td>
<td>Function that use for calculate result which come from knowledge base table</td>
</tr>
<tr>
<td>Answer_type</td>
<td>Varchar(10)</td>
<td>Show answer type of “auto”, which let system autocalculate answer or “define”, which let user define answers.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Choice1</td>
<td>Varchar(100)</td>
<td>If answer type is define, correct answer is defined here.</td>
</tr>
<tr>
<td>Choice2</td>
<td>Varchar(100)</td>
<td>If answer type is define, second choice is defined here.</td>
</tr>
<tr>
<td>Choice3</td>
<td>Varchar(100)</td>
<td>If answer type is define, third choice is defined here.</td>
</tr>
<tr>
<td>Choice4</td>
<td>Varchar(100)</td>
<td>If answer type is define, forth choice is defined here.</td>
</tr>
<tr>
<td>Choice5</td>
<td>Varchar(100)</td>
<td>If answer type is define, fifth choice is defined here.</td>
</tr>
</tbody>
</table>

Table 2.19. Parameter

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question_id</td>
<td>Varchar(6)</td>
<td>Question id that assigned in question table</td>
</tr>
<tr>
<td>Parameter_id</td>
<td>Varchar(6)</td>
<td>Assigned parameter number</td>
</tr>
<tr>
<td>Param_name</td>
<td>Varchar(20)</td>
<td>Parameter name</td>
</tr>
<tr>
<td>Param_define</td>
<td>Varchar(20)</td>
<td>Assign either &quot;auto random&quot; or &quot;user define&quot;</td>
</tr>
<tr>
<td>Param_value</td>
<td>Varchar(80)</td>
<td>If param_define set to &quot;user define&quot;, user define value is stored here</td>
</tr>
<tr>
<td>Min_number</td>
<td>Varchar(5)</td>
<td>If param_define set to &quot;autorandom&quot;, minimum number must be set to random number</td>
</tr>
<tr>
<td>Field Name</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Max_number</td>
<td>Varchar(5)</td>
<td>If param_define set to &quot;autorandom&quot;, minimum number must be set to random number</td>
</tr>
<tr>
<td>Show_value</td>
<td>Enum value &quot;ON&quot; or &quot;OFF&quot;</td>
<td>Choosing to show parameter value on screen and then used them for create answer(on) or not show on screen but use for create answer</td>
</tr>
</tbody>
</table>

Table 2.20. Score Table

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score_id</td>
<td>Integer(9)</td>
<td>Auto increment number</td>
</tr>
<tr>
<td>Userid</td>
<td>Varchar(9)</td>
<td>User id</td>
</tr>
<tr>
<td>Score</td>
<td>Varchar(10)</td>
<td>Score that user got</td>
</tr>
<tr>
<td>Status</td>
<td>Enum type “pass” or “not pass”</td>
<td>Student pass or not pass based on percent pass of exam control table</td>
</tr>
<tr>
<td>Chapter_test</td>
<td>Integer(2)</td>
<td>Chapter that student tested</td>
</tr>
<tr>
<td>Date_test</td>
<td>Varchar(30)</td>
<td>Date and time stamp after taking test</td>
</tr>
</tbody>
</table>

2.2 Detailed Design

Detailed design shows the logical algorithms. In other words, detailed design shows the program instruction in plain English language. The developer gets benefit from
detailed design since it is easier to read and detect error. As a result, the developer who wants to extend WISE system can enhance it faster, saving time and saving budget. The next section shows the covering all the classes that are listed in the class diagram on Figure 2.6.

2.2.1 Connect Database Class

<table>
<thead>
<tr>
<th>Class name: connectsql</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose: Connect to physical database</td>
</tr>
<tr>
<td>Begin class</td>
</tr>
<tr>
<td>Mydriver=path of JDBC driver</td>
</tr>
<tr>
<td>MyURL=path of physical database</td>
</tr>
<tr>
<td>Function Makeconnection: no return value</td>
</tr>
<tr>
<td>/* connect physical database */</td>
</tr>
<tr>
<td>Begin</td>
</tr>
<tr>
<td>Activate driver</td>
</tr>
<tr>
<td>Establish connection to physical database</td>
</tr>
<tr>
<td>End</td>
</tr>
<tr>
<td>Function shutdown: no return value</td>
</tr>
<tr>
<td>/* shutdown connection */</td>
</tr>
<tr>
<td>Begin</td>
</tr>
<tr>
<td>Close connection</td>
</tr>
<tr>
<td>End</td>
</tr>
<tr>
<td>End class</td>
</tr>
</tbody>
</table>

Figure 2.7. Connect Database Class
2.2.2 Query Bean Class

Class name: queryBean
Purpose: for query information from database

Class begin with extends connectsql class
ResultSet: myresultset=null
Statement: stmt=null

Function setConnection: no return value
/* link to connectsql to connect database */
Begin
    Connectsql.makeConnection();
End

Function getNextRecord: return Boolean
/* check next record in database */
begin
    return if there is next record
end

function getColumnString: return string
/* get string value from specific column */
parameter in: string:clmn
begin
    return string value of that column.
End

function getColumnInt: return string
/* get integer value from specific column */
parameter in: string:clmn
begin
    return integer value of that column.
end

function isRecord: return Boolean
/* query database */
parameter in: string:query
begin
    set myresultset=null
    prepare statement
    check if that weather query get result
    return weather query has result (not equal null)
end

function moveFirst: no return value
/* move cursor to first record */

Figure 2.8. Query Bean Class

61
function moveLast: no return value
  /* move cursor to last record */
  begin
    move cursor to the end of database
    return if move cursor to last record(previous) is ok
  end

function cleanup: no return value
  /* clear all statement values */
  begin
    close statement
  end

function setShutDown: no return value
  /* shutdown this connection */
  begin
    activate close statement
    disconnect database
  end

class end

Figure 2.8 Query Bean Class (continued)
### 2.2.3 Data Manipulation Class

**Class name:** InsDelUpdBean extends connectsql  
**Purpose:** to insert, delete and update data to database

```
Class begin  
Statement stmt=null  
Int rowaffect=0  

Function InsDelUpd: a construction class  

Function setConnection: no return value  
/* connect database */  
Begin  
    Connectsql.makeConnection  
End  

Function setInsDelUpd: return int  
/* prepare statement for ins,del or upd */  
parameter in: query  
begin  
    set myquery=query  
    prepare statement before query  
    execute query statement, get number of row effect  
    return number of row that effect from query statement  
end  

function setInsert: return int  
/* prepare insertion */  
parameter in: query  
begin  
    call setInsDelUpd with query parameter  
    return rowaffect  
end  

function setDelete: return int  
/* prepare deletion */  
parameter in: query  
begin  
    call setInsDelUpd with query parameter  
    return rowaffect  
end  
```

Figure 2.9. Data Manipulation Class

63
function setUpdate: return int
/* prepare update */
parameter in: query
begin
    call setInsDelUpd with query parameter
    return rowaffect
end

function cleanup: no return value
/* clear all statement values */
begin
    close statement
end

function setShutDown: no return value
/* shutdown this connection */
begin
    activate close statement
    disconnect database
end

class end
2.2.4 Connect File Class

Class name: connectfile
Purpose: connect to read or write file

Class begin

Function connectfile: a construction class

Function openwritefile: no return value
/* write html heading to file Q and file A */
Parameter in: String Q,A;
Begin
    Connect filename(Q);
    Connect filename(A);
    Write html heading to file Q;
    Write test answer to file A;
End;

Function closewritefile: no return value
/* write html ending to file Q and file A */
Parameter in: String Q,A;
Begin
    Write html ending to file Q;
    Close file Q;
    Write "0" to file A; // 0=end of file
    Close file A;
End;

Function openReadfile: no return value
/* connect file A */
Parameter in: String A;
Begin
    Connect filename(A);
End;

Function closeReadfile: no return value
/* connect file A */
Parameter in: String A;
Begin
    Close connect filename(A);
End;
End class

Figure 2.10. Connect File Class
2.2.5 File Bean Class

Class name: fileBean extends connectfile
Purpose: write data to file

Function filebean: a construction class

Function getRadioForm: return string
/* return radio html button with choice content */
Parameter in: string choice
   Int I
Begin
   Return radio html button with value=choice
End

Function makeHTML: no return value
Parameter in:
   Int I
   String q,a,b,c,d,e,ans
Begin
   Connectfile
   Write test question and answer to file
End
End class

Figure 2.11. File Bean Class
2.2.6 Read File Bean Class

Class name: readFileBean extends connectfile
Purpose: read answer from file A

Function readFilebean: a construction class

Function readcorrectanswer: return vector of answer
Begin
  Prepare file to read
  While not end of file (not equal to 0) do
    Read answer and put it to vector
  End while
End

End class

Figure 2.12. Read File Bean Class
2.2.7 Score Bean Class

Class name: scoreBean
Purpose: check score after student take test

Begin class

Function scorebean: a construction class

Function setAnswer: no return value
/* check answer */
Parameter in
Vector student_answer, correct_answer
Int percent_pass
Begin
  Read student answers to vector student_answer
  Read correct answers to vector correct_answer
  For loop of size of vector
    If (student answer= correct answer)
      Count score
  End for loop
  Find passing score
End

Function getScore: return int
/* get score */
Begin
  Return score
End

Function ispassChapter: return Boolean
/* check student pass exam or not */
begin
  check score with passing score
  return true if pass
  else return false
end

end class

Figure 2.13. Score Bean Class
2.2.8 Create Choice Class

Class name: createchoice
Purpose: create multiple choice base on correct answer

Begin class
Function createchoice: a construction class

Function makechoiceArray: no return value
/* create choices from correct array answer */
Parameter in:
Int value[]
String choice[]
Begin
   Int I, first_half, second_half

   Get first half of array to first_half
   Get second half of array to second_half
   Create correct choice
   Create second choice
   Create third choice
   Create forth choice
   Create fifth choice

End

End class

Figure 2.14. Create Choice Class
2.2.9 Shuffle Class

Class name: shuffle
Purpose: shuffle multiple choices

Begin class
Function shuffle: a construction class

Function makeShuffle: return string
Parameter in: string choice[]
Begin
    Int k, rand
    Int ans=0
    Boolean flag=true

    For k=1 to 5
        While flag==true do
            Random number with in 5 choices
            If random number never used then
                Flag=false
            Switch (random number)
            If 1: a=choice[random-1]
                Clear choice[random-1]
            If 2: b=choice[random-1]
                Clear choice[random-1]
            If 3: c=choice[random-1]
                Clear choice[random-1]
            If 4: d=choice[random-1]
                Clear choice[random-1]
            If 5: e=choice[random-1]
                Clear choice[random-1]
            End switch
            End while
        If random=1 // find correct answer after shuffle
            Ans=k
        End if
        Switch(ans)
        1: answer=choice a
        2: answer=choice b
        3: answer=choice c
        4: answer=choice d
        5: answer=choice e
        flag=true;
    End if
    End for
    random number between 1 to 3
    if random=3 set choice e=none of above
    return answer
End
End class

Figure 2.15. Shuffle Class
2.2.10 Random Number Class

Class name: randomnumber
Purpose: random number between given two numbers

Begin class
Function randomNumber: a construction class

Function getNumber: return int
Parameter in
Int min,max;
Begin
  Return random number between these two given numbers
End

End class

Figure 2.16. Random Number Class
2.2.11 Add Question General Class

Class name: addquestionG
Purpose: add general question to database

Begin class
Function addquestionG: a construction class

Function checkVariable: return Boolean
/* check variable from addQuestionG form */
paramter in:
string q,a,b,c,d,e,topic
begin
  if q has no value then error question found
  if a has no value then error choice A found
  if b has no value then error choice B found
  if c has no value then error choice C found
  if d has no value then error choice D found
  if e has no value then error choice E found
  if topic has no value then error topic found
  return error variable if error found
end

function setQuestion1:return Boolean
/* add new question to database */
parameter in
String q,a,b,c,d,e,topic
Begin
  SetQuestion_no(topic)
  Connect db
  Set update to add 1 to question on that chapter
  Update information and check row affect
  Set insert of question number, question type info
  Insert statement to table question
  Set insert question, answer statement
  Insert statement to table question_general
  Disconnect db
  Return true if success
Else return false
End

Figure 2.17. Add Question General Class
Function setQuestion2: return Boolean
/* delete question from database table */
Parameter in:
String q, a, b, c, d, e, topic, question_id
Begin
Set delete statement
SetQuestion1(q, a, b, c, d, e, topic)
Delete at row(question_id,"G")
If successful delete return true
Else return false
End

Function setQuestion_no: no return value
/* prepare question number for insert before
insert into table */
parameter in:
string topic
begin
string query
string name
connect to db
select content, chapter_id and section_id from topic information.
Select total number of question in that chapter
Set question number by concatenate
chapter_id+section_id +(total question+1)
End

Function getError: return string
Begin
Return error if any
End
End class

Figure 2.17. Add Question General Class (continued)
2.2.12 Add Question Java Class

Class name: addquestionJ
Purpose: add JAVA question to database

Begin class
Function addquestionJ: a construction class

Function setQuestionInfo: no return value
Parameter in:
String sortname
Begin
SortQuery= new queryBean
Set select statement from tableofcontent table
Connect to db
If (query has record) the
Get chapter_id to chapter
Get section_id to section
Get number of question to total
End if
Disconnect db
End

Function setQuestion: return Boolean
/* insert java question to db */
parameter in:
string question, paramno, function_call
string answer, p[][][], sortname
begin
init value of roweffect1=0
init value of roweffect2=0
init value of roweffect3=0
init value of roweffect4=0
int number_of_question
string question_id
string addtoparameteretable=""

setquestioninfo(sortname)
number_of_question=total
connect to db for add
set update number of question+1 to tableofcontent table
set insert to question table
(chapter, section, question_id, "J");

Figure 2.18. Add Question Java Class
if error not true(=1) then
    set insert to question_java table
    set insert to parameter table
    disconnect database
    if no error return true
    else return false
end

function setQuestionEdit: return Boolean
parameter in:
    string answer, choice[], p[][], sortname, old_question_id
begin
    add new edit question record to database
    delete the old question record from database
end

end class

Figure 2.18. Add Question Java Class (continued)
2.2.13 Question Retrieval Manager Class

Class name: Qmanager
Purpose: this class manage and create question test for student

Begin class
Function Qmanager: a constructor class

Function getQuestion: return string
/* random to get question either general question or java question */
Parameter in:
Int ch, sections
String choices[]
Begin
Int rand1
Int number_of_question=1
String question_type=""
String question_number=""
String query_question_table=""

QueryBean db=new queryBean()
QuestionG qg=new questionG()
QuestionJ qj=new questionJ()
Db set connection
For(;;)
Query_number= select no_of_question
If(db.isRecord(query_number))
Get no_of_question to number_of_question
If(number_of_question equal 0)
Random new section
Else break
End for
Random number and put to rand1
Set query to select question from question_id that get from chapter+section+rand1
Set query to select type of that question either "G" or "J"
If (type="G")
Question=qg.getQuestion(question_id,choices)
Else
Question=qj.getQuestion(question_id,choices)
Shuffle sl=new shuffle()
Answer=make shuffle answer(choices)
Return question

Figure 2.19. Question Retrieval Class
Function getAnswer: return String
/* get answer */
Begin
   Return answer
End

Function getdupinfo: return string
/* send question_id to check that this question is already in the test, yes question duplicate or no question will be in the test */
Begin
   Return (question_id)
End

End class

Figure 2.19. Question Retrieval Class (continued)
### 2.2.14 Question General Class

<table>
<thead>
<tr>
<th>Class name: questionG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose: retrieve general question from database</td>
</tr>
<tr>
<td>(no calculation question)</td>
</tr>
</tbody>
</table>

Begin class
Function questionG: a constructor class

Function getQuestion: string
/*function to get question and answer before shuffle */
begin
    queryBean db=new queryBean()
    db sets database connection
    query= select question from question_id
    if (record found) then
        get question to question valriable
        get choice[0] to choiceA
        get choice[1] to choiceB
        get choice[2] to choiceC
        get choice[3] to choiceD
        get choice[4] to choiceE
    end if
    db set disconnection
    return question
end
end class

Figure 2.20. Question General Class
2.2.15 Question Java Class

Class name: questionJ
Purpose: retrieve JAVA question from database
           (the calculation question type)

Begin class
Function questionJ: a constructor class

Function getQuestion: return string value
Parameter in:
String Question_id,choice[]
Begin
QueryBean qj=qj.queryBean()
Qj set db connection
Query=select * from question_java
If (there is record)
  Get question content to question variable
  Get no_of_param to no_of_param variable
  Get function_call to function_call variable
  Get answer_type to answer_type variable
  If (answer_type not equal user define)
    Choice[0]=get value from column choice1
    Choice[1]=get value from column choice2
    Choice[2]=get value from column choice3
    Choice[3]=get value from column choice4
    Choice[4]=get value from column choice5
  End if
End if
Query =select * from paramter by question_id
If (there is record)
  While (nextRecord)
    Parameters[I][0]=get value from param_id
    Parameters[I][1]=get value from param_name
    Parameters[I][2]=get value from param_define
    Parameters[I][3]=get value from param_value
    Parameters[I][4]=get value from min_number
    Parameters[I][5]=get value from max_number
    Parameters[I][6]=get value from show_value
    Increment I by 1
  End if
Disconnect database
If(answer_type=auto)
  CreateAnswer(choices)
Return question

Figure 2.21. Question Java Class
Function createQuestion: return String value
/* create full question */
begin
    for (i = 0 to no_of_parameter-1) 
        find position "[p" in question
        save postion in index2
        tempq= cut all question up to index2 position
        then select parameter
        read if temp_v=[p1] then
            param_value=get param1
        read if temp_v=[p2] then
            param_value=get param2
        read if temp_v=[p3] then
            param_value=get param3
        read if temp_v=[p4] then
            param_value=get param4
        read if temp_v=[p5] then
            param_value=get param5
        question_set=concat temp_q and param_value
        index2=index2
    end for
    question_set=concat question set+ substring index1
    return full question
end

function getParameter: return String
/* prepare set parameter */
parameter in: string temp_v
begin
    String value="";
    if (temp_v equals("[p1]"))
        value=setParameter("1");
    if (temp_v equals("[p2]"))
        value=setParameter("2");
    if (temp_v equals("[p3]"))
        value=setParameter("3");
    if (temp_v equals("[p4]"))
        value=setParameter("5");
    if (temp_v equals("[p5]"))
        value=setParameter("5");
    return value;
end

Figure 2.21. Question Java Class (continued)
function setParameter: return string
/* set parameter */
parameter in:
string id
begin
    for (i=0 to i<no_of_parameter) do
        if(id equals(parameters[i][0])) then
            if(parameters[i][2] equals("user define")) then
                if(parameters[i][1].equals("array")) then
                    value=getArrayUserDefine(parameters[i][3])
                end if
                p[i]=parseInt(temp);
                value=temp;
            end if
        else if(parameters[i][2] equals("auto random"))
            if(parameters[i][1].equals("array"))
                size=parameters[i][3]
                min=parameters[i][4];
                max=parameters[i][5]
                value=getArrayRandom(size, min, max)
            else
                min=parameters[i][4]
                max=parameters[i][5]
                p[i]=getValueRandom(min, max)
                value=""+p[i]
            end if
        end if
    end if
    if(parameters[i][6] equals("OFF")) value=""
end
return value;
end

function getarrayuserDefine: return string value
/* get array that user define */
parameter in
string array_userdefine
begin
    add "!" to array_userdefine to set end point
    ch=array_userdefine.charAt(i);
    while (ch!='!'){
        if(Character.isDigit(ch))
            temp+=""+ch
        else{

Figure 2.21. Question Java (continued)
if (!temp.equals(""))
    v.addElement(temp)
    temp=""}

i=i+1
ch=array_userdefine.charAt(i)
arrays=new int[v.size()]
for(i=0;i<v.size();i++)
    temp=v.elementAt(i).toString()
    arrays[i]=Integer.parseInt(temp)
    value+=temp+" "
}
return value end

function getArrayRandom: return string
/* get array random */
parameter in:
string size_str, min_str, max_str
begin
    randomNumber=rn.new randomNumber
    min=parseInt(min_str)
    max=parseInt(max_str)
    try size=parseInt(size_str)
    catch error if size=8
    arrays= new array[size]
    for (I=0 to size-1) do
        array[I]=random number between min and max
    end for
    return array_value
end

function getValueRandom: return int
/* get random value */
begin
    min=parseInt(min_str)
    max=parseInt(max_str)
    return (get random number between min and max)
end

function createAnswer: no return value
parameter in:
String choices[]
Begin
    Sortgen.sortmain(arrays,p,function_call,choices)
End
End class

Figure 2.21. Question Java (continued)
2.2.16 Table of Content Check Class

Class name: TOCcheck
Purpose: check, add, modify and delete table of content information.

Begin class
Function TOCcheck: a constructor class

Function setChapter: no return value
Parameter in:
String chapter
Begin
    This.chapter = chapter
End

Function setSection: no return value
Parameter in:
String section
Begin
    This.section = section
End

Function setContent: no return value
Parameter in:
String content
Begin
    This.content = content
End

Function startCheck: return Boolean value
Begin
    Try{
        Chapter_id=parsenInt(chapter)
    }catch (if number error found){
        found=true;
        error_message="chapter must be integer"
    }
    Try{
        section_id=parsenInt(section)
    }catch (if number error found){
        found=true;
        error_message="section must be integer"
    }
    return found_error;
End

Figure 2.22. Table of Content Check Class
Function geterror: return string value
Begin
    return error_message
End

Function addContent: return boolean value
Begin
    Set query= insert to table-of-content table values
    (chapter_id, section_id, content)
    InsDelUpd add=new InsDelUpdBean()
    Add.setconnection()
    Add.setInsert(query)
    Add.disconnect
    If error of duplicate record
        Error_message="duplicate"
        Return true
    If insert ok then
        Return true
End

Function delContent: return Boolean value
Begin
    try
        Set delete statement to delete by content value
        InsDelUpdBean del=new InsDelUpdBean()
        Del.setConnection()
        Del.setDelete(delete)
        Del.setshutdown()
        Catch (error deletion){
            Error_message="error deletion"
            Return true
        If deletion is ok then
            Return true
        End
    End
End class

Figure 2.22. Table of Content Check Class (continued)
Class name: KnowledgeBaseCheck
Purpose: check, add, modify for knowledge base

Begin class
Function knowledgbaseCheck: a constructor class

Function setQuestion: no return value
Parameter in: string question
Begin
   This.question=question
End

Function setChoiceA: no return value
Parameter in: string choiceA
Begin
   This.choiceA=choiceA
End

Function setChoiceB: no return value
Parameter in: string choiceB
Begin
   This.choiceB=choiceB
End

Function setChoiceC: no return value
Parameter in: string choiceC
Begin
   This.choiceC=choiceC
End

Function setChoiceD: no return value
Parameter in: string choiceD
Begin
   This.choiceD=choiceD
End

Function setChoiceE: no return value
Parameter in: string choiceE
Begin
   This.choiceE=choiceE
End

Figure 2.23. Knowledge Base Check Class
Function setP1: no return value  
Parameter in: string P1  
Begin  
This.P1 = P1  
End  

Function setP2: no return value  
Parameter in: string P2  
Begin  
This.P2 = P2  
End  

Function setP3: no return value  
Parameter in: string P3  
Begin  
This.P3 = P3  
End  

Function setP4: no return value  
Parameter in: string P4  
Begin  
This.P4 = P4  
End  

Function setP5: no return value  
Parameter in: string P5  
Begin  
This.P5 = P5  
End  

Function setfunction_call: no return value  
Parameter in: string function_call  
Begin  
This.function_call = function_all  
End  

Function setTopic: no return value  
Parameter in: string topic  
Begin  
This.topic = topic  
End

Figure 2.23. Knowledge Base Check Class (continued)
Function checkKnowledgeBase: return boolean value
Begin
   if (question.length() == 0) {
      found_error = true;
      errors = Question field must have value
   }
   if (choiceA.length() == 0) {
      found_error = true;
      errors = choiceA field must have value
   }
   if (choiceB.length() == 0) {
      found_error = true;
      errors = choiceB field must have value
   }
   if (choiceC.length() == 0) {
      found_error = true;
      errors = choiceC field must have value
   }
   if (choiceD.length() == 0) {
      found_error = true;
      errors = choiceD field must have value
   }
   if (choiceE.length() == 0) {
      found_error = true;
      errors = choiceE field must have value
   }
   if (function_call.length() == 0) {
      found_error = true;
      errors = Function call field must have value
   }
   if (topic.length() == 0) {
      found_error = true;
      errors = Topic: Please choose topic
   }
   return found_error;
End

Function addSample: return Boolean
   /* add knowledgebase to database */
   begin
      InsDelUpdBean add = new InsDelUpdBean()
      Query = insert into knowledgebase
                  values(question, p1, p2, p3, p4, p5, choiceA, choiceB,
                         choiceC, choiceD, choiceE)
      Add.setConnection()
      Add.insert()
      If no error then
         Return true
      Else return false
      End
   Function getError: return string value
   Begin
      Return errors
   End
End class

Figure 2.23. Knowledge Base Check Class (continued)
2.2.18 Knowledge Base Deletion Class

Class name: Kbase_delete
Purpose: delete for knowledge base

Begin class
Function kbase_delete: a constructor class

Function delete_row: return Boolean value
Parameter in: string question

Begin
QueryBean check=new queryBean()
This.question=question
Delete_query= delete from knowledgebase where question_sample=question;
Del.setConnection()
Setdelete(delete)
Del.shutdown()
If delete is ok then return true
Else return false
End
End class

Figure 2.24. Knowledge Base Deletion Class
2.2.19 User Information Class

Class name: UserInfoBean
Purpose: check, add, delete, update for users

Begin class
Function userinfobean: a constructor class

Function serUsername: no return value
Parameter in: string username
Begin
    This.username=username
End

Function seroldpassword: no return value
Parameter in: string oldpassword
Begin
    This.oldpassword=oldpassword
End

Function serpassword1: no return value
Parameter in: string password1
Begin
    This.password1=password1
End

Function serpassword2: no return value
Parameter in: string password2
Begin
    This.password2=password2
End

Function serUser: no return value
Parameter in: string user
Begin
    This.user=user
End

Function serlast: no return value
Parameter in: string last
Begin
    This.last=last
End

Figure 2.25. User Information Class
Function `serEmail`: no return value  
Parameter in: string `email`  
Begin  
   This.email=email  
End

Function `startCheck`: no return value  
Parameter in: string `check`  
Begin  
   if (!check.equals("upd"))  
      if (username.length()==0)  
         error_found=true  
         error_message=Username field must have value  
      endif  
   endif  
   if (check.equals("upd"))  
      if (oldpassword.length()==0)  
         error_found=true  
         error_message=Oldpassword field must have value  
      endif  
   endif  
   if (password1.length()==0)  
      error_found=true  
      error_message=Password field must have value  
   endif  
   if (password2.length()==0) then error_found=true;  
      error_message=Password field must have value  
   endif  
   if (check.equals(""))  
      if (user.length()==0)  
         error_found=true  
         error_message+="<li>User field must have value  
      endif  
   endif  
   if (first.length()==0)  
      error_found=true  
      error_message=Firstname field must have value  
   endif  
   if (last.length()==0){  
      error_found=true  
      error_message=Lastname field must have value  
   }endif  
   if (email.length()==0){  
      error_found=true  
      error_message=Email field must have value  
   }else  
      index=email.indexOf("@");  
      if (index==-1)

Figure 2.25 User Information Class (continued)
error_message=Invalid Email
endif

if (error_found==false)
  if (check.equals("S"))
    checkValid1();
    status="S";
    else if (user.equals("Student"))
      checkValid1();
      status="S";
    endif
  else if (user.equals("Instructor"))
    checkValid2()
    status="P"
  endif
endif
return error_message;
End

Function checkValid1: no return value
/* validate student account */
Begin
  Try
    Ssn_bound=parslnt(username)
    If((ssn is not between 1000000000-900000000)
      Error_message="invalid SSN"
    Catch (numberformatexception nfe)
      Error_message=SSN contain 9 digits
    End
  )
Function checkValid2: no return value
/* validate professor account */
Begin
  Try{
    Ssn_bound=parslnt(username)
    If((ssn contains number)
      Error_message="professor account should not all number"
    )
    )Catch{ (numberformatexception nfe)
      If(username.length > 8)
        Error=username contain 8 cahracter
    }passwordcheck()
  )
End

Function passwordCheck: no return value

Figure 2.25 User Information Class (continued)
Begin
If(password1 not equals password2) then
   Error=password are not the same
End

Function addUser: no return value
Begin
Stmt=insert all information to userinfo table
InsUpdDelBean add=new InsupdDelBean
Try{
   Add.setConnection()
   Setinsert(stmt)
   Add.setShutDown()
}catch{ (sqlexception sql)
   error=username already taken
   error+=Please try again
   return false
}
if (add success) then return true
else return false
End

Function delUser: no return value
Begin
Stmt=delete from userinfo table by username
InsUpdDelBean del=new InsupdDelBean
Try{
   Add.setConnection()
   Setdelete(stmt)
   Add.setShutDown()
}catch{ (sqlexception sql)
   error=username does not exist
   error+=Please try again
   return false
}
if (del success) then return true
else return false
End

Function userChange: return boolean value
Begin
if (first.length()==0){
   error_found=true
   error_message=Firstname field must have value
}
if (last.length()==0){
   error_found=true
   error_message=Lastname field must have value

Figure 2.25. User Information Class (continued)
if (email.length() == 0) {
    error_found = true
    error_message = Email field must have value
} else {
    index = email.indexOf("@")
    if (index == -1) {
        error_found = true
        error_message = Invalid Email
    }
}
return error_found;

Function passwordChange: return boolean value
Begin
    if (oldpassword.length() == 0) {
        error_found = true
        error_message = Old password field must have value
    } else {
        if (password1.length() == 0) {
            error_found = true
            error_message = Password field must have value
        }
        if (password2.length() == 0) {
            error_found = true
            error_message = Confirm Password field must have value
        }
        if (error_found == false) {
            passwordCheck()
            if (error_message.length() != 0)
                error_found = true
        }
    }
return error_found
End

Function updateUser: return boolean value
Begin
    if (user.equals("Instructor"))
        status = "P"
    else status = "S"
    stmt1 = "update userinfo set firstname='"+first+"', lastname='"+last+"', email='"+email+"', login_type='"+status+"'"
    InsDelUpdBean upd = new InsDelUpdBean()
    try {
        upd.setConnection()
    }
Figure 2.25 User Information Class (continued)
upd.setDelete(stmt1)
upd.setShutDown()
} catch (SQLException sql){
    error_message=Database Error
    return false
}
if (update success) return true
else return false
End

Function updatepassword: return boolean value
Begin
    if (user.equals("Instructor"))
        status="P"
    else status="S"
stmt2="update userinfo set password='"+password1+'"
InsDelUpdBean upd=new InsDelUpdBean();
try{
    upd.setConnection()
    upd.setDelete(stmt2)
    upd.setShutDown()
} catch (SQLException sql){
    error_message=Database Error
    return false
}
if (update success) return true
else return false
End

Function geterror: return string value
Begin
    Return error
End

Figure 2.25. User Information Class (continued)
CHAPTER THREE
TESTING

Testing or software validation is the testing process to ensure that the program as implemented meets the expectation of the user [3]. The purpose of system validation is to have assurance about the software quality and functionalities. This guarantees system performance and reliability also.

3.1 The Testing Process

There are three testing processes that are used in testing WISE system: unit testing, subsystem testing, and system testing.

Except for small computer programs, it is unrealistic to attempt to test systems using only unit testing. Large systems are built out of many small subsystems, which may themselves be built out of procedures. WISE system was tested by all three testing methods.

3.2 Unit Testing

Unit testing is the basic level of testing where individual components are tested to ensure that they operate correctly. These individual components can be object, class, program and etc. The following test rules
are defined to check that each component meets specification:

(a) Verify the handling of all valid input data type
(b) Verify the handling of error condition
(c) Check normal and abnormal program termination
(d) Check all buttons work as expected

Unit test offers the most effective way to detect problems when comparing with the other testing methods. On the other hand, this testing method cost more time and money to execute. The later testing methods will show an effective way to save time, which will be discussed in next two sections. The unit testing results of WISE system are shown in Table 3.1.

Table 3.1. Unit Test Results

<p>| Forms         | Tests Performed                                                                 | Results |
|---------------|*********************************************************************************|---------|
| Tutorial Applet | • Test all sorting algorithm applets that describe in last section             | Pass    |
| Authorization | • Verify handling valid data input                                            | Pass    |
|               | • Ensure all buttons and links work as expected                            |         |
| Student Sign Up | • Verify handling valid data input                                          | Pass    |
|               | • Ensure all buttons and links work as expected                            |         |</p>
<table>
<thead>
<tr>
<th>Forms</th>
<th>Tests Performed</th>
<th>Results</th>
</tr>
</thead>
</table>
| Password            | • Verify handling valid data input  
                          • Ensure all buttons and links work as expected | Pass    |
| Student Main Menu   | • Check all submit buttons                                                      | Pass    |
| Testing Selection   | • Check combo box  
                          • Check all buttons                                      | Pass    |
| Testing Page        | • Check all text appearance  
                          • Verify handling mouse input  
                          • Check submit button                                        | Pass    |
| Result Score Page   | • Verify content appearance  
                          • Check submit button                                           | Pass    |
| User Modification   | • Verify handling data input  
                          • Check button labels appearance  
                          • Check buttons work as expected                             | Pass    |
| Student feedback    | • Verify handling data input  
                          • Check selection box appearance  
                          • Check all buttons work as expected                         | Pass    |
| Instructor Main Menu| • Check all contents  
                          • Check all buttons and links work as expected               | Pass    |
| General Question Creator | • Verify handling data input  
                                          • Verify selection box area  
                                          • Check all submit button works as expected                    | Pass    |
<table>
<thead>
<tr>
<th>Forms</th>
<th>Tests Performed</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZ question creator Page 1</td>
<td>• Verify handling data selection&lt;br&gt;• Check all buttons work as expected</td>
<td>Pass</td>
</tr>
<tr>
<td>EZ question creator Page 2</td>
<td>• Verify handling data selection&lt;br&gt;• Check all buttons work as expected</td>
<td>Pass</td>
</tr>
<tr>
<td>EZ question creator Page 3</td>
<td>• Verify handling data input&lt;br&gt;• Verify handling data selection&lt;br&gt;• Check all buttons work as expected</td>
<td>Pass</td>
</tr>
<tr>
<td>EZ question creator Page 4</td>
<td>• Verify handling data input&lt;br&gt;• Verify handling data selection&lt;br&gt;• Check all buttons work as expected</td>
<td>Pass</td>
</tr>
<tr>
<td>EZ question management</td>
<td>• Check handling edit buttons&lt;br&gt;• Check handing delete buttons&lt;br&gt;• Check view content apperance</td>
<td>Pass</td>
</tr>
<tr>
<td>Knowledge Base creator</td>
<td>• Verify handling input data&lt;br&gt;• Check retrieval selection box information from database&lt;br&gt;• Check all buttons work as expected</td>
<td>Pass</td>
</tr>
<tr>
<td>Forms</td>
<td>Tests Performed</td>
<td>Results</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Knowledge Base Management</td>
<td>• Check response edit button</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• Check response delete button</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Check retrieval information from database</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Check confirm deletion</td>
<td></td>
</tr>
<tr>
<td>Score Management</td>
<td>• Check score retrieval by specific id</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• Check score retrieval by all id</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Check All buttons work as expected</td>
<td></td>
</tr>
<tr>
<td>Preference Page</td>
<td>• Check instructor’s information retrieval from database</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• Verify handling data input</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Check all submit buttons work as expected</td>
<td></td>
</tr>
<tr>
<td>Content Creator</td>
<td>• Verify handling data input</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• Check all buttons work as expected</td>
<td></td>
</tr>
<tr>
<td>Content Manager</td>
<td>• Verify table of content retrieval information from database</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• Check edit content response</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Check delete content response</td>
<td></td>
</tr>
<tr>
<td>System Administrator</td>
<td>• Verify all menu contents</td>
<td>Pass</td>
</tr>
<tr>
<td>Main Menu</td>
<td>• Check all links work as expected</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Check security feature</td>
<td></td>
</tr>
</tbody>
</table>
### 3.3 Subsystem Testing

Subsystem testing is the next step up in the testing process where all related units form a subsystem to do a certain task. Thus, the subsystem test process is useful for detecting interface errors and specific functions. Table 3.2 shows subsystem test results in detail.

<table>
<thead>
<tr>
<th>Forms</th>
<th>Tests Performed</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Creator</td>
<td>• Verify handling all data input</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• Check all buttons work properly</td>
<td></td>
</tr>
<tr>
<td>User Manager</td>
<td>• Verify user information retrieval from database</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• Check edit content response</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Check delete content response</td>
<td></td>
</tr>
<tr>
<td>Log Off</td>
<td>• Check logoff link work properly</td>
<td>Pass</td>
</tr>
</tbody>
</table>
Table 3.2. Subsystem Test Results

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Test performed</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTSS tutorial system</td>
<td>• Test all tutorial and applet of $O(n^2)$ and $O(n\log n)$</td>
<td>Pass</td>
</tr>
<tr>
<td>Question subsystem</td>
<td>• Test random question, create answer and link statistic</td>
<td>Pass</td>
</tr>
<tr>
<td>Knowledge Base subsystem</td>
<td>• Test create question, manage question database</td>
<td>Pass</td>
</tr>
<tr>
<td>User subsystem</td>
<td>• Test user by create, manage, update user</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• Test security on user account</td>
<td></td>
</tr>
<tr>
<td>Score subsystem</td>
<td>• Check and test grade to pass chapter.</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• Check score database and test score controller</td>
<td></td>
</tr>
</tbody>
</table>

3.4 System Testing

System testing is the testing process that uses real data, which the system is intended to manipulate, to test the system. First all subsystem will be integrated into one system. Then test the system by using a variety of data to see the overall result.

System testing WISE system begins with the following steps (Table 3.3):
Table 3.3. System Test Results

<table>
<thead>
<tr>
<th>System Testing</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Install WISE system into computer science server.</td>
<td>Pass</td>
</tr>
<tr>
<td>2. Start up all services such as JSP engine, MySQL database engine.</td>
<td>Pass</td>
</tr>
<tr>
<td>3. Running testing by using real data on all forms and reports.</td>
<td>Pass</td>
</tr>
</tbody>
</table>

3.5 Sample Session

The following section shows samples of how to use WISE system in the demonstration of tutorials, knowledge base, general question creator and EZ question creator.

3.5.1 Demonstration of Tutorial

Figure 3.1. Sorting Algorithms Tutorial
The format of the tutorial consists of four Web pages per sorting algorithms (see Figure 3.5.1), which include basic operation, sorting example, sorting animation applets and analysis of sorting algorithm. Figure 3.5.1 (1) shows basic operation in plain text. Figure 3.5.1(2) explains more details of the algorithm. Figure 3.5.1(3) shows the sorting animation. Students just click on "forward" buttons to start animation. Figure 3.5.1 (4) shows analysis of algorithm. Each analysis of algorithm explains the average case only.

3.5.2 General Question Creator

![Figure 3.2. General Question Creator](image)

General question creator (see Figure 3.5.2) tools help to create general questions. Figure 3.5.2 (1) shows the interface for creation of questions. After filling out all the information, The system saves it to the database and show the confirmation of the save operation.
3.5.3 Knowledge Base Creator

Figure 3.3. Testing Knowledge Base Creator

Figure 3.5.3 (1-2) shows fragments of the Java program that the instructor has to do to find the answer. It also shows another program fragment to define how to create choices. Figure 3.5.3 (3-4) shows the input question information, which inform other instructors that there is this question ready to create those exercises.
3.5.4 Easy Question Creator

EZ question creator is shown in Figure 3.5.4 (1-5). In step one, the instructor has to choose which sorting question and type of answer to create. Step two shows the knowledge base question based on step one. After selecting
the type of question in step two, the instructor can modify the question in step three. This question will later be used in the interactive exercise after students finished their tutorial. In step four, the instructor chooses between letting program create answers or having the user defined them. Step five shows all the question information that WISE collected from step 1-4. After clicking “finish” button, WISE saves all question information to the database.
CHAPTER FOUR
MAINTENANCE MANUAL

It is impossible to produce systems of any size, which do not need to be maintained. System maintenance is an important step to ensure that the system runs smoothly and meets the customer’s expectation. In WISE system, there are five major maintenance issues: WISE’s files and directories, WISE installation, system migration, and database maintenance and program modification.

4.1 Files and Directories

I have arranged related file and directories based on the design in Chapter two. There are seven main directories, see Tables 4.1-4.7. All directories are under
/usr/local/tomcat/webapps/wise/web-inf/classes/

Table 4.1. Files in Main Directory

<table>
<thead>
<tr>
<th>Java files under main classes directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Connectsql.java</td>
</tr>
<tr>
<td>2. Connectfile.java</td>
</tr>
<tr>
<td>3. QueryBean.java</td>
</tr>
<tr>
<td>4. InsDelUpdBean.java</td>
</tr>
<tr>
<td>5. Shuffle.java</td>
</tr>
<tr>
<td>6. CreateChoice.java</td>
</tr>
<tr>
<td>7. FileBean.java</td>
</tr>
<tr>
<td>8. RandonNumber.java</td>
</tr>
</tbody>
</table>
4.2 Software Installation

This section shows how to install WISE to in the Linux operating system. I assume that JSP Tomcat server and mySQL are already installed in Linux system. The following steps are needed for system migration.

1. Download "wise.tar.gz" file that is available at gtss.ias.csusb.edu website.

2. Copy the file to /usr/local/tomcat/webapps.

3. extract the file by using this linux command "tar -xzf wise.tar.gz".

4. Edit file /usr/local/tomcat/conf/server.xml by adding the following line in the context manager section "<Context path="/wise"

108
5. Restart JSP tomcat server.

4.3 System Migration

WISE system is designed to run across platforms but if system administrator plans to move from one platform to another there is nothing to do with WISE system except a few check on configuration.

When system administrator plan to move system to different platform, for example from Linux to windows. The following steps are needed for system migration.

1. Make back up of the WISE system which is located at /usr/local/tomcat/webapps/WISE.

2. Make another backup of WISE database which is located at /usr/local/mysql/data/WISE

3. System Administrator must download, install and configure JSP and MySQL windows version.


5. Copy back all backup files in No1. to both JSP and MySQL directories.
4.4 Database Maintenance

The MySQL database locate at /usr/local/MySQL in Linux platform. In order to perform maintenance process, system administrator has to log on to MySQL server to do maintenance job. Administrator can grants user to maintain database by give permission in user table and db table of mysql database. Mysql database is located at /usr/local/mysql/data/mysql. In user table, there are a variety of permissions such as grant user to modify table, or grant user to shut down server. Db table is a table that limit user to access databases.

Example: To add a user, you must first logon to the MySQL database as administrator and use the mysql database. To do this, perform the following steps:

1. Change to mysql installation directory.
   
   cd /usr/local/mysql

2. Ensure that the mysqld is up and running.
   
   bin/mysqladmin -p ping

3. You will be prompted for your root password. After you have entered it correctly, you be able to access mysql database by typing in the following:

   use mysql;
4. To create new user use the following command:
   
   ```sql
   insert into user(host,user,password)
   values
   ('gtss.ias.csusb.edu','scott',password('test'));
   ```

5. To add select privileges to scott, administrator uses the following command:

   ```sql
   .insert into db(host,user,db,select_priv)
   values('gtss.ias.csusb.edu','scott','WISE','Y');
   ```

4.5 Recompilation

Usually, Most of sixty-five percents of maintenance task apply to modifying a program after it has been delivered and is in use. These modifications may involve simple changes to correct code errors, more extensive changes to correct design errors or rewrite code to meet new requirements.

JSP keeps all compiled java programs (filename.class) in "/usr/local/tomcat/webapps/wise/web-inf/classes". Modified java program must recompile by using "javac filename.java" command at Linux prompt. After compiled, Java will create "filename.class", which will be moved to the directory mentioned above.

Alternative way is to create jar file. The jar is the way to group all compiled Java program together as one
file. The advantage is portable and easy to manage. To create jar file, system administrator must compile all Java program to get "dot class" files. The next step is to group those files together by using the following command "jar cvf wise.jar *.class". Then using "mv wise.jar /usr/local/tomcat/lib" to move the "dot jar" to the place that Java program can link to.
CHAPTER FIVE
CONCLUSIONS AND FUTURE DIRECTIONS

5.1 Conclusion

WISE system shows another successful project in GTSS project, which includes complete sorting algorithm tutorials system and interactive self-evaluation exercise system. With Java object-oriented design and MYSQL database design, WISE system delivers fast, reliable and highly portable Web-based exercise management system.

By using this master project, students can evaluate their understanding of Analysis of sorting algorithm through on interactive online system. Also instructors are able to create and manage questions to evaluate students' performance via WISE self-evaluation tools. Tools for creating self-evaluation exercises include EZ question creator for creating question database and automatic exercises, and knowledge base creator for creating new intelligent Java programs for creating automatic answer to prevent cheating.

WISE can be accessed anywhere by any Web browser. Java runtime environment software version 1.1 or later is required to access the system. A user can download this free software from Sun Website or automatically download
on Windows platform. By using session tracking technology, the user activity information can be kept securely on WISE server. This ensures the user privacy.

Moreover, the experiences in implementing this project consist of using various technologies: Java, JDBC, MySQL, and Web server, which provided valuable knowledge for myself.

5.2 Future Directions

WISE system shows a new way in implementing interactive self-evaluation system on the Web. WISE's features were declared earlier in Chapter One. Using these as the starting point, WISE may have additional and improved features:

5.2.1 Adaptive Test

By using artificial intelligence concept, new adaptive test can be developed. Adaptive test is an intelligent tests that respond with the user's answer. When the user answers a question correctly, the next question will be harder otherwise easier question will be asked. Test engine will calculate the score based on the weighted of the correct answers.
5.2.2 Graphic Score Analyzer

With the current score representation, WISE system still shows an individual score. This analyzer tool will include for all the registered users in WISE system. Graphic score analyzer will help students compare their score with others. With graphic color bar chart or pie charts, the students can measure their abilities.
<table>
<thead>
<tr>
<th>Terminology</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insertion Sort</td>
<td>Sort by repeatedly taking the next item and inserting it into the final data structure in its proper order with respect to items already inserted. Run time is $O(n^2)$.</td>
</tr>
<tr>
<td>Heap Sort</td>
<td>A sorting algorithm that works by first organizing the data to be sorted into a special type of binary tree called a heap. The heap itself has, by definition, the largest value at the top of the tree, so the heap sort algorithm must also reverse the order. It does this with the following steps: 1. Remove the topmost item (the largest) and replace it with the rightmost leaf. The topmost item is stored in an array. 2. Re-establish the heap. Repeat steps 1 and 2 until there are no more items left in the heap.</td>
</tr>
<tr>
<td>Merge Sort</td>
<td>A sorting algorithm, which splits the items to be sorted into two groups, recursively sorts each group, and merge them into a final, sorted sequence. Run time is $O(n \log n)$.</td>
</tr>
<tr>
<td>Bubble Sort</td>
<td>Sorting by comparing each adjacent pair of items in a list in turn, swapping the items if necessary, and repeating the pass through the list until no swaps are done.</td>
</tr>
<tr>
<td>Terminology</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Quick Sort</td>
<td>An in-place sort algorithm that uses the divide and conquer paradigm. It picks an element from the array (the pivot), partitions the remaining elements into those greater than and less than this pivot, and recursively sorts the partitions. There are many variants of the basic scheme above: to select the pivot, to partition the array, to stop the recursion on small partitions, etc.</td>
</tr>
<tr>
<td>Java Applet</td>
<td>Java Applet is a kind of mini-application, design to be run by a Java-enabled Web Browser such as Internet Explorer or Netscape Navigator, or in the context of some others &quot;applet viewer&quot;</td>
</tr>
<tr>
<td>Java 2 Development Kit</td>
<td>Java 2 SDK is Java development program from Sun Microsystems JSDK will be used to compile and run java program for this project.</td>
</tr>
<tr>
<td>Java Server Pages</td>
<td>JSP is a Java-based technology that simplifies the process of developing dynamic web sites. JSP is also a type of server-side scripting language. Although there are many server-side scripting languages such as ASP, PHP, Java Script, JSP is the best choice for platform independent.</td>
</tr>
<tr>
<td>JAKARTA-TOMCAT</td>
<td>Jakarta Project is to provide commercial-quality server solutions based on the Java Platform that is developed in an open and cooperative fashion. Simply, Tomcat is JSP server. Tomcat will receive JSP file, execute JSP command and response back to client. Jakarta-Tomcat is free and available on the Internet at <a href="http://jakarta.apache.org">http://jakarta.apache.org</a></td>
</tr>
<tr>
<td>Terminology</td>
<td>Definition</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Java Database Connectivity</td>
<td>JDBC technology is an Application Programming Interface that lets you access virtually any tabular data source from the Java programming language. In other words, It helps java program to communicate, query and update databases. JDBC needs JDBC driver as a connector (pipe) between java program and database program. There are many vendors provide these drivers.</td>
</tr>
</tbody>
</table>
REFERENCES


