COMPREHENSIVE EXAM TEST PREPARATION SYSTEM

A Thesis
Presented to the
Faculty of
San Diego State University

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

by
Sanketkumar Amrutlal Patel
Spring 2011
The Undersigned Faculty Committee Approves the
Thesis of Sanketkumar Amrutlal Patel:

Comprehensive Exam Test Preparation System

Carl Eckberg, Chair
Department of Computer Science

Joseph Lewis
Department of Computer Science

Carmelo Interlando
Department of Mathematics and Statistics

Mar. 16, 2011
Approval Date
Copyright © 2011

by

Sanketakumar Amrutlal Patel

All Rights Reserved
DEDICATION

This thesis is dedicated to my parents who have supported me all the way since the beginning of my studies.
ABSTRACT OF THE THESIS

Comprehensive Exam Test Preparation System
by
Sanketkumar Amrutlal Patel
Master of Science in Computer Science
San Diego State University, 2011

This thesis contains two parts: Database Design and Implementation and Website Design and Development.

The database design separates questions and solutions/choices into two tables, which gives great flexibility to support different type of questions such as Multiple Choice, True/False, Short Answer/Essay Question and Long Answer Question.

The website is designed to provide a convenient UI (User Interface) for two different users: professor and student. It allows professor to search, edit, add, delete questions, as well as provide solutions to some selected questions. It also helps students to form a study group for a specific exam and search questions by topic, question type and exam time. Both students and professors can add their comments to a question for discussion. Additionally, the Check-Your-Solution feature is provided exclusively for Multiple Choice and True/False questions.

With the help of this Test Preparation System, students can practice the old exam questions and get inspired by professors or other group members in order to pass the real test.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Abstract</th>
<th>vi</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>viii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>ix</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>xi</td>
</tr>
</tbody>
</table>

## CHAPTER

1. **Introduction**
   1.1 Motivation
   1.1.1 Project Background
   1.1.2 Related Work
      1.1.2.1 Oracle and Java Certification Trainer
      1.1.2.2 Enhanced Framework for Computer Based Testing
   1.2 Features

2. **Technology Used**
   1.1 Overview
   1.2 Apache
   2.3 PHP
   2.4 MySQL

3. **Database Design and Implementation**
   3.1 Database Design
      3.1.1 Requirements Analysis
      3.1.2 Data Model
      3.1.3 Table Details
         3.1.3.1 Table: Users
         3.1.3.2 Table: Exams
         3.1.3.3 Table: Topics
         3.1.3.4 Table: Question_types
         3.1.3.5 Table: Exam_name
LIST OF TABLES

Table 3.1. Structure of Table ‘Users’ .......................................................................................14
Table 3.2. Structure of Table ‘Exams’ .....................................................................................15
Table 3.3. Structure of Table ‘Topics’ .....................................................................................16
Table 3.4. Structure of Table ‘Question_types’ .......................................................................17
Table 3.5. Structure of Table ‘Exam_time’ .............................................................................18
Table 3.6. Structure of Table ‘Exhibit_types’ .........................................................................19
Table 3.7. Structure of Table ‘Questions’ ................................................................................20
Table 3.8. Structure of Table ‘Ques_link_topics’ ....................................................................22
Table 3.9. Structure of Table ‘Choices’ ...................................................................................23
Table 3.10. Structure of Table ‘Solutions’ ...............................................................................24
Table 3.11. Structure of Table ‘Study_groups’ .......................................................................27
LIST OF FIGURES

Figure 2.1. Architecture of online test preparation system .......................................................5
Figure 3.1. One-to-many relationship between table parent and table child. .........................11
Figure 3.2. Many-to-many relationship between table A and table B ....................................12
Figure 3.3. Database design for online test preparation system ...........................................12
Figure 3.4. Sample data in the ‘users’ table ...........................................................................14
Figure 3.5. Sample data in the ‘exams’ table ........................................................................15
Figure 3.6. Sample data in the ‘topics’ table .......................................................................16
Figure 3.7. Sample data in the ‘Question_types’ table ...........................................................17
Figure 3.8. Sample data in the ‘Exam_time’ table .................................................................18
Figure 3.9. Sample data in the ‘Exhibit_types’ table ..............................................................19
Figure 3.10. Sample data in the ‘Questions’ table .................................................................21
Figure 3.11. Sample data in the ‘Ques_link_topics’ table ......................................................22
Figure 3.12. Sample data in the ‘Choices’ table .....................................................................23
Figure 3.13. Sample data in the ‘Solutions’ table .................................................................25
Figure 3.14. Three many-to-many relationships among ‘users’, ‘exams’ and ‘exam_time’ ..................................................................................................................26
Figure 3.15. Breaking three many-to-many relationships to three one-to-many relations by creating a new table ‘Study_groups’ .......................................................27
Figure 3.16. Sample data in the ‘Comments’ table .................................................................28
Figure 4.1. Design for login form ...........................................................................................42
Figure 4.2. Design for registration form ..................................................................................42
Figure 4.3. Design form for Study_groups form .................................................................46
Figure 4.4. Design for search form .......................................................................................49
Figure 4.5. Design for new question form ..............................................................................51
Figure 4.6. Form to add a solution to the new question .......................................................54
Figure 4.7. Form to add choices to the new question .........................................................54
Figure 4.8. Form to edit non-choice question ........................................................................59
Figure 4.9. Form to edit choice question ..............................................................................60
ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Professor Carl Eckberg for his consistent support and advice. I would also like to thank Dr. Joseph Lewis and Dr. Carmelo Interlando for serving in my thesis committee.
CHAPTER 1

INTRODUCTION

The emergence and continued growth of the Internet is making available new means of communication that affect many aspects of modern human life. Recently, there has been much attention on the Internet as a new medium to aid distance learning and training. For example, the well-known ETS (Educational Test Service) supports online tutorial and practice for TOEFL and GRE in order to help students get familiar with various test formats and pass the actual tests.

Many researches [1-7] have shown that there are three important factors that contribute to this explosion in online learning/training system.

First and foremost, the widespread use of the personal computers and new networking technologies make it possible for distant learning and training.

Second, the internet provides a new learning experience for students by allowing them to access the system from anywhere at any time.

Third, those learning systems generally include interactive methods such as bulletin boards, discussion rooms, which help two-way communication among students and between teachers and students to improve students’ comprehension level and learning motivation.

However, there are disadvantages for those learning/training systems also. The development cost is expensive and it is difficult to establish a system which is suitable for every need [8]. Secondly, if the database is badly designed, and cannot be extended, it may cause the abandonment of the system in the future and require rebuilding everything from scratch. Furthermore, in case of connection failure and system failure, data transferring over the Internet may also cause a problem.

Therefore, how to maximize gain and minimize pain of the Internet has become a big challenge for design and implementation of an online learning/training system.

1.1 MOTIVATION

This section talks about project background and related work.
1.1.1 Project Background

Graduate students of the Computer Science Department at San Diego State University must select PLAN A (thesis) or PLAN B (comprehensive examinations) in order to achieve a Master Degree in Computer Science [9]. PLAN B requires students to pass three comprehensive exams, or pass two comprehensive exams and take one advisor approved extra class, or pass one comprehensive exam and take two advisor approved extra classes. There are five comprehensive exams available for students: Automata and Formal Languages (AFL), Programming Languages (PL), Data Structures and Algorithms (DSA), Operating Systems and Architecture (OSA), and Intelligent Systems and Robotics (ISR). The five comprehensive exams are offered twice a year before the start of the spring semester and fall semester, and students can try at most three times for each exam [10]. Since the number of students that can be accommodated in PLAN A is limited by faculty resources, most graduate students should choose PLAN B. Although students can obtain the copies of previous exams at Cal Copy for self-study, the pass rate is still not high.

Professor Carl Eckberg had helped graduate students to form a study group to prepare an AFL (Automata and Formal Languages) PLAN B test. During this process, they gained deeper knowledge through the discussions with Professor Carl Eckberg and each other. The successful pass for their AFL PLAN B test reminder Carl Eckberg that interaction and communication between professors and students and among students may help more PLAN B graduate students pass their comprehensive exams. Thus, he proposed an online test preparation system for the Computer Science comprehensive examinations to take advantage of the Internet. Besides, he also clearly defined the specific requirements for this system. A web browser is the only tool required for accessing and using this system. The system should support different types of questions such as Multiple Choice, True/False, Short Answer/Essay Question and Long Answer Question. Through the web interface, professors are allowed to search, add, delete and modify questions and solutions, while students can only search the question by topic, question type and exam time. Furthermore, with the aid of this system, students can easily form a study group and discuss questions with professors and other students. Short answer questions are less common on the exams, but valuable for studying.
To be successful, more than a blog is likely needed, so faculty contributions to the website will be very important.

### 1.1.2 Related Work

Two database-centered testing tools have been developed by some of the Professor Eckberg’s graduate students, as aids for teaching and certification exams. Although they are not web-based, the use of those testing tools stimulates the idea of this project.

#### 1.1.2.1 Oracle and Java Certification Trainer

Mr. Michael Li and Mrs. Helena Kramer [11-12] have developed a tool as part of their thesis, which helps students to train themselves for Oracle certification exams and Sun’s Java certification exams. The most interesting part of database design is the “exhibit” attribute with CLOB data type, which allows user to store any object data into the database. Since the certification exams only include multiple-choice questions with normally four to six choices, the database design combines question and seven choices into one table. The UI (User Interface) for users allows them to search questions by topics and question types, but it does not support the UI for professors to add, delete and update questions. Moreover, the UI is written by Java and it requires pre installing this Java program to every client system before use of the system.

#### 1.1.2.2 Enhanced Framework for Computer Based Testing

Sunil Ramaiah [3] has extended Michael Li and Helena Kramer’s work by adding more enhancements for UI. Some of main driving points of his work are shown below:

- Give flexibility to add, modify and delete questions
- Generate real time tests at universities and training centers
- Create test bank easily from an XML file which contains all the questions.
- No dependence on a DBA to create and maintain the test bank
- Generate 4 versions of a given tests by randomly ordering the questions
- View grades of all students for each test bank
- Email grades to all students
- Assign and email username and password to students who are going to take the tests
Since the database design doesn’t change, this enhanced framework also supports multiple-choice questions only for certification exams. Additionally, the Java-UI limits the use of this system to a preinstalled environment.

1.2 Features

By analyzing the need and resolving the previous work, we have developed an innovative online test preparation system to help graduate students of the Computer Science department pass their PLAN B tests. The main characteristics of the system are summarized as follows:

- **Web-based access**: The user only needs a web browser to access and use the system.

- **Login and register**: Every user, either professor or student, is required to login or register an account in order to use the system. A randomly generated password will be emailed to a first-time registrant. Once the user has logged in, he or she will have the right to change the password. For those who forgot their password, the “Mail me my password” will send the password to their email addresses.

- **Supporting different types of questions**: Unlike the previous work, the database design separates questions and solutions/choices into two different tables, which gives great flexibility to support different types of questions such as Multiple Choice, True/False, Short Answer/Essay and Long Answer questions.

- **Allowing professors to search, add, delete and modify questions and solutions**: Professors can search questions by topics, question types and exam time. In addition, the easy-use interface greatly increases the efficiency to add, delete and modify questions and solutions.

- **Allowing students to search and view all the questions**: Students can also search questions by topics, question types and exam time. Moreover, they can view all the old questions for each comprehensive exam entered in the database.

- **Helping students to form a study group**: Each time students log in to the system, they will have the option to choose a study group for an incoming PLAN B test. Once they choose one or more options, the system will send a group list to all group members to help them communicate with each other.

- **Providing discussion for each question**: Both students and professors can add their comments to a question for discussion.

- **Self-assessment for Multiple Choice and True/False questions**: When students practice Multiple Choice and True/False questions, they can do self-assessment by clicking “Check Your Solution” button.
CHAPTER 2

TECHNOLOGY USED

This chapter will discuss the technology used in this thesis including Apache, PHP and MySQL.

1.1 OVERVIEW

The web application server is Apache with version 2.2.14. Database is MySQL with version 5.1.25 and PHP 5.2.11 is used for the server-side scripting language. Figure 2.1 shows where these components reside in the application.

![Figure 2.1. Architecture of online test preparation system.](image)

The Apache web server intercepts HTTP requests and either serves them directly or passes them on to the PHP interpreter for execution. The PHP interpreter parses and executes PHP code which ends with a .php suffix, and returns the resulting page to the web server. The MySQL RDBMS serves as the data storage engine, accepting connection from the PHP and inserting, deleting, updating or retrieving data.

1.2 APACHE

The Apache Web Server commonly known as Apache was created by Robert McCool in 1995 [13]. Over the years Apache diverged from the original version as many features were added to it. As open source software, Apache is developed and maintained by the Apache Server Community under the auspices of the Apache Software Foundation. It can be used under a wide variety of operating systems, including UNIX, GNU, Linux, Microsoft
Windows and Mac. According to Netcraft surveys [14], Apache has become the most popular web server since 1996 and as of September 2009 more than 54% of all websites and 66% of the million busiest sites are running under Apache.

The main reason why Apache is the most used product in the Internet is because of the great advantages it offers for both users and developers:

First, Apache has various useful features, many implemented as compiled modules which extend the core functionality. Moreover, Apache is always innovative and is able to implement the latest protocols.

Second, Apache’s module structure allows users or developers to build a server that is customizable.

Another driving point for Apache is its strong administrator tools. All configuration files are in ASCII format and can be edited using any text editor. With the help of those well-documented configuration files, an administrator can easily monitor or configure the server.

Apache Web Server is efficient. A lot of effort has been put into optimizing Apache’s code for performance. Therefore, it runs faster and consumes fewer resources than many other servers.

As open-source software, Apache can be extended to fit users’ needs. In other words, the users can write their own code and adapt to the Apache server features. When any bugs are found, they can be quickly communicated and rapidly fixed. By sharing and integrating the work from thousands of programmers around the world, Apache has become a robust web server with great popularity.

### 2.3 PHP

PHP is a widely used server-side scripting language that was designed for Web development and can be embedded into HTML [15]. It was created by Rasmus Lerdorf as a means of monitoring page views for his online resume in 1994. Originally it was called “Personal Home Page”. Later in 1997, Zeev Suraski and Andi Gutmans rewrote the parser and formed the base of PHP 3. They also changed its name to PHP Hypertext Preprocessor. Since then, a community of followers and developers formed and began using and further developing PHP. As of April 2007, PHP is installed on over 20 million websites and 1 million web servers [16].
The reason for PHP’s popularity is fairly simple. Unlike most modern server-side languages, PHP uses clean, simple syntax which make it easy to understand and learn, and encourage rapid application development. PHP can be intermixed with HTML to make up a website. When a visitor comes to the website, the PHP code is interpreted and executed by the web server. Because PHP is executed on the server and not on the client, the user does not need special browser plug-ins to see the PHP in action.

Another good thing about PHP is that it’s free – its source code is freely available on the web, and developers can install and use it without paying license fees. Moreover, the free community makes the PHP world a better place. The PHP users or developers can always find support from the community and post their own work for others to enjoy.

PHP also runs on almost every operating system including UNIX, Linux, Microsoft Windows and Mac OS. Additionally, because PHP code is interpreted and not compiled, PHP scripts written on one platform usually work on another platform as long as an interpreter exists.

PHP doesn’t occupy a lot of the system’s resources so it runs faster than most other scripting languages like JSP, ASP.NET and Perl. It is also fairly stable since it is open source; the PHP community works together to fix bugs and integrate new technologies into the core language.

Another key advantage of PHP is its extensibility. Through modular extensions, PHP enables developers to easily add support for new technologies to the language. For example, the developers can find or contribute the modules for PDF files, flash movies, electronic payments and more.

In addition, PHP provides tons of database interfaces and other modules. It supports MySQL, Oracle, Microsoft Access and plenty of others. By simplifying and streamlining database access, PHP enables developers to build complicated data-driven web application while enjoying the short developing cycle.

2.4 MySQL

MySQL is a multi-user relational database management system (RDBMS). It is owned and supported by a Swedish company called MySQL AB, now a subsidiary of Oracle Corporation. This company offers two licenses; one is free under the terms of GNU General
Public License, the other is a proprietary license available for purchase. MySQL runs on a variety of platforms including Linux, UNIX, Microsoft Windows and Mac. It has become the database of choice for LAMP applications, a set of programs, Linux, Apache, MySQL, PHP/Perl/Python often used to run dynamic websites or servers. The popularity of MySQL is closely tied to the popularity of PHP, which is often combined with MySQL. This well-matched couple often works together seamlessly and smoothly. Several prominent websites such as Flickr, Google, Facebook, Yahoo and YouTube use MySQL for data storage and logging of user data [17].

The following sections describe MySQL’s most compelling features.

MySQL offers high performance through the table and index partitioning, ultra-fast load utility, distinctive memory caches and full-text indexes. And MySQL is faster than almost all its competitors, including commercial systems like Microsoft SQL Server and IBM DB2.

MySQL also supports robust transactions. It is completely ACID (atomic, consistent, isolated, durable), and has unlimited row-level locking, distributed transaction capability, and multi-version transaction support.

Its high performance query engine, fast data insert capability and strong support for specialized web functions has made MySQL the standard for high traffic websites.

For data security, MySQL comes with a powerful mechanism to prevent unauthorized users from accessing the system. The safe and secure connection to the system can be established though SSH and SSL. Encryption and decryption functions protect sensitive data from unauthorized viewers.

MySQL is easy to use, administer, and optimize. A simple SQL command-line interface is the primary user interface to the server; users can use a browser-based GUI to administer one or more MySQL database servers like MySQL Administrator, MySQL Migration Toolkit and MySQL Query Browser.

Furthermore, MySQL is free under the General Public License. The developer and users can download and modify the source code to their needs with zero cost, which create an active and enthusiastic global community of MySQL. This community plays an active role in keeping MySQL ahead of its competition, both by testing the software for reliability on worldwide installations and by integrating the latest technologies to the core engine.
With over 6 million installations worldwide and an average of 50,000 daily downloads, MySQL is clearly on the move [17].
CHAPTER 3

DATABASE DESIGN AND IMPLEMENTATION

This chapter will discuss the design and implementation of the database.

3.1 DATABASE DESIGN

The design phase of the database includes requirements analysis, data model and table details.

3.1.1 Requirements Analysis

The Comprehensive Exam Test Preparation System is to be used by the Computer Science Department to help graduate students pass their PLAN B tests. The system will archive previous PLAN B test questions and answers into a database and use web browsers to access and use the system. It should support different types of questions such as Multiple Choice, True/False, Short Answer/Essay Question and Long Answer Question. For Multiple Choice questions, it covers multiple choices with a single correct answer and multiple choices of the form: choose all correct answers.

There are two kinds of users that can use the system, professors and students. Different user types have different privileges. All the users need to register an account with the site to provide their names and email addresses. If a new registrant is a student, he/she is also required to enter the student ID during the registration. All this information will be saved to a database to keep track of user information. Once a user successfully registers an account, a randomly generated password will be mailed to his/her email address. Using the combination of email address and mailed password, the first-time registrant can login to the system.

Whenever a student logs in, he/she will have choice to join study group(s). If he/she chooses one or more study groups, the system will send group email to all the members to help them find study partner(s). Students can search questions by topics, question types and exam time. They can also view all the old questions for each comprehensive exam. A user who clicks on Mail me my password gets an email with the password.
Professors are allowed to search, add, delete and modify questions and solutions/choices. When they add new questions to a database through the web browser, they can copy and paste the text part of the question, or upload a PDF/GIF/JPEG/HTML/DOC file to the question form. By doing this, the questions which contain tables, pictures, or special characters can be easily stored into the database.

Both students and professors can add their comments to a question for discussion.

### 3.1.2 Data Model

Data modeling is the act of exploring data-oriented structures. The goal is depicting the data tables, the data columns of those tables, and the relationship between the tables.

To better understand the design of the data model, let’s first talk about the relationships between two tables and the notations used in this thesis.

There are three types of relationships associated between two tables:

- **One-to-one**: Two tables are related in a one-to-one relationship if, for every row in the first table, there is at most one row in the second table. The one-to-one relationship does not exist in this thesis and it also seldom occurs in the real world.

- **One-to-many**: Two tables are related in one-to-many relationship if for every row in the first table, there can be zero, one, or many rows in the second table, but for every row in the second table there is exactly one row in the first table. The one-to-many relationship is also referred to as a parent-child relationship. An arrowhead is used in this thesis to indicate which the ‘one’ is and which the ‘many’ is, as in Figure 3.1.

![Figure 3.1. One-to-many relationship between table parent and table child.](image)

- Many-to-many: Two tables are related in a many-to-many relationship when for every row in the first table, there can be many rows in the second table and for every row in the second table, there can be many rows in the first table. Many-to-many relationships cannot be directly modeled in a relational database. The solution is to create a new table which will act as the ‘many’ in two ‘one-to-many’ relationships, with each of the original tables acting as the ‘one’. This is shown in Figure 3.2 with the new table identified as ‘X’. The design of the database is shown in Figure 3.3.

As in Figure 3.3, the database design separates questions and solutions/choices into two different tables, which gives great flexibility to support different types of questions.
such as Multiple Choice, True/False, Short Answer/Essay Question and Long Answer Question. The attribute “q_exhibit” of table ‘questions’ and the attribute “s_exhibit” of table ‘solutions’ will support users to upload PDF/GIF/JPEG/HTML/DOC files to the questions and solutions/choices. And the attribute “q_exhibit_type” and “s_exhibit_type” will help to display those files in the right format. The attribute “user_level” of table ‘users’ is designed
to differentiate professors and students, and the attribute “REDID” is used to uniquely identify a student. To support future extension of the database, there are two extra columns, “col1” and “col2” in table ‘questions’, ‘solutions’, ‘users’ and ‘comments’.

Since each question may cover multiple topics and each topic may be contained in multiple questions, a linking table, ‘ques_link_topics’, is created to break the many-to-many relationship between table ‘questions’ and table ‘topics’ into two one-to-many relationships (table ‘questions’ and table ‘topics’ are both related to table ‘ques_link_topics’ in one-to-many relationships). Similarly the table ‘study_groups’ resolves the many-to-many relationships among table ‘users’, ‘exams’ and ‘exam_time’.

### 3.1.3 Table Details


#### 3.1.3.1 TABLE: USERS

Table ‘users’ will maintain user information such as email address, first name, last name and etc. This table will be used for login, registration and communication among study group members. The structure of table ‘users’ is shown in Table 3.1. And the sample data is shown in Figure 3.4.

#### 3.1.3.2 TABLE: EXAMS

All exam names are stored into the table ‘exams’. And table ‘questions’ will have an attribute called ‘exam_name’ to reference the ‘exam_name’ attribute of table ‘exams’. To simplify queries and avoid join-operations between the table ‘exams’ and the table ‘questions’, the string-type primary key is used here as depicted in Table 3.2. The sample data is shown in Figure 3.5.

#### 3.1.3.3 TABLE: TOPICS

Each comprehensive exam has its own set of topics and different exams may have the same topic. To uniquely identify a record in the table ‘topics’, we need to know both the exam name and the topic name. Moreover, there is a many-to-many relationship between
Table 3.1. Structure of Table ‘Users’

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Type</th>
<th>Required Attribute</th>
<th>Constraints</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>email</td>
<td>VARCHAR(100)</td>
<td>Yes</td>
<td>Primary Key</td>
<td>The user’s email address will be used for login, password acquisition and recovery, communication among study group members</td>
</tr>
<tr>
<td>password</td>
<td>VARCHAR(100)</td>
<td>Yes</td>
<td></td>
<td>Login password which is any combination among A-Z, a-z and 0-9</td>
</tr>
<tr>
<td>first_name</td>
<td>VARCHAR(100)</td>
<td>Yes</td>
<td></td>
<td>User’s first name</td>
</tr>
<tr>
<td>last_name</td>
<td>VARCHAR(100)</td>
<td>Yes</td>
<td></td>
<td>User’s last name</td>
</tr>
<tr>
<td>user_level</td>
<td>ENUM(‘student’, ‘professor’)</td>
<td>Yes</td>
<td></td>
<td>Either ‘student’ or ‘professor’</td>
</tr>
<tr>
<td>REDID</td>
<td>VARCHAR(12)</td>
<td>No</td>
<td></td>
<td>Student’s ID number to uniquely identify a student</td>
</tr>
<tr>
<td>ts</td>
<td>TIMESTAMP</td>
<td>No</td>
<td></td>
<td>Store the timestamp whenever the record is created or modified</td>
</tr>
<tr>
<td>col1</td>
<td>VARCHAR(255)</td>
<td>No</td>
<td></td>
<td>For future extension</td>
</tr>
<tr>
<td>col2</td>
<td>VARCHAR(255)</td>
<td>No</td>
<td></td>
<td>For future extension</td>
</tr>
</tbody>
</table>

Figure 3.4. Sample data in the ‘users’ table.
Table 3.2. Structure of Table ‘Exams’

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Type</th>
<th>Required Attribute</th>
<th>Constraints</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>exam_name</td>
<td>VARCHAR(100)</td>
<td>Yes</td>
<td>Primary Key</td>
<td>Store comprehensive exam names like Programming Languages, Automata and Formal Languages, etc.</td>
</tr>
<tr>
<td>ts</td>
<td>TIMESTAMP</td>
<td>No</td>
<td></td>
<td>Store the timestamp whenever the record is created or modified</td>
</tr>
</tbody>
</table>

Figure 3.5. Sample data in the ‘exams’ table.

<table>
<thead>
<tr>
<th>Exam_Name</th>
<th>ts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automata and Formal Language</td>
<td>2009-09-27 14:01:57</td>
</tr>
<tr>
<td>Data Structures and Algorithms</td>
<td>2009-09-25 14:01:57</td>
</tr>
<tr>
<td>Intelligent Systems and Robotics</td>
<td>2009-09-28 14:01:57</td>
</tr>
<tr>
<td>Operating Systems and Architecture</td>
<td>2009-09-26 14:01:57</td>
</tr>
<tr>
<td>Programming Languages</td>
<td>2009-09-24 14:01:57</td>
</tr>
</tbody>
</table>

the table ‘topics’ and the table ‘questions’. Thus, an integer id is chosen as the primary key for the table ‘topics’ to increase the execution speed. The detail structure of the table ‘topics’ is shown in Table 3.3. And the sample data is shown in Figure 3.6. Each topic list have “other” category.

3.1.3.4 TABLE: QUESTION_TYPES

Table ‘question_types’ is used to store all the question types supported by this online test preparation system such as Multiple Choice, True/False, Definition, Theorem and Long Answer Question. And table ‘questions’ will have an attribute called ‘qtype_name’ to reference the ‘qtype_name’ attribute of table ‘question_types’. To simplify queries and avoid join-operations between the table ‘question_types’ and the table ‘questions’, the string-type primary key is used here as shown in Table 3.4. And the sample data is shown in Figure 3.7.
Table 3.3. Structure of Table ‘Topics’

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Type</th>
<th>Required Attribute</th>
<th>Constraints</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>tid</td>
<td>INTEGER</td>
<td>Yes</td>
<td>Primary Key</td>
<td>Auto increment value that serves as primary key for record identification</td>
</tr>
<tr>
<td>exam_name</td>
<td>VARCHAR(100)</td>
<td>Yes</td>
<td>Foreign Key</td>
<td>Store the comprehensive exam name and serve as foreign key to reference the table ‘exams’</td>
</tr>
<tr>
<td>topic_name</td>
<td>VARCHAR(100)</td>
<td>Yes</td>
<td></td>
<td>Store the topic name associated with the exam name</td>
</tr>
<tr>
<td>ts</td>
<td>TIMESTAMP</td>
<td>No</td>
<td></td>
<td>Store the timestamp whenever the record is created or modified</td>
</tr>
</tbody>
</table>

Figure 3.6. Sample data in the ‘topics’ table.
Table 3.4. Structure of Table ‘Question_types’

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Type</th>
<th>Required Attribute</th>
<th>Constraints</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>qtype_name</td>
<td>VARCHAR(50)</td>
<td>Yes</td>
<td>Primary Key</td>
<td>Store the question types supported by the online test preparation system such as Multiple Choice, True/False, Definition, Theorem and Long Answer Question</td>
</tr>
<tr>
<td>ts</td>
<td>TIMESTAMP</td>
<td>No</td>
<td></td>
<td>Store the timestamp whenever the record is created or modified</td>
</tr>
</tbody>
</table>

Figure 3.7. Sample data in the ‘Question_types’ table.

3.1.3.5 TABLE: EXAM_NAME

Table ‘exam_time’ is used to store the information regarding the exam time such as “2008 Fall”, “2009 Spring” and “2009 Fall”. To simplify queries and avoid join-operations between tables, this table uses exam time as the primary key. The details of this table are shown in Table 3.5. The sample data is shown in Figure 3.8.

3.1.3.6 TABLE: EXHIBIT_TYPES

Table ‘exhibit_types’ is used to store all the exhibit file formats supported by this application such as “pdf”, “gif”, “jpg”, “html” and “doc”. To simplify queries and avoid join-operations between tables, this table uses file format as the primary key. The details of this table are depicted in Table 3.6. And the sample data is shown in Figure 3.9.
Table 3.5. Structure of Table ‘Exam_time’

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Type</th>
<th>Required Attribute</th>
<th>Constraints</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>etime</td>
<td>VARCHAR(50)</td>
<td>Yes</td>
<td>Primary Key</td>
<td>Store the exam time such as “2008 Fall”, “2009 Spring” and “2009 Fall”</td>
</tr>
<tr>
<td>ts</td>
<td>TIMESTAMP</td>
<td>No</td>
<td></td>
<td>Store the timestamp whenever the record is created or modified</td>
</tr>
</tbody>
</table>

Figure 3.8. Sample data in the ‘Exam_time’ table.
Table 3.6. Structure of Table ‘Exhibit_types’

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Type</th>
<th>Required Attribute</th>
<th>Constraints</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>exhibit_type</td>
<td>VARCHAR(50)</td>
<td>Yes</td>
<td>Primary Key</td>
<td>Store the file format such as “pdf”, “gif”, “jpg”, “html” and “doc”</td>
</tr>
<tr>
<td>ts</td>
<td>TIMESTAMP</td>
<td>No</td>
<td></td>
<td>Store the timestamp whenever the record is created or modified</td>
</tr>
</tbody>
</table>

Figure 3.9. Sample data in the ‘Exhibit_types’ table.

3.1.3.7. TABLE: QUESTIONS

Table ‘questions’ will maintain the questions of the previous PLAN B tests. If a question has multiple choices, this table would only store the question part, and the corresponding choices would be stored into the table ‘solutions’. An auto increment integer serves as the primary key to uniquely identify each question. The attribute “question” with TEXT data type is used for storing long text in a question. And the attribute “q_exhibit” with LONGBLOB data type allows for storing binary files into the table. With the help of “q_exhibit_type”, the binary data stored in “q_exhibit” can be displayed in the right format when needed. If users want to add special symbols in the question field than he/she can use the words table for that. Table 3.7 shows the details of table ‘questions’. The sample data is shown in Figure 3.10.
Table 3.7. Structure of Table ‘Questions’

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Type</th>
<th>Required Attribute</th>
<th>Constraints</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>qid</td>
<td>INTEGER</td>
<td>Yes</td>
<td>Primary Key</td>
<td>Auto increment value that serves as primary key for question identification</td>
</tr>
<tr>
<td>exam_name</td>
<td>VARCHAR(100)</td>
<td>Yes</td>
<td>Foreign Key</td>
<td>Store the exam name and serve as foreign key to reference the table ‘exams’</td>
</tr>
<tr>
<td>qtype_name</td>
<td>VARCHAR(50)</td>
<td>Yes</td>
<td>Foreign Key</td>
<td>Store the question type and serve as foreign key to reference the table ‘question_types’</td>
</tr>
<tr>
<td>etime</td>
<td>VARCHAR(50)</td>
<td>No</td>
<td>Foreign Key</td>
<td>Store the exam time and serve as foreign key to reference the table ‘exam_time’</td>
</tr>
<tr>
<td>question</td>
<td>TEXT</td>
<td>No</td>
<td></td>
<td>Store question’s text part</td>
</tr>
<tr>
<td>q_has_exhibit</td>
<td>TINYINT</td>
<td>Yes</td>
<td></td>
<td>Serve as a Boolean: “Does this question have exhibit?” 0 means no; 1 means yes</td>
</tr>
<tr>
<td>q_exhibit_type</td>
<td>VARCHAR(50)</td>
<td>No</td>
<td>Foreign Key</td>
<td>Store the exhibit type such as “pdf”, “jpg”, “gif”, “html” or “doc”</td>
</tr>
<tr>
<td>q_exhibit</td>
<td>LONGBLOB</td>
<td>No</td>
<td></td>
<td>Store the binary exhibit file</td>
</tr>
<tr>
<td>has_solution</td>
<td>TINYINT</td>
<td>Yes</td>
<td></td>
<td>Serve as a Boolean: “Solution provided?” 0 means no; 1 means yes</td>
</tr>
<tr>
<td>s_has_choices</td>
<td>TINYINT</td>
<td>Yes</td>
<td></td>
<td>Serve as a Boolean: “Is this a Multiple Choice question?” 0 means no; 1 means yes</td>
</tr>
</tbody>
</table>

(table continues)
Table 3.7. (continued)

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Type</th>
<th>Required Attribute</th>
<th>Constraints</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_has_mulAns</td>
<td>TINYINT</td>
<td>Yes</td>
<td></td>
<td>Serve as a Boolean:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>“Is this a Multiple Choice question with multiple answers?”</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 means no; 1 means yes</td>
</tr>
<tr>
<td>s_is_verified</td>
<td>TINYINT</td>
<td>Yes</td>
<td></td>
<td>Serve as a Boolean: “Is solution verified?”</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 means no; 1 means yes</td>
</tr>
<tr>
<td>has_comments</td>
<td>TINYINT</td>
<td>Yes</td>
<td></td>
<td>Serve as a Boolean:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>“Does this question have comments?”</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 means no; 1 means yes</td>
</tr>
<tr>
<td>reference</td>
<td>VARCHAR(255)</td>
<td>No</td>
<td></td>
<td>Store the reference information for this question</td>
</tr>
<tr>
<td>ts</td>
<td>TIMESTAMP</td>
<td>No</td>
<td></td>
<td>Store the timestamp whenever the record is created or modified</td>
</tr>
<tr>
<td>col1</td>
<td>VARCHAR(255)</td>
<td>No</td>
<td></td>
<td>For future extension</td>
</tr>
<tr>
<td>col2</td>
<td>VARCHAR(255)</td>
<td>No</td>
<td></td>
<td>For future extension</td>
</tr>
</tbody>
</table>

Figure 3.10. Sample data in the ‘Questions’ table.
3.1.3.8 TABLE: QUES_LINK_TOPICS

Table ‘ques_link_topics’ is used to break many-to-many relationship between table ‘questions’ and table ‘topics’ into two one-to-many relationships (table ‘questions’ and table ‘topics’ are both related to table ‘ques_link_topics’ in one-to-many relationships). The primary key of table ‘ques_link_topics’ combines the primary keys of table ‘questions’ and table ‘topics’, which denotes the many-to-many relationship between the table ‘questions’ and the table ‘topics’. The details of table ‘ques_link_topics’ are depicted in Table 3.8. The sample data is shown in Figure 3.11.

Table 3.8. Structure of Table ‘Ques_link_topics’

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Type</th>
<th>Required Attribute</th>
<th>Constraints</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>qid</td>
<td>INTEGER</td>
<td>Yes</td>
<td>Part of Primary Key, Foreign Key</td>
<td>Question id to uniquely identify each question, which is part of the primary key of table ‘ques_link_topics’ and serves as foreign key referencing the table ‘questions’</td>
</tr>
<tr>
<td>tid</td>
<td>INTEGER</td>
<td>Yes</td>
<td>Part of Primary Key, Foreign Key</td>
<td>Topic id, as part of the primary key of table ‘ques_link_topics’, serves as foreign key to reference exam name and topic name in table ‘topics’</td>
</tr>
<tr>
<td>ts</td>
<td>TIMESTAMP</td>
<td>No</td>
<td></td>
<td>Store the timestamp whenever the record is created or modified</td>
</tr>
</tbody>
</table>

Figure 3.11. Sample data in the ‘Ques_link_topics’ table.
The many-to-many relationship between the table ‘questions’ and the table ‘topics’ can be clearly demonstrated from Figure 3.11 (p. 22). For qid=1, we can find two different tid: 7, 12. For qid=2, we can find one different tid: 11. Thus, table ‘ques_link_topics’ works like a linkage table connecting table ‘questions’ and table ‘topics’.

3.1.3.9 TABLE: CHOICES

Table ‘choices’ is used to store all the choice options supported by this online test preparation system such as ‘A’, ‘B’, ‘C’, ‘D’. To simplify queries and avoid join-operations between tables, this table uses a choice letter as the primary key. The details of this table are depicted in Table 3.9. The sample data is shown in Figure 3.12.

Table 3.9. Structure of Table ‘Choices’

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Type</th>
<th>Required Attribute</th>
<th>Constraints</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>choice</td>
<td>VARCHAR(12)</td>
<td>Yes</td>
<td>Primary Key</td>
<td>Store the choice letter such as A’, ‘B’, ‘C’, ‘D’</td>
</tr>
<tr>
<td>ts</td>
<td>TIMESTAMP</td>
<td>No</td>
<td></td>
<td>Store the timestamp whenever the record is created or modified</td>
</tr>
</tbody>
</table>

Figure 3.12. Sample data in the ‘Choices’ table.
3.1.3.10 TABLE: SOLUTIONS

Provided solutions or multiple choice options will be stored in the table ‘solutions’. To quickly identify each record in the table, an auto increment integer, sid, is selected as primary key. The foreign key, qid, is used to reference the table ‘questions’ to find the corresponding question part. The attribute “solution” with TEXT data type is used for storing long text. Also there is a words table to add special symbols in this field. And the attribute “s_exhibit” with LONGBLOB data type allows for storing binary files into the table. With the help of “s_exhibit_type”, the binary data stored in “s_exhibit” can be displayed in the right format when needed. Table 3.10 shows the details of table ‘solutions’. And the sample data is shown in Figure 3.13.

Table 3.10. Structure of Table ‘Solutions’

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Type</th>
<th>Required Attribute</th>
<th>Constraints</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>INTEGER</td>
<td>Yes</td>
<td>Primary Key</td>
<td>Auto increment value that serves as primary key for solution/choice identification</td>
</tr>
<tr>
<td>qid</td>
<td>INTEGER</td>
<td>Yes</td>
<td>Foreign Key</td>
<td>Question id serves as foreign key to reference the corresponding question in table ‘questions’</td>
</tr>
<tr>
<td>choice</td>
<td>VARCHAR(12)</td>
<td>No</td>
<td>Foreign Key</td>
<td>Store the choice letter such as A’, ‘B’, ‘C’, ‘D’ and serve as foreign key to reference table ‘choices’</td>
</tr>
<tr>
<td>solution</td>
<td>TEXT</td>
<td>No</td>
<td></td>
<td>Store the text part of a solution/choice</td>
</tr>
<tr>
<td>s_has_exhibit</td>
<td>TINYINT</td>
<td>Yes</td>
<td></td>
<td>Serve as a Boolean: “Does this solution have exhibit?” 0 – No, 1 – Yes</td>
</tr>
<tr>
<td>s_exhibit_type</td>
<td>VARCHAR(50)</td>
<td>No</td>
<td>Foreign Key</td>
<td>Store the exhibit type such as “pdf”, “jpg”, “gif”, “html” or “doc”</td>
</tr>
</tbody>
</table>

(table continues)
<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Type</th>
<th>Required Attribute</th>
<th>Constraints</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_exhibit</td>
<td>LONGBLOB</td>
<td>No</td>
<td></td>
<td>Store the binary exhibit file</td>
</tr>
<tr>
<td>isAnswer</td>
<td>TINYINT</td>
<td>Yes</td>
<td></td>
<td>Serve as a Boolean: “Is this the answer?” 0 means no; 1 means yes</td>
</tr>
<tr>
<td>isVerified</td>
<td>TINYINT</td>
<td>Yes</td>
<td></td>
<td>Serve as a Boolean: “Has this solution/choice been verified?” 0 means no; 1 means yes</td>
</tr>
<tr>
<td>info</td>
<td>VARCHAR(255)</td>
<td>No</td>
<td></td>
<td>Store the information for this solution/choice</td>
</tr>
<tr>
<td>ts</td>
<td>TIMESTAMP</td>
<td>No</td>
<td></td>
<td>Store the timestamp when the record is created or modified</td>
</tr>
<tr>
<td>col1</td>
<td>VARCHAR(255)</td>
<td>No</td>
<td></td>
<td>For future extension</td>
</tr>
<tr>
<td>col2</td>
<td>VARCHAR(255)</td>
<td>No</td>
<td></td>
<td>For future extension</td>
</tr>
</tbody>
</table>

Figure 3.13. Sample data in the ‘Solutions’ table.
3.1.3.11 Table: Study_groups

The data stored in the table ‘study_groups’ is used to answer the question: who plans to attend which comprehensive exams at which time. For each student, he/she can attend multiple comprehensive exams at different times. Many students can attend the same exam in spring or fall. And five comprehensive exams are offered in each spring or fall. Totally, there are three many-to-many relationships as shown in Figure 3.14. Since MySQL cannot directly model many-to-many relationship between tables, a new table, ‘study_groups’, is created to act as ‘many’ in three ‘one-to-many’ relationships, with ‘users’, ‘exams’ and ‘exam_time’ acting as the ‘one’ (see Figure 3.15). The details of table ‘study_groups’ are depicted in Table 3.11. And the sample data is shown in Figure 3.16.

![Diagram of many-to-many relationships among 'users', 'exams' and 'exam_time'.](image)

3.1.3.12 Table: Comments

Table ‘comments’ is used to store comments or opinions to any question. An auto increment integer serves as the primary key to uniquely identify each record in the table ‘comments’. The foreign key ‘qid’ will reference table ‘questions’ to find the question part and another foreign key ‘uemail’ will reference table ‘users’ to find who gave this comment.
Figure 3.15. Breaking three many-to-many relationships to three one-to-many relations by creating a new table ‘Study_groups’.

Table 3.11. Structure of Table ‘Study_groups’

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Type</th>
<th>Required Attribute</th>
<th>Constraints</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>uemail</td>
<td>VARCHAR(100)</td>
<td>Yes</td>
<td>Part of Primary Key, Foreign Key</td>
<td>User email, as part of the primary key of table ‘study_groups’, serves as foreign key to reference table ‘users’</td>
</tr>
<tr>
<td>exam_name</td>
<td>VARCHAR(100)</td>
<td>Yes</td>
<td>Part of Primary Key, Foreign Key</td>
<td>Exam name, as part of the primary key of table ‘study_groups’, serves as foreign key to reference table ‘exams’</td>
</tr>
<tr>
<td>etime</td>
<td>VARCHAR(100)</td>
<td>Yes</td>
<td>Part of Primary Key, Foreign Key</td>
<td>Exam time, as part of the primary key of table ‘study_groups’, serves as foreign key to reference table ‘exam_time’</td>
</tr>
<tr>
<td>ts</td>
<td>TIMESTAMP</td>
<td>No</td>
<td></td>
<td>Store the timestamp whenever the record is created or modified</td>
</tr>
</tbody>
</table>
3.2 DATABASE IMPLEMENTATION

This section shows you how to use MySQL syntax to implement the data model that is described in section 3.1.2, “Data Model”. In other words, it shows you how to create tables and populate tables with data.

3.2.1 Creating the Tables

The CREATE TABLE statement is used to create each table designed in the data model. This statement has a complicated form, but it is basically a list of attributes (columns) of the table. Moreover, InnoDB is chosen as the database engine to support transactions when inserting or updating a question or solution. The following example shows the statements for the data model [18]:

```sql
CREATE TABLE users
(
    email VARCHAR(100) NOT NULL,
    password VARCHAR(100) NOT NULL,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    user_level ENUM('student', 'professor') DEFAULT 'student',
    REDID VARCHAR(12),
    ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    col1 VARCHAR(255),
    col2 VARCHAR(255),
    PRIMARY KEY (email)
) TYPE=InnoDB;

CREATE TABLE exams
(
    exam_name VARCHAR(100) NOT NULL,
```
CREATE TABLE topics
(
    tid INTEGER NOT NULL AUTO_INCREMENT,
    exam_name VARCHAR(100) NOT NULL,
    topic_name VARCHAR(100) NOT NULL,
    ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    PRIMARY KEY (tid),
    CONSTRAINT topics_FK_1
        FOREIGN KEY (exam_name)
        REFERENCES exams (exam_name)
        ON UPDATE CASCADE
) TYPE=InnoDB;

CREATE TABLE question_types
(
    qtype_name VARCHAR(50) NOT NULL,
    ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    PRIMARY KEY (qtype_name)
) TYPE=InnoDB;

CREATE TABLE exam_time
(
    etime VARCHAR(50) NOT NULL,
    ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    PRIMARY KEY (etime)
) TYPE=InnoDB;

CREATE TABLE exhibit_types
(
    exhibit_type VARCHAR(50) NOT NULL,
    ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    PRIMARY KEY (exhibit_type)
) TYPE=InnoDB;
CREATE TABLE questions
(
    qid INTEGER  NOT NULL AUTO_INCREMENT,
    exam_name VARCHAR(100) NOT NULL,
    qtype_name VARCHAR(50) NOT NULL,
    etime VARCHAR(50),
    question TEXT,
    q_has_exhibit TINYINT default 0 NOT NULL,
    q_exhibit_type VARCHAR(50),
    q_exhibit LONGBLOB,
    has_solution TINYINT default 0 NOT NULL,
    s_has_choices TINYINT default 0 NOT NULL,
    s_has_mulAns TINYINT default 0 NOT NULL,
    s_is_verified TINYINT default 0 NOT NULL,
    has_comments TINYINT default 0 NOT NULL,
    reference VARCHAR(255),
    ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    col1 VARCHAR(255),
    col2 VARCHAR(255),
    PRIMARY KEY (qid),
    CONSTRAINT questions_FK_1
        FOREIGN KEY (exam_name)
        REFERENCES exams (exam_name)
        ON UPDATE CASCADE,
    CONSTRAINT questions_FK_2
        FOREIGN KEY (qtype_name)
        REFERENCES question_types (qtype_name)
        ON UPDATE CASCADE,
    CONSTRAINT questions_FK_3
        FOREIGN KEY (etime)
        REFERENCES exam_time (etime)
        ON UPDATE CASCADE,
    CONSTRAINT questions_FK_4
        FOREIGN KEY (q_exhibit_type)
        REFERENCES exhibit_types (exhibit_type)
        ON UPDATE CASCADE
) TYPE=InnoDB;
CREATE TABLE ques_link_topics
(
    qid INTEGER NOT NULL,
    tid INTEGER NOT NULL,
    ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    PRIMARY KEY (qid, tid),
    CONSTRAINT qLinkT_FK_1
        FOREIGN KEY (qid)
        REFERENCES questions (qid),
    CONSTRAINT qLinkT_FK_2
        FOREIGN KEY (tid)
        REFERENCES topics (tid)
) TYPE=InnoDB;

CREATE TABLE choices
(
    choice VARCHAR(12) NOT NULL,
    ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    PRIMARY KEY (choice)
) TYPE=InnoDB;

CREATE TABLE solutions
(
    sid INTEGER NOT NULL AUTO_INCREMENT,
    qid INTEGER NOT NULL,
    choice VARCHAR(12),
    solution TEXT,
    s_has_exhibit TINYINT default 0 NOT NULL,
    s_exhibit_type VARCHAR(50),
    s_exhibit LONGBLOB,
    isAnswer TINYINT default 0 NOT NULL,
    isVerified TINYINT default 0 NOT NULL,
    info VARCHAR(255),
    ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    col1 VARCHAR(255),
    col2 VARCHAR(255),
    PRIMARY KEY (sid),
    CONSTRAINT solutions_FK_1

FOREIGN KEY (qid)
REFERENCES questions (qid),
CONSTRAINT solutions_FK_2
FOREIGN KEY (choice)
REFERENCES choices (choice)
ON UPDATE CASCADE,
CONSTRAINT solutions_FK_3
FOREIGN KEY (s_exhibit_type)
REFERENCES exhibit_types (exhibit_type)
ON UPDATE CASCADE
)TYPE=InnoDB;

CREATE TABLE comments
(
    cid INTEGER NOT NULL AUTO_INCREMENT,
    qid INTEGER NOT NULL,
    uemail VARCHAR(100) NOT NULL,
    comment TEXT NOT NULL,
    ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    col1 VARCHAR(255),
    col2 VARCHAR(255),
    PRIMARY KEY (cid),
    CONSTRAINT comments_FK_1
    FOREIGN KEY (qid)
    REFERENCES questions (qid),
    CONSTRAINT comments_FK_2
    FOREIGN KEY (uemail)
    REFERENCES users (email)
)TYPE=InnoDB;

CREATE TABLE study_groups
(
    uemail VARCHAR(100) NOT NULL,
    exam_name VARCHAR(100) NOT NULL,
    etime VARCHAR(50) NOT NULL,
    ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    PRIMARY KEY (uemail, exam_name, etime),
    CONSTRAINT study_groups_FK_1
    FOREIGN KEY (uemail)
    REFERENCES users (email)
)TYPE=InnoDB;
3.2.2 Populating the Tables

Here are two ways to populate the tables. One is to insert data into a table directly from the backend by using INSERT INTO statements. The following example illustrates this direct insertion:

```
INSERT INTO exams (exam_name) VALUES('Programming Languages');
INSERT INTO question_types (qtype_name) VALUES('Multiple Choice');
INSERT INTO exam_time (etime) VALUES('2009 Fall');
INSERT INTO exhibit_types (exhibit_type) VALUES('pdf');
INSERT INTO choices (choice) VALUES('A');
INSERT INTO topics (exam_name, topic_name) VALUES('Programming Languages', 'Classification and Comparison of Languages');
```

The other way to populate the tables is through the web interface. Users can simply input data into a form and PHP code will insert the form data into the database. The following example adds a new question to the ‘questions’ table. Here is the form located in “new.php”:

```php
<?php
  echo "<form enctype="multipart/form-data" method="POST" action="process_new.php?exam_name=$exam_name">";
  
  echo "<ul><li><b>Topic(s): </b>(To select more than one entry, hold down the <b>Ctrl</b> key)";
  
  if (isset($_SESSION['new_topics'])) {
    $sel_arr = $_SESSION['new_topics'];
  }
  
  foreach ($arr as $topic) {
    echo "<li><input type="checkbox" name="new_topics[$topic]" value="$topic"> $topic</li>";
  }
  
  echo "</ul>";
  
  echo "<textarea name="question"></textarea>";
  
  echo "<input type="file" name="file" accept="application/pdf">";
  
  echo "<input type="submit" name="submit" value="Submit">";
  
  echo "</form>";
```
else {
    $sel_arr = NULL;
}
sel_menu($query, $conn, 'topic_name', $sel_arr);
echo "</li></ul>";

echo "<ul><li><b>Question Type: </b>";
echo "<select name="qtype">";
$query = "SELECT qtype_name FROM question_types order by ts";
if (isset($_SESSION['new_qtype'])) {
    $sel = $_SESSION['new_qtype'];
} else {
    $sel = NULL;
}
.sel_menu($query, $conn, 'qtype_name', $sel);
echo "</li></ul>";

echo "<ul><li><b>Exam Time: </b>";
echo "<select name="etime">";
$query = "SELECT etime FROM exam_time";
if (isset($_SESSION['new_etime'])) {
    $sel = $_SESSION['new_etime'];
} else {
    $sel = NULL;
}
.sel_menu($query, $conn, 'etime', $sel);
echo "</li></ul>";

echo "<ul><li><b>Question: </b></li>";
echo "<textarea rows="5" cols="60" wrap="physical" name="question">";
if (isset($_SESSION['new_q'])) {
    echo $_SESSION['new_q'];
} echo "</textarea>";
if (isset($_SESSION['new_qHasEx'])) {
    $v = $_SESSION['new_qHasEx'];
} else {
    $v = 0;
}
show_T_or_F("Question has Exhibit?", "q_has_exhibit", $v);

$sel = NULL;
if (isset($_SESSION['new_qExT'])) {
    $sel = $_SESSION['new_qExT'];
} else {
    $sel = NULL;
}

if (isset($_SESSION['new_hasS'])) {
    $v = $_SESSION['new_hasS'];
} else {
    $v = 0;
}
show_T_or_F("Solution Provided?", "has_solution", $v);
if (isset($_SESSION['new_sHasCho'])) {
Here, the \$_SESSION variables are used to remember the previous form data in case an error occurs, the user can simply try again or just modify the wrong field. And the $conn is a MySQL link identifier for the connection to a MySQL server. The two helping functions, sel_menu and show_T_or_F, are shown below.

```php
function sel_menu($query, $conn, $col_name, $select_name ) {
    $result = mysql_query($query, $conn);
    if (!$result) {

    }
}
```
When a user clicks the submit button in the HTML form, the form data is sent to "process_new.php", which will retrieve the value from the form with the PHP $_REQUEST variables. Then construct INSERT INTO statement based on the form value and execute this
statement by using the mysql_query() function. Thus, a new question will be added to the ‘questions’ table. Here is the “process_new.php”:

```php
<?php
// Construct question query
$q_q =  "INSERT INTO questions SET exam_name='$exam_name' "
    ." , qtype_name="$_REQUEST['qtype'].'" "
    ." , etime="$_REQUEST['etime'].'" "
    ." , question="$question' "
    ." , q_has_exhibit="$_REQUEST['q_has_exhibit'].'" "
    ." , q_exhibit_type="$_REQUEST['q_exhibit_type'].'" "
    ." , has_solution="$_REQUEST['has_solution']"
    ." , s_has_choices="$_REQUEST['s_has_choices']"
    ." , s_has_mulAns="$_REQUEST['s_has_mulAns'].'" "
    ." , q_has_exhibit=1 "

if ($_REQUEST['q_has_exhibit'] == 1) {
    $fp = fopen($_FILES['q_exhibit']['tmp_name'], 'r');
    if (!$fp) {
        $process_err[] = "Failed to open ".$_FILES['q_exhibit']['tmp_name'];
        $process_ecnt++;
    }
    $file_size = filesize($_FILES['q_exhibit']['tmp_name']);
    if ($file_size > 2000000) {
        $process_err[] = "The size of the uploaded file is larger than 2M. Please check your file."
        $process_ecnt++;
    }
    $data_q = fread($fp, $file_size);
    $data_q = base64_encode($data_q);
    fclose($fp);

    $q_q .= " , q_exhibit='$data_q' ";
}

$q_q .= " , has_solution="$_REQUEST['has_solution']"
    ." , s_has_choices="$_REQUEST['s_has_choices']"
    ." , s_has_mulAns="$_REQUEST['s_has_mulAns'].'" "
```

if (isset($_REQUEST['reference']) && trim($_REQUEST['reference']) != "") {
    $reference = addslashes(htmlspecialchars($_REQUEST['reference']));
    $q_q .= ", reference='$reference' ";
}

// The question query is successfully constructed.
if ($process_ecnt == 0) {
    // Start transaction before adding a new question to the table
    setAuto0($conn);
    begin($conn);

    // Flag to show if the transaction is successful:
    // 1 means success and 0 means failure.
    $flag_insert = 1;

    // Execute the question query.
    $res_q = mysql_query($q_q, $conn);
    if (!$res_q || mysql_affected_rows($conn) != 1) {
        $flag_insert = 0;
    }
    else {
        $res_lastId = mysql_query("select LAST_INSERT_ID()", $conn);
        if (!$res_lastId) {
            $flag_insert = 0;
        }
        else {
            $row = mysql_fetch_row($res_lastId);
            $newqid = $row[0];
            // Update ques_link_topics
            for ($i = 0; $i < count($_REQUEST['new_topics']); $i++) {
                $q_tid = "select tid from topics where exam_name='$exam_name' and topic_name='$_REQUEST['new_topics'][{$i}].'" ";
                $r_tid = mysql_query($q_tid, $conn) or die(mysql_error());
                $row = mysql_fetch_row($r_tid);
                $tid = $row[0];

                $res_qLinkT = mysql_query("INSERT INTO ques_link_topics SET qid=$newqid, tid=$tid", $conn);
                if (!$res_qLinkT || mysql_affected_rows($conn) != 1) {

```php
function setAuto0($conn) {
    // make sure flag the DB to not commit after each executed query
    $r_setAuto = mysql_query("SET AUTOCOMMIT=0", $conn) or die(mysql_error());
}
function setAuto1($conn) {
    $r_setAuto = mysql_query("SET AUTOCOMMIT=1", $conn) or die(mysql_error());
}
function begin($conn) {
    $r = mysql_query("BEGIN", $conn) or die(mysql_error());
}
function commit($conn) {
    $r = mysql_query("COMMIT", $conn) or die(mysql_error());
}
```

You may notice that we use transactions when adding a new question to the table ‘questions’ to keep database integrity and consistency. If no error occurs during the execution of the transaction then the system commits the transaction. All data manipulations within the scope of the transaction are performed. If an error occurs during the transaction, the database will be restored to the state it was in before the transaction started. The PHP functions for a transaction are shown below:
function rollback($conn) {
    $r = mysql_query("ROLLBACK", $conn) or die(mysql_error());
}
?>
CHAPTER 4

WEBSITE DESIGN AND DEVELOPMENT

This chapter will discuss the website design and development including login and register, study groups, searching questions, adding new questions, editing existing question, adding comments, and self-assessment for multiple-choice questions.

4.1 LOGIN AND REGISTER

The design for login and registration are shown in Figure 4.1 and Figure 4.2.

Figure 4.1. Design for login form.

Figure 4.2. Design for registration form.
These two forms use \$_SERVER["PHP_SELF"] as a form action to make the form submission to the same page that displayed it. Thus, we can put the logic to handle the form in the same page as the logic that displays it. The logical flow looks like this:

- If the request isn’t a form submission, show the form.
- If the request is a form submission, validate the submitted parameters.
- If the submitted parameters are valid, process them.
- If the submitted parameters are invalid, show the form.

With the function-based structure, the pseudo code to accomplish this looks something like:

```php
// The form is submitted.
if (isset($_POST['submit'])) {

    // Validating the form
    $retval = validate_form();

    // If the submitted data is valid, so process it
    if ($retval) {
        show_success();
    }
    // If the form has errors, pass errors to show_form()
    else {
        Show_form($form_values, $form_errors);
    }
}
// The form was not submitted yet, display the form.
else {
    Show_form();
}
```

In the form validation, we need to check whether users entered all the required fields and whether the input data is in correct format. Since the email address has a special format like `username@domain.extension`, we need use PHP `eregi` function to check whether the given string matches a particular regular expression. The ‘i’ in the function name means ‘case insensitive’. The regular expression for email format validation is show below:

```
^[a-z0-9_.+-]+@[a-z0-9-]+\.[a-z]{2,3}$
```
Let’s split it into sections and explain each part individually:

^[a-z0-9-]+(\.[a-z0-9-]+)+*$

This part of the expression validates the ‘username’ section of the email address. The hat sign (^) at the beginning of the expression represents the start of the string. If we didn’t include this, then some user could type anything before the valid email address and it would still validate. Contained in the square brackets are the characters we want to allow in this part of the address. The plus (+) sign after the square brackets indicates ‘one or more of the contents of the previous brackets’. And ()* means zero or more occurrences of the characters contained in the curly braces. The backslash followed by a period sign tells the expression that a period is required at this point in the expression. Thus, we allow usernames such as sanket and sanket.xm.

@^[a-z0-9-]++$

This part of the expression validates the domain name in the email format. The ‘@’ sign means we require the presence of the @ sign immediately following the username. As before, we have a series of characters in square brackets followed by a plus (+) sign, which requires one or more of those characters contained in the previous square brackets.

(\.[a-z0-9-]+)*(\.[a-z]{2,3})$

The final part of the expression validates the extension name in the email format. (\.[a-z0-9-]+)* means zero or more character groups which always begins with a ‘.’ followed by one or more of the preceding characters contained in the square bracket. And (\.[a-z]{2,3})$ means that at the end of the email format there must be a period sign followed by two or three letters chose N from a to z. Therefore, edu, com, net, org are all valid. The $ sign indicates the end of the string. The following PHP code gives an example of how to use eregi to validate email format.

```php
$regular_exp = "^[a-z0-9-]+(\.[a-z0-9-]+)+*$

."@[a-z0-9-]+(\.[a-z0-9-]+)+*$

."(\.[a-z]{2,3})$";

if (!eregi($regular_exp, $email)) {
    $_SESSION['login_form_errors']['login_email'] = "Invalid email format";
    $_SESSION['num_login_errors'] += 1;
}
```
After a user registers an account, a randomly generated password will be emailed to this first-time registrant. The following function `generate_pass` is used to generate this password which is a random combination of consonants and vowels.

```php
function generate_pass($length) {
    srand((double)microtime()*1000000);
    $vowels = array("a", ",e", ",i", ",o", ",u");

    $num_vowels = count($vowels);
    $num_cons = count($cons);

    for($i = 0; $i < $length; $i++){
        $password .= $cons[rand(0, $num_cons - 1)]
                    . $vowels[rand(0, $num_vowels - 1)];
    }
    return substr($password, 0, $length);
}
```

### 4.2 Study Groups for Students

The study-groups feature is for students only. Each time a student logs in, the study-groups form will be prompted. Students can choose zero or more options by checking the check box. The design for the study-groups form is shown in Figure 4.3:

The first and second available times will be substituted with the real exam time according to the web-viewing time. Since the five comprehensive exams are offered in each January and August. We can divide one year into two parts: if a student visits this site between January and July, the first available time would be the current year fall and the second available time would be next year spring; otherwise, the first available time should be next year spring and the second available time should be next year fall. For example, suppose a student viewed the web in October 2009. Then, the first available time would be “2010 Spring” and the second available time would be “2010 Fall”.

Once the student has selected a new study group, a group list will be emailed to the student’s login email addresses to help him/her find study partners.
Here is the pseudo code to accomplish this:

```php
// If the study-groups form was submitted.
if (isset($_REQUEST['submit_study_groups'])) {

    // If the student has chosen at least one study group.
    if (isset($_REQUEST['study_groups']) && count($_REQUEST['study_groups']) > 0) {

        // Find each study group selected by the student.
        foreach ($_REQUEST['study_groups'] as $mygroup) {

            // If a student choose one study group by checking the check box,
            // the value of this check box will have the following format:
            // <exam name>-<exam time>
            // There is a dash sign to separate exam name and exam time.
            // For example: Programming Languages-2010 Spring
```
Thus, we can retrieve exam name and exam time from the value of the check box by performing the following three steps.

```php
=pos = strpos($mygroup, '-');
$myexam_name = substr($mygroup, 0, $pos);
$myexam_time = substr($mygroup, $pos+1);
```

// Query database to check if the student already joined this study group.

// If student didn’t join this study group yet, add student to the group by using INSER INTO statement.

// If this study group has somebody else besides current student, mail the group list to group members by using the function mail(mail-to list, subject, message)

} else // Show study-groups form to student

// Calculate first and second available exam time.
$cur_year = intval(date('Y'));
$next_year = $cur_year + 1;
$month = intval(date('n'));
if ($month >= 1 && $month <= 7) {
  $first_choice = "$cur_year Fall";
  $second_choice = "$next_year Spring";
} else {
  $first_choice = "$next_year Spring";
  $second_choice = "$next_year Fall";
}

// HTML form for study groups
echo "<form method="post" action="{$_SERVER['PHP_SELF']}">
  <table>
    $q_exams = "SELECT exam_name FROM exams order by ts";
4.3 Searching Questions in an Exam

In an exam, both students and professors can search questions by topics, question types and exam time. The search form is designed in Figure 4.4.

The drop-down list box associated with each search criterion saves the space on the web page and makes pages simple and clean. Here is the form code:

```php
$query = "SELECT topic_name FROM topics WHERE exam_name='" . $exam_name . "';
if (isset($_SESSION['search_topics'])) {
    $select_topics = $_SESSION['search_topics'];
} else {
    $select_topics = NULL;
}
```

```php
click_menu($query, $conn, 'topic_name', $select_topics);
```
Figure 4.4. Design for search form.

```php
$cur_year = intval(date('Y')); $next_year = $cur_year + 1;
```
$query = "SELECT etime FROM exam_time where etime < '$next_year' OR etime='Other (Not Actually Asked Before)'';"
if (isset($_SESSION['search_etimes'])) {
    $select_etimes = $_SESSION['search_etimes'];
}
else {
    $select_etimes = NULL;
}
sel_menu($query, $conn, 'etime', $select_etimes);
echo "</li>";
echo "</ul>";
echo "<input type="submit" value="Search" name="subSearch" />";
echo "</form>"

The $_SESSION variables are used to remember previous search parameters and the helping function sel_menu is shown in section 3.2.2. To select multiple values for each search criterion, we need to put [] at the end of the form element name so that the PHP interpreter knows to accept multiple values.

The logic to process the search form is shown below:

- If one or more topics are selected:
  - Query database table 'ques_link_topics' to find all qids with selected topics.
- If one or more question types are selected:
  - Construct the question type list.
- If one or more entries in exam time are selected:
  - Construct exam time list.
  - Generate the query with found qids, the question type list and the exam time list.
  - Query database table ‘questions’ by using the query generated above.
  - Get the result sets from the database and display the search result to the website.

### 4.4 Adding New Question to an Exam

Only professors are allowed to add new questions to an exam. The new question form is designed in Figure 4.5.

When entering a new question, professor must select topic(s), question type and exam time, and answer all True/False questions. Moreover, the question text field and the exhibit field cannot be both empty. Here is the PHP code to accomplish this:
Figure 4.5. Design for new question form.

// Validate new question form

// Array to hold form errors
if (isset($field_err)) {
    unset($field_err);
}
$field_err = array();

// Count the number of form errors
if (isset($field_ecnt)) {
    unset($field_ecnt);
}
$field_ecnt = 0;

if (!isset($_REQUEST['new_topics'])) {
    $field_err[] = "You forgot to select "Topic(s)" field."
    $field_ecnt++;
}
if (!isset($_REQUEST['new_qtype'])) {
    $field_err[] = "You forgot to select ""Question Type"" field.";
    $field_ecnt++;
}
if (!isset($_REQUEST['new_etime'])) {
    $field_err[] = "You forgot to select ""Exam Time"" field.";
    $field_ecnt++;
}
if (!isset($_REQUEST['q_has_exhibit'])) {
    $field_err[] = "You forgot to select ""Question has Exhibit?"" field.";
    $field_ecnt++;
}
if (!isset($_REQUEST['has_solution'])) {
    $field_err[] = "You forgot to select ""Solution provided?"" field.";
    $field_ecnt++;
}
if (!isset($_REQUEST['s_has_choices'])) {
    $field_err[] = "You forgot to select ""Multiple-Choice Question?""";
    $field_ecnt++;
}
if (!isset($_REQUEST['s_has_mulAns'])) {
    $field_err[] = "You forgot to select ""Multiple-Choice with Multiple-Answers" field.";
    $field_ecnt++;
}

// Question text field and exhibit field cannot be both empty.
if ( (!isset($_REQUEST['question']) || $_REQUEST['question'] == "") &&
    (!isset($_FILES['q_exhibit']) ||
     !file_exists($_FILES['q_exhibit']['tmp_name']) ||
     $_FILES['q_exhibit']['size'] <= 0 ) ) {
    $field_err[] = "You forgot to fill in the ""Question"" field or you "
    ."forgot to upload a file to ""Question Exhibit"" field.";
    $field_ecnt++;
}
if ( $_REQUEST['q_has_exhibit'] == 1 &&
    ( !isset($_FILES['q_exhibit']) ||
    !file_exists($_FILES['q_exhibit']['tmp_name']) ||
    $_FILES['q_exhibit']['size'] <= 0 ) ) {
    $field_err[] = "You have set "Question has exhibit?" field to True. " ."But you forgot to upload a file for "Question Exhibit"”;
    $field_ecnt++;
}

if ( file_exists($_FILES['q_exhibit']['tmp_name']) &&
    i) $_REQUEST['q_has_exhibit'] == 0 ) {
    $field_err[] = "You have uploaded a file for "Question Exhibit" field" ."but you forgot to change "Question has Exhibit?" field to True.";
    $field_ecnt++;
}

if ($_REQUEST['s_has_mulAns'] == 1 && $_REQUEST['s_has_choices'] == 0) {
    $field_err[] = "You have set TRUE for "Multiple-Choice Question with" ."Multiple-Answers?" field, but you forgot to change" ." "Multiple-Choice Question?" field to True.”;
    $field_ecnt++;
}

if ($field_ecnt > 0) {
    foreach ($field_err as $f_error) {
        echo $f_error;
    }
    echo "Please press your browser’s BACK button and correct this.”;
}

If the new question is a non-choice question with solution provided, the “Save” button in the new question form will bring the user to the next form to add a solution to this new question as shown in Figure 4.6.

If the new question has choices, the adding-choice form will be automatically prompted after the user clicks the “Save” button. The form is shown in Figure 4.7. The user can select a choice letter from a drop-down list and fill in the text or exhibit for that choice.
Figure 4.6. Form to add a solution to the new question.

Figure 4.7. Form to add choices to the new question.
At least two choices must be selected and no two choices should have the same representation letter. If the new question is a multiple choice with single answer, then the user can only select one answer; if the new question is a multiple choice with multiple answers, than at least two answers should be provided.

Here is the form validation code for choice form:

```
// Validate choice form
if (isset($choF_err)) {
    unset($choF_err);
}
$choF_err = array();
if (isset($choF_ecnt)) {
    unset($choF_ecnt);
}
$choF_ecnt = 0;

if (isset($same_selC)) {
    unset($same_selC);
}
$same_selC = array();
if (isset($num_sameC)) {
    unset($num_sameC);
}
$num_sameC = 0;

// Total number of answers
if (isset($num_ans)) {
    unset($num_ans);
}
$num_ans = 0;

// Number of real choice(s) selected
if (isset($num_mysel)) {
    unset($num_mysel);
}
$num_mysel = 0;

for ($i = 0; $i < $desired_numCho; $i++) {
```
// This choice is selected:
if ($_REQUEST['cho'][$i] != 'NotSelected') {

    // Check duplicate choice letter
    foreach ($_REQUEST['cho'] as $k => $cval) {

        // Found duplicate choice letter
        if ($k != $i && $cval == $_REQUEST['cho'][$i]) {
            $flag_was_rem = 0;
            foreach($sameSelC as $sameC) {
                if ($_REQUEST['cho'][$i] == $sameC) {

                    // The duplicate choice letter has already be remembered.
                    $flag_was_rem = 1;
                    break;
                }
            }
            // Remember this duplicate choice letter:
            if ($flag_was_rem == 0) {
                $sameSelC[] = $_REQUEST['cho'][$i];
                $num_sameC++;
            }
            break;
        }
    }

    // If the choice is selected, choice’s solution part and exhibit
    // part cannot be empty at the same time.
    if ( (!isset($_REQUEST['sol'][$i]) || $_REQUEST['sol'][$i] == "")&&
         (!isset($_FILES['sEx_'.$i]) ||
          !file_exists($_FILES['sEx_'.$i]["tmp_name"]) ||
          $_FILES['sEx_'.$i]['size'] <= 0 ) ) {
        $mycho_inx = $i + 1;
        $choF_err[] = "Error for Choice number $mycho_inx: You forgot to"
                      ."fill in the "Solution" field or you forgot to "
                      ."upload a file to "Solution Exhibit" field."
        $choF_ecnt++;
    }
}
if ($_REQUEST['sHasEx'][$i] == 1 && (!isset($_FILES['sEx_'.$i]) || !file_exists($_FILES['sEx_'.$i]['tmp_name']) || $_FILES['sEx_'.$i]['size'] <= 0)) {
    $mycho_inx = $i + 1;
    $choF_err[] = "Error for Choice number $mycho_inx: You have set "
    .""Solution has exhibit?" field to True. But you forgot"
    ." to upload a file for "Solution Exhibit" field.";
    $choF_ecnt++;
}
if (file_exists($_FILES['sEx_'.$i]['tmp_name']) &&
    $_REQUEST['sHasEx'][$i] == 0) {
    $mycho_inx = $i + 1;
    $choF_err[] = "Error for Choice number $mycho_inx: You have "
    ."uploaded a file for "Solution Exhibit" field, but "
    ."you forgot to change "Solution has Exhibit?" to True.";
    $choF_ecnt++;
}
if ($_REQUEST['isAns'][$i] == 1) {
    $num_ans++;
}
$num_mysel++;
}
// end outer-most for loop
if ($num_mysel < 2) {
    $choF_err[] = "You should select at least two choices.";
    $choF_ecnt++;
}
if ($num_sameC > 0) {
    foreach ($same_selC as $sameC) {
        $choF_err[] = "Some selected choices have the same 
        ."representation letter "$sameC". ";
        $choF_ecnt++;
    }
}
if (($s_has_mulAns == 0) && ($num_ans > 1)) {
    $choF_err[] = "This is a Multiple-Choice with Single-Answer question, 
    ."but you provided more than one answer.";
    $choF_ecnt++;
4.5 Editing Question in an Exam

Again, only professors have the privilege to edit an existing question. Since the question already existed, we can tell whether it is a choice question or a non-choice question and display it in the right format. The form to edit a non-choice question is shown in Figure 4.8, and the form to edit a choice question is shown in Figure 4.9.

The number of choices displayed in Figure 4.9 is not fixed; instead it depends on the real number of choices associated with the question. The form handling and processing are similar to the new question form.

4.6 Adding Comments to a Question

Both professors and students can add comments to a question for discussion. The form to add comment is shown in Figure 4.10.

The PHP code to accomplish this is shown below:

```php
// If user did not clicks "See All Comments" button, then display
// this button only; otherwise, show all comments.
if (!isset($_REQUEST['see_comments'])) {
    echo "<form method="post" action="$_SERVER["PHP_SELF'"]? "
    . "exam_name=$exam_name&pagenum=$pagenum&qid=$qid"">
    echo "<input type="submit" name="see_comments" 
    . "value="See All Comments">";
    echo "</form>";
}
else { // Display all comments
```
Figure 4.8. Form to edit non-choice question.
Figure 4.9. Form to edit choice question.
Figure 4.10. Display all comments to a question.

```php
$query_comments = "select * from comments where qid=$qid"
    . " order by cid desc";
$result_comments = mysql_query($query_comments, $conn) or
die(mysql_error());
echo "<table id="all_coms">";
while($row_comments = mysql_fetch_array($result_comments)) {
    $cid = $row_comments['cid'];
    $uemail = $row_comments['uemail'];
    $comment = stripslashes($row_comments['comment']);
    $ts = $row_comments['ts'];

    // found user's full name form table 'users'
    $query_users = "select * from users where email='$uemail'";
    $result_users = mysql_query($query_users, $conn) or
die(mysql_error());
    $row_users = mysql_fetch_array($result_users);
    $full_name = $row_users['first_name']." ". $row_users['last_name'];

echo "<tr><td class="single_com">";
```
// If user clicks "Add Comment" button, insert comment form input
// to the database table 'comments'.
if (isset($_REQUEST[add_com])) {
    if (!isset($_REQUEST[com]) || $_REQUEST[com] == "") {
        echo "Your comment field is empty. Please try again.";
    } else {
        $com = addslashes(htmlspecialchars($_REQUEST[com]));
        $query_com = "insert into comments set qid=$qid, 
                     ." uemail='$_SESSION[user_email]', 
                     ." comment='$com' ";
        $result_com = mysql_query($query_com, $conn) or die(mysql_error());
        // Update table 'questions' and set question's "has_comments"
        // field to true.
        if ($has_comments == 0) {
            $query_q = "update questions set has_comments=1 where qid=$qid";
            $result_q = mysql_query($query_q, $conn) or die(mysql_error());
            $has_comments = 1;
            show_comments($conn, $qid, $exam_name, $pagenum);
        }
        echo "You have successfully added a new comment to this question.");
        . "Press \"See All Comments\" to view all comments.";
    }
} else {
    $com = addslashes(htmlspecialchars($_REQUEST[com]));
    $query_com = "insert into comments set qid=$qid, 
                 ." uemail='$_SESSION[user_email]', 
                 ." comment='$com' ";
    $result_com = mysql_query($query_com, $conn) or die(mysql_error());
    // Update table 'questions' and set question's "has_comments"
    // field to true.
    if ($has_comments == 0) {
        $query_q = "update questions set has_comments=1 where qid=$qid";
        $result_q = mysql_query($query_q, $conn) or die(mysql_error());
        $has_comments = 1;
        show_comments($conn, $qid, $exam_name, $pagenum);
    }
    echo "You have successfully added a new comment to this question.");
    . "Press \"See All Comments\" to view all comments.";
}
4.7 Self-Assessment for Multiple-Choice Question

The user can see self-assessment for multiple-choice question. The page design is shown in Figure 4.11.

![Check Solution for Multiple Choice](image)

**Figure 4.11. Check Solution for multiple choices.**

If the choice question has only one answer, the radio button would be displayed in front of each option; if the choice question has multiple answers, then the check box would show up for each option.

After the user has selected one option and clicked “Check Your Solution” button, the result page would be displayed as in Figure 4.12. The user can check the solution many times until the result is correct.

The PHP code to accomplish this is shown below:

```php
// Use $_SESSION variables to remember user’s choice(s).
if (isset($_REQUEST['choAns'][$qid])) {
    $_SESSION['show_choAns'][$qid] = $_REQUEST['choAns'][$qid];
}
```
// Array to remember correct answer
if (isset($correct_answer)) {
    unset($correct_answer);
}
$correct_answer = array();

// Show all choice options:
$q_sols = "select * from solutions where qid=$qid";
    $res_sols = mysql_query($q_sols, $conn) or die(mysql_error ());

echo "<form method="post" action="{$_SERVER['PHP_SELF']}? "
    . "exam_name=$exam_name&pagenum=$pagenum&qid=$qid">";

while ($row = mysql_fetch_array($res_sols)) {
    $sid = $row['sid'];
    $choice = $row['choice'];
    $solution = $row['solution'];
    $s_has_exhibit = $row['s_has_exhibit'];
    $s_exhibit_type = $row['s_exhibit_type'];
    $s_exhibit = $row['s_exhibit'];
    $isAnswer = $row['isAnswer'];
    $isVerified = $row['isVerified'];
    $info = $row['info'];

    if ($isAnswer) {
        $correct_answer[] = $choice;
    }
}
if ($s_has_mulAns) {// Check box for multiple answers
    echo "<input type="checkbox" name="choAns[$qid][]" "
        . "value="$choice" ";
}
else { // Radio button for single answer
    echo "<input type="radio" name="choAns[$qid][]" "
        . "value="$choice" ";
}

// Display use’s choice(s):
if (isset($_SESSION['show_choAns'][$qid])) {
    foreach ($_SESSION['show_choAns'][$qid] as $an) {
        if ($an == $choice) {
            echo " checked="yes" ";
        }
    }
}
else {
    echo " / $choice: ";
}

if ($s_has_exhibit) {
    $s_file = "mytmp/".session_id().date('YnjHis')
        . ".sEx".$.id.".".s_exhibit_type;
    $show_text = "EXHIBIT for $choice";
    show_exhibit($s_exhibit, $s_file, $show_text, $s_exhibit_type);
}
else {
    echo "<br/>";
}
//end while loop

// If user clicks "Check Your Solution" button,
// show self-assessment result.
if (isset($_REQUEST['checkMyS'])) { //check solution and show result
    echo "<table width="55" border="2">
        <tr><td width="30"><b>Your Answer:</b></td>
        <td width="25"><b>";
    
    foreach ($_SESSION['show_choAns'][$qid] as $an) {
        if ($an == $choice) {
            echo " checked="yes" ";
        }
    }
}
    echo "</form>
";}
```php
$flag = 0; // 0 means wrong result
if (count($correct_answer) == count($_SESSION['show_choAns'][$qid])) {
    $foundCnt = 0; // How many use's choices are correct.
    foreach ($correct_answer as $c_an) {
        foreach ($_SESSION['show_choAns'][$qid] as $an) {
            if ($c_an == $an) {
                $foundCnt++;
                break;
            }
        }
    }
    // All use's choices are correct.
    if ($foundCnt == count($correct_answer)) {
        $flag = 1;
    }
}

if ($flag == 1) { // Correct result
    echo "<td width="25"><img src="css/images/correct.jpg" "
    . "height="25" width="25"></td>";
} else { // Wrong result
    echo "<td width="25"><img src="css/images/wrong.jpg" "
    . "height="25" width="25"></td>";
}
echo "</tr></table>";
```
CHAPTER 5

APPLICATION

The objective of this project is to implement an online test preparation system to help computer science graduate students to pass their comprehensive exams. Two types of users, professors and students, have different privileges. Students have an option to join a study group(s) to communicate with other students, and they can also view or search questions by topics, question type and exam time. Professors are allowed to search, add, delete and modify questions and solutions. To assist discussion among students and between professors and students, both students and professors can add their comments to a question.

To gain access to the system each user should be authenticated. When the user types the website’s URL, the login page is displayed as shown in Figure 5.1.

Figure 5.1. Login page.

Professors must use their email addresses posted at http://www.cs.sdsu.edu/faculty/index.html, a public website of the Computer Science Department at SDSU; the system can automatically identify the user level based on a login email. For the user who does not have an account, the link located at the bottom of the login page will bring him/her to the registration page as shown in Figure 5.2.
To secure use of the system, the first-time registrants will receive their login password through their login email; thus a student cannot pretend to be a professor.

Next, we will show the application for different user levels: Student and Professor.

5.1 STUDENT

Once a student logs into the system, the page with the menu to choose study group(s) will be displayed, as shown in Figure 5.3.
If the student selects one or more study groups, the group list will be mailed to the student’s login email. By pressing the “NEXT” button with or without selection of study group(s), the student is directed to the index page, which is shown in Figure 5.4.

![Image](image.png)

**Figure 5.4. Index page.**

A student can select any exam from the index page by clicking the name of the exam. Figure 5.5 shows an example of the questions page in the selected exam.

The left column of the questions page supports search functionality for the student. The student can search questions by topics, question types and exam time. Figure 5.6 shows an example of the search result.

If the student clicks the “Add comment or view solution to this question” link located at the end of each question, the selected question page would be displayed as in Figure 5.7.

The student can also view a solution and add a comment to the selected question through the text area located at the bottom of the page as shown in Figure 5.8.

If the student wants to view questions in a different exam, the “Home” sign located at the top area will direct the student to the index page. Then the student can choose another exam and view or search questions in that exam.

Once the student logs in, the “Change Password” sign either located at the left column or the top area will bring the student to the change password page, as shown in Figure 5.9. Now, it’s time to change the machine-generated password to something easy to remember.
Figure 5.5. Student’s question page in the selected exam.

Figure 5.6. Search result page.
Figure 5.7. Student’s selected question page.

Figure 5.8. Comments page/view solution page.
5.2 Professor

If the user logs into the system as a professor, then the user is directed to the index page, as shown in Figure 5.4 (p. 69). After selecting an exam, the questions page in the selected exam will be displayed as shown in Figure 5.10. Compared with the student’s questions page, this page has an additional “Add new question” link located at the top-right area to allow professors to add a new question to the selected exam.

Figure 5.10. Professor’s questions page in a selected exam.
The “Add new question” link will bring the professor to the new question page, as shown in Figure 5.11. Through this process a professor can add different types of questions by selecting topics, question-types and exam time. Professors also have the ability to add special symbols used in different exams.

If the new question is a non-choice question with solution provided, the “Save” button will direct the professor to the solution page as in Figure 5.12.

If the new question is a choice question, the “Save” button in the new question page will bring the choices page to the professor as shown in Figure 5.13.

The “Add comment or view solution to this question” link located at the end of each question in the questions page will bring the selected question page to the professor. The selected question page is shown in Figure 5.14 with “Edit” link at the top of the question.

If the question has no choices, the “Edit” link would bring the non-choice question edit page to the professor; otherwise, the choice question edit page will be displayed. Those two types of edit page are shown in Figure 5.15 and Figure 5.16.
Figure 5.12. Solution page.
Figure 5.13. Choices page.
Figure 5.14. Professor’s selected question page.
Figure 5.15. Non-choice question edit page.
Figure 5.16. Choice question edit page.

At the end of those edit pages, there are three options: Save, Delete, Cancel, as shown in Figure 5.17.

The “Save” button will save the changes made to the question, while the “Delete” button is used to delete this question from the database. And the “Cancel” link will bring the professor back to the questions page.

The professor can also search question, add a comment and change password just like what the student did in section 5.1.
Figure 5.17. Three options to manipulate the question.
CHAPTER 6

FUTURE WORK

There are certain areas where incremental improvements and enhancements can be made.

This system has no administrator interface to control all the database tables. If an administrator wants to delete an already graduated student, add new exam, topics, question types than he can do it manually from the backend. An administrator web-interface can be developed as a future enhancement.

The current system only serves as a test aid to help students prepare their tests. In the next step, we can build a test generator to create real time tests and let students remotely take the tests online.

Another enhancement would be the database backup and recovery to minimize the data loss due to the corruption, crash or hardware failure.
CHAPTER 7

CONCLUSION

This project implements a web-based system to help computer science graduate students prepare for their comprehensive exams.

Students can use this system to join study groups, find study partners, view questions, search questions based on different criteria, check solutions, add or view comments. Professors can search, add, update questions and solutions, and add comments to communicate with students. Also there is a facility for both students and professors to add special symbols while adding and updating comments and questions.

Thus, this system can become an important means of communication between students and professors or among students.

The implementation of the project has given a accurate knowledge of how PHP is used to create web pages and how to connect to MySQL database. Overall the development of this project was an excellent learning opportunity.
REFERENCES


