A Wiki paradigm to manage online course content

Elharith Omer Elrufaie

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

Part of the Databases and Information Systems Commons

Recommended Citation
Elrufaie, Elharith Omer, "A Wiki paradigm to manage online course content" (2004). Theses Digitization Project. 2446.
https://scholarworks.lib.csusb.edu/etd-project/2446

This Project is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.
A WIKI PARADIGM TO MANAGE ONLINE COURSE CONTENT

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

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

by
Elharith Omer Elrufaie
June 2004
A WIKI PARADIGM TO MANAGE ONLINE COURSE CONTENT

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

by
Elharith Omer Elrufaie
June 2004

Approved by:
Dr. David Turner, Chair, Computer Science
Dr. Richard Botting
Dr. Tong Lai Yu

6/2/2004
ABSTRACT

The term Wiki means quick in Hawaiian. Wiki, by creator Ward Cunningham's original description, is "the simplest online database that could possibly work!" A wiki allows users to freely create and edit web page content using any regular web browser. Consequently, a wiki page is a web page that can be created and edited in a quick and easy way. In this context, the wiki paradigm can be used to enable a quick, easy and convenient management system of online course content. However, some limitations exist in currently available wiki systems that must first be handled to enable successful online course management. In this project, I have improved a previous work done by a CSUSB alumnus to extend the wiki paradigm for use in classroom. In this new version of the wiki-based course management system, the system works as a generic teaching tool for any information technology department, where there exist many instructors, courses and students. Instructors and students will contribute content using XHTML markup instead of the traditional wiki markup language which adds an excellent learning value to the system.
ACKNOWLEDGMENTS

I would like to dedicate my work to my family, especially to my father and mother, for being such a good example to follow, and for their continuous encouragement and support even when being a part. They have always believed in me and never doubted my potential.

My sincere thanks goes to my advisor, Dr. David Turner, for his continuous help to promote my technical and research skill. I appreciate his enthusiasm for his work and for being such a nice, supportive and helpful advisor. I also would like to thank Vishal Dharod, Buket Tuna and Volkan Uzun for being such good friends in this place. Additionally, I'd like to thank my friends Ibrahim and Ali Mamoun for their useful and valuable advices and guidance through my years in the United States.

Finally, I would like to thank the department of Computer Science at CSUSB for supporting my mission to pursue my M.S degree.

The support of the National Science Foundation (NSF) under award 9810708 is gratefully acknowledged.

Elharith Elrufaie
# TABLE OF CONTENTS

ABSTRACT ................................................................. iii

ACKNOWLEDGMENTS ......................................................... iv

LIST OF TABLES ............................................................. viii

LIST OF FIGURES ............................................................ ix

CHAPTER ONE: SOFTWARE REQUIREMENT SPECIFICATION

1.1 Introduction ........................................................... 1

1.2 Purpose of this Project ............................................. 1

1.2 Context of the Problem ............................................. 2

1.3 Project Products .................................................... 3

1.4 Definition of Terms and Abbreviations ......................... 3

1.5 Preliminary Design .................................................. 5

  1.5.1 Users Characteristics ......................................... 5

  1.5.2 Users Functions ................................................. 6

CHAPTER TWO: SYSTEM DESIGN

2.1 Architecture .......................................................... 10

  2.1.1 Client Tier ...................................................... 10

  2.1.2 Middle Tier .................................................... 10

  2.1.3 Database Tier ................................................. 11

2.2 Database Design .................................................... 11

  2.2.1 Database Schema Conceptual Model ......................... 11

  2.2.2 Database Schema Logical Model ......................... 12

2.3 System Components ................................................ 16

2.4 Software Interfaces ............................................... 17
CHAPTER THREE: IMPLEMENTATION

3.1 Introduction .............................................. 18
3.2 Basic Interface ........................................... 18
  3.2.1 Authorization and Authentication ............. 18
  3.2.2 New User Registration .................. 19
  3.2.3 Student Main Menu ...................... 20
  3.2.4 Student Add Course Page ............... 21
  3.2.5 Student Edit Information Page ........ 22
  3.2.6 Instructor Main Menu ................. 23
  3.2.7 Instructor Add Courses ................. 25
  3.2.8 View Course Details .................. 26
  3.2.9 Administrator Main Menu ............ 27
  3.2.10 Edit Users' Information Page .......... 28
  3.2.11 XHTML Edit Page ..................... 30

CHAPTER FOUR: VERIFICATION AND VALIDATION

4.1 Unit Test .............................................. 33
4.2 Subsystem Testing ................................. 35
4.3 System Testing ....................................... 37

CHAPTER FIVE: MAINTENANCE MANUAL

5.1 Software Installation ................................. 39
  5.1.1 RedHat Installation ...................... 39
  5.1.2 PostgreSQL Installation ................. 40
  5.1.3 JAVA 2 Platform, Standard Edition (J2SE) .... 42
CHAPTER FIVE: INSTALLATION AND CONFIGURATION

5.1 Variables Modification

5.1.4 Tomcat ........................................ 43

5.1.5 JAVA Database Connectivity (JDBC) .... 46

5.2 Variables Modification .......................... 47

5.2.1 System Variables .............................. 47

5.3 Installation/Migration ............................ 49

5.4 Backup and Restore .............................. 49

5.4.1 System Backup ................................. 49

5.4.2 Database Backup ............................... 50

5.4.3 System Restore ................................. 50

5.4.4 Database Restore ............................... 50

CHAPTER SIX: CONCLUSION AND FUTURE DIRECTIONS

6.1 Conclusion ....................................... 52

REFERENCES ......................................... 54
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Terms and Abbreviations</td>
<td>4</td>
</tr>
<tr>
<td>Table 2</td>
<td>Users Table</td>
<td>13</td>
</tr>
<tr>
<td>Table 3</td>
<td>Course Table</td>
<td>14</td>
</tr>
<tr>
<td>Table 4</td>
<td>Student Status Table</td>
<td>15</td>
</tr>
<tr>
<td>Table 5</td>
<td>Page Table</td>
<td>16</td>
</tr>
<tr>
<td>Table 6</td>
<td>Unit Test Results (Forms)</td>
<td>34</td>
</tr>
<tr>
<td>Table 7</td>
<td>Unit Test Results (Class: User)</td>
<td>35</td>
</tr>
<tr>
<td>Table 8</td>
<td>Unit Test Results</td>
<td>35</td>
</tr>
<tr>
<td>Table 9</td>
<td>Subsystem Test Results</td>
<td>36</td>
</tr>
<tr>
<td>Table 10</td>
<td>System Test Results</td>
<td>38</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1. Use Case Diagram ........................................ 7
Figure 2. System Architecture ...................................... 10
Figure 3. Entity Relationship Diagram .............................. 12
Figure 4. Login Page .................................................. 19
Figure 5. New User Registration ..................................... 20
Figure 6. Student Homepage .......................................... 21
Figure 7. Student to Add New Courses ............................. 22
Figure 8. Student Edit Information Page .......................... 23
Figure 9. Instructor Main Page ...................................... 24
Figure 10. View Student Information ............................... 25
Figure 11. Instructor to Create New Courses .................... 26
Figure 12. Course Information ...................................... 27
Figure 13. Administrator Main Page ............................... 28
Figure 14. Administrator View Users .............................. 29
Figure 15. Administrator Edit Users ............................... 30
Figure 15. XHTML Edit Page ........................................ 31
Figure 16. XHTML Error Page ...................................... 32
CHAPTER ONE
SOFTWARE REQUIREMENT SPECIFICATION

1.1 Introduction

This system will run as a communication interface between students and the instructor. Using this interface, pages are created and modified quickly and easily. An instructor will be able to publish course materials, post announcements, presentations timetable, etc into the web. If the student is enrolled in the course, she will be able to edit these pages and can add, for example, her presentation title to the presentations timetable, or her paper abstract to the term paper page, etc. Moreover, all the pages can be used as a public or private repository, in which students post their homework and limit access of these materials to the instructor.

1.2 Purpose of this Project

The purpose of this project is to develop a new version of the Wiki-Style Administration of Online Course Content. The current version of this system was developed by Jimmy Wang in the fall of 2003 in fulfillment of his Master degree requirements [6]. The main purpose is to implement a teaching and a learning tool that works as an
easy and quick communication interface between each instructor and his students. By teaching tool, we mean that both instructor and students can post coursework, and share information and knowledge in the course web page. By learning, we mean that users will be required to use XHTML [8] for input; all input is parsed to determine if it is valid XHTML text, and will not be accepted otherwise. This system extends the traditional wiki by providing authorization functionality, that is, only authorized users (per class) are eligible to make changes in course pages.

The second purpose is to design an easily extendable and maintainable architecture, which provides a generic wiki system that can work for any information technology department, and handles sets of courses and instructors.

1.2 Context of the Problem

The context of the problem is to extend the traditional wiki paradigm and the current running Wiki-Style Administration of Online Course Content by designing and implementing a system that eliminates current limitations and adds learning benefit.
1.3 Project Products

This project would lead to the following products:

- **Web Application**: A web application that follows the Model View Controller (MVC) architecture, uses PostgreSQL database for storing data, and uses Hibernate as its persistence framework. This web application should achieve the needs of a communication interface between instructor and students. It should also serve as a learning tool, because it requires the use of XHTML as input markup language.

- **Users manual**: an implementation manual will be available for the user.

- **Systems Manual**: a project report (this report) will be available with design details and specifications.

1.4 Definition of Terms and Abbreviations

See table 1 for list of terms and abbreviations.
## Table 1. Terms and Abbreviations

<table>
<thead>
<tr>
<th>API</th>
<th>Applications Programming Interface.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSCI</td>
<td>Computer Science</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets.</td>
</tr>
<tr>
<td>CSUSB</td>
<td>California State University, San Bernardino.</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface.</td>
</tr>
<tr>
<td>HIBERNATE</td>
<td>It is object/relational persistence and query service for Java. Hibernate lets you develop persistent objects following common Java idiom - including association, inheritance, polymorphism, composition and the Java collections framework.</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language.</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol.</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers.</td>
</tr>
<tr>
<td>Java Servlet</td>
<td>A Servlet can be thought of as an applet that runs on the server side -- without a face.</td>
</tr>
<tr>
<td>JDBC</td>
<td>Java Database Connectivity.</td>
</tr>
<tr>
<td>JSP</td>
<td>Java Server Page.</td>
</tr>
<tr>
<td>MVC</td>
<td>Model View Controller.</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>The most advanced open source database system.</td>
</tr>
<tr>
<td>RDBS</td>
<td>A relational database stores data in tables (relations). A database is a collection of tables.</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language.</td>
</tr>
<tr>
<td>SRS</td>
<td>Software Requirement Specification.</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Socket Layer</td>
</tr>
<tr>
<td>Tomcat</td>
<td>Tomcat is the Servlet container that is used in the official Reference Implementation for the Java Servlet and Java Server Pages technologies.</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator.</td>
</tr>
<tr>
<td>XHTML</td>
<td>eXtensible Hypertext Transfer Protocol.</td>
</tr>
</tbody>
</table>
1.5 Preliminary Design

There are four types of user classes implemented in this project:

1. Registered Student.
2. Instructor.
4. Guest.

A menu page is associated with each type of user. The pages will allow the user to enter, retrieve and/or maintain the data depending on the privileges.

1.5.1 Users Characteristics

Users of the Wiki system would fall into one of the following groups:

1.5.1.1 Student. The user in this group is enrolled in a class at CSCI department, CSUSB. To use the system, the user needs to obtain a login name and password. Once a student adds a class, his class instructor would be the only one to activate his membership to this class. If the instructor accepted him, after comparing his information with the class roster, the student status is automatically changed to active, and afterwards the student can edit and create pages in the course homepage. Otherwise, the student status is set to rejected. In
either case, the student is informed by email of any change in his status.

1.5.1.2 Instructor. The user of this group is a faculty in the Department of Computer Science, CSUSB. Registration of instructors is activated upon approval of administrator.

1.5.1.3 Administrator. The user in this group could be a faculty, or a staff, or any system administrator.

1.5.2 Users Functions

The following use case diagram best describes user functionalities.
Figure 1. Use Case Diagram
1.5.2.1 Student functions. This menu page provides four options for this class of user:

1. Create pages in the wiki system.
2. Delete pages created by him.
3. Set properties of each page, such as read-only or private.
4. Modify pages in wiki system, unless the page is read-only.
5. Modify personal information.
6. Add and remove courses.

1.5.2.2 Instructor functions. This menu has the same functionality as the student menu page in addition to the following:

1. Start a new class on the system.
2. Change the status of his classes (active or inactive).
3. Approve and reject students on his classes.
4. View his students' information.
5. Contact all his students by email.

1.5.2.3 Administrator functions. This menu page provides the same functionality as instructor menu in addition to:
1. Approve and reject pending instructors.

2. Delete a user.

3. Edit user’s information.

1.5.2.4 Guest functions. Guest user will have the ability to view the public pages only, with no authority to edit any page.
CHAPTER TWO
SYSTEM DESIGN

2.1 Architecture

This project follows a 3-tier architecture. The following figure best describes this architecture:

![System Architecture Diagram]

Figure 2. System Architecture

2.1.1 Client Tier

The first tier is the client that represents web browser viewing the system pages.

2.1.2 Middle Tier

The middle is the wiki Server that contains Jakarta Tomcat web server module (with JSP and Servlet API's).

This tier follows the Model View Controller (MVC) architecture. In the MVC design pattern, the application server is segregated into three parts:
1. Model: Stores the data and deals with all the data retrieval issues.

2. View: Displays information and results.

3. Controller: Resolves all business issues, calls appropriate methods in model to forward back to the view.

2.1.3 Database Tier

The third tier is the data source, where Hibernate persistent objects framework maps Java objects to a PostgreSQL relational database management system.

2.2 Database Design

2.2.1 Database Schema Conceptual Model

In designing the schema of this project, three distinct parts have been identified:

1. Courses: collection of courses that are created by faculty or administrator and made available for students to add.

2. Pages: includes all wiki pages information, such as title, contents and page settings.

3. Users: Defines system user and his roles. Every user in the system has a relation to some courses. Meaning that a user can have courses
that she created (if administrator or instructor), or added (if student).

The following Entity Relationship diagram (ER) best describes the system.

![Entity Relationship Diagram](image)

Figure 3. Entity Relationship Diagram

2.2.2 Database Schema Logical Model

The conceptual model ER diagram maps the following relational table design. In the following tables, underlined fields indicate the table primary key.
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>INT8</td>
<td>Auto incremented id.</td>
</tr>
<tr>
<td>role</td>
<td>VARCHAR(255)</td>
<td>Identifies user’s role. ‘A’ for Administrator, ‘I’ for Instructor and ‘S’ for Student.</td>
</tr>
<tr>
<td>username</td>
<td>VARCHAR(255)</td>
<td>Unique name for user.</td>
</tr>
<tr>
<td>password</td>
<td>VARCHAR(255)</td>
<td>Password for user to login.</td>
</tr>
<tr>
<td>firstName</td>
<td>VARCHAR(255)</td>
<td>User’s first name.</td>
</tr>
<tr>
<td>lastName</td>
<td>VARCHAR(255)</td>
<td>User’s last name.</td>
</tr>
<tr>
<td>email</td>
<td>VARCHAR(255)</td>
<td>User’s email.</td>
</tr>
<tr>
<td>status</td>
<td>VARCHAR(255)</td>
<td>Shows status of Instructor in the system. Switched from ‘PENDING’ to ‘ACTIVE’ for every activated instructor.</td>
</tr>
</tbody>
</table>

- Id field is auto incremented primary key.
- Class should not be null.
- username should not be null.
Table 3. Course Table

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>INT8</td>
<td>Auto incremented id.</td>
</tr>
<tr>
<td>instructor_id</td>
<td>INT8</td>
<td>Stores course creator if was Instructor.</td>
</tr>
<tr>
<td>admin_id</td>
<td>INT8</td>
<td>Stores course creator if was Administrator.</td>
</tr>
<tr>
<td>title</td>
<td>VARCHAR(255)</td>
<td>Course title.</td>
</tr>
<tr>
<td>quarter</td>
<td>VARCHAR(255)</td>
<td>Course Quarter.</td>
</tr>
<tr>
<td>year</td>
<td>INT4</td>
<td>Course Year.</td>
</tr>
<tr>
<td>status</td>
<td>VARCHAR(255)</td>
<td>Course status, can be 'ACTIVE' or 'INACTIVE'.</td>
</tr>
</tbody>
</table>

- id field is auto incremented primary key.
- instructor_id is a foreign key to the id field in Users table.
- admin_id is a foreign key to the id field in Users table.
Table 4. Student Status Table

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>student_id</td>
<td>INT8</td>
<td>Stores student id.</td>
</tr>
<tr>
<td>Course_id</td>
<td>VARCHAR(255)</td>
<td>Stores course id.</td>
</tr>
<tr>
<td>Status</td>
<td>VARCHAR(255)</td>
<td>Represent student’s status for each class.</td>
</tr>
</tbody>
</table>

- student_id is a foreign key to the id field in Users table.
- course_id is a foreign key to the id field in Course table.
- student_id and course_id forms a composite primary key.
Table 5. Page Table.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>INT8</td>
<td>Stores student id.</td>
</tr>
<tr>
<td>Name</td>
<td>VARCHAR(255)</td>
<td>Stores course id.</td>
</tr>
<tr>
<td>owner_id</td>
<td>VARCHAR(255)</td>
<td>Represent student's status for each class.</td>
</tr>
<tr>
<td>Content</td>
<td>TEXT</td>
<td>Stores page XHTML contents.</td>
</tr>
<tr>
<td>Visible</td>
<td>BOOL</td>
<td>Identifies if the page is visible to all users.</td>
</tr>
<tr>
<td>modifiable</td>
<td>BOOL</td>
<td>Identifies if the page can be modified by others.</td>
</tr>
</tbody>
</table>

- id is an auto incremented primary key.
- owner_id is a foreign key to the id field in the Users table.

2.3 System Components

To enable successful and flexible interaction between the system tiers, some languages, scripts and tools were used. Below I explain the reasons of using each of these technologies:

- Java was used for its reputation of robustness for web applications use. Java API has a rich set of functions and classes, which allows the
flexibility and the ability to build such web applications.

- PostgreSQL was used for data storage medium. PostgreSQL is a real multi-user database and is royalty-free open source software.
- Hibernate is a persistence service that stores Java objects in relational databases. Hibernate is a full-featured, open source mapping framework for the Java platform.

2.4 Software Interfaces

- Internet browser: No browser constraints.
- Operating System (OS): No OS constraints.
- Database: PostgreSQL.
- Web Server: Jakarta Tomcat 5.
- Java API: JDK 1.4.2.
CHAPTER THREE
IMPLEMENTATION

3.1 Introduction

In this chapter, we will discuss the system’s basic interface and functionality. One of the main goals of this new version of wiki is to develop a system with a good look and feel. For this purpose, I use Cascading Style Sheets (CSS) to develop the pages; I also use a set of contrasting colors. Every page is divided into two parts: menu options and body. Menu options vary according to the user role. However, the wiki part of the page in the body is the same for all users. The wiki part of the page checks the user input, and reports any invalid XHTML input with a friendly error message. In the following subsections, I describe in more detail the functionality of the system.

3.2 Basic Interface

3.2.1 Authorization and Authentication

This is the main login page (see figure 3). All users, students, instructors and the system administrator will use this page to log in. This page uses server side authentication and authorization using Java Servlets.
3.2.2 New User Registration

New students and instructors use this page to obtain username and password (see figure 4).
3.2.3 Student Main Menu

Student main page shows the student status in the wiki system. It lists the student's status per class. A student status can be active (if an instructor accepted him) so that she can edit the course page. Otherwise, the status is pending (waiting for course instructor authorization).
3.2.4 Student Add Course Page

The student uses this page to view all active courses in the system and any course to his records. The student is added directly with a pending status by default (see figure 6).
3.2.5 Student Edit Information Page

The student uses this page to view and edit his information, such as first name, last name, email and password (see figure 7).
3.2.6 Instructor Main Menu

The instructor’s main page shows a list of all students who want to enroll in their courses. The instructor is given the option to view their information, such as name and email to ensure they are activating the correct student. Activating a student will change the student status to active, thus she will be able to edit the course page (see figures 8 and 9).
Figure 9. Instructor Main Page
3.2.7 Instructor Add Courses

The instructor uses this page to view his current courses in the system and to create new courses as well (see figure 10).
Figure 11. Instructor to Create New Courses

3.2.8 View Course Details

Instructor uses this page to view students' status per course, and the course web page (see figure 11).
3.2.9 Administrator Main Menu

The administrator main page shows a list of all pending instructors and the administrator's pending students. Activating an instructor will allow the instructor to create new courses. Note that the instructor is activated once, unlike a student, who needs activation per course. The administrator is also given the option to view the pending instructors and student information such as name and email (see figure 12).
Figure 14. Administrator View Users
3.2.11 XHTML Edit Page

This is the most important page that the system has. To edit the page, browse to the page you want to edit and double click anywhere in the page body. The whole text of the page should be transformed into a large text area box to allow changing page contents. Input language in this version is XHTML, unlike the wiki markup language used in this first version of this project. If the user who is editing the page is the owner of the page, there will be two more check boxes shown at the bottom of the text field named "modifiable by users" and "visible by
others.” These are the page attributes that can be set by the owner of the page to grant or deny to other users the ability to view or modify the page (figure 15).

Figure 15. XHTML Edit Page
Error on line 24: The element type "p" must be terminated by the matching end-tag "</p>".

Figure 16. XHTML Error Page
CHAPTER FOUR
VERIFICATION AND VALIDATION

The system validation test is a kind of test process which can ensure that our program meets the expectation of the user. The purpose of the system validation is to provide a high degree of assurance that a specific process will consistently produce a result which meets predetermined specifications and quality attributes. This can also guarantee the system performance and reliability.

4.1 Unit Test

Unit test is the basic level of testing where individual components are tested to ensure that they operate correctly. These individual components can be object, class, program, etc.
Table 6. Unit Test Results (Forms)

<table>
<thead>
<tr>
<th>Forms</th>
<th>Tests Performed</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add User Page</td>
<td>• Verify handling valid data input.</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• Check all the buttons work properly.</td>
<td></td>
</tr>
<tr>
<td>Frame allocate page</td>
<td>• Verify the frame allocation is properly.</td>
<td>Pass</td>
</tr>
<tr>
<td>Edit Page</td>
<td>• Verify the timer work properly.</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• Check the insert function work properly.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Check all the button work properly.</td>
<td></td>
</tr>
<tr>
<td>Login Page</td>
<td>• Check all the button work properly.</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• Verify the page can get the error message and work properly by the message.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Verify the user save in session after login</td>
<td></td>
</tr>
<tr>
<td>Logout Page</td>
<td>• Check all the button work properly.</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• Verify the user remove from session after logout</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Check the page redirect to proper page after logout</td>
<td></td>
</tr>
<tr>
<td>Update My Account Page</td>
<td>• Check all the buttons work properly.</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• Verify the page get the correct user account information.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Verify handling valid data input.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Verify the data updated correctly.</td>
<td></td>
</tr>
</tbody>
</table>
Table 7. Unit Test Results (Class: User)

<table>
<thead>
<tr>
<th>Forms</th>
<th>Tests Performed</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_no</td>
<td>• Make sure the returned number is</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>correct.</td>
<td></td>
</tr>
<tr>
<td>Getid</td>
<td>• Make sure the returned id is</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>correct.</td>
<td></td>
</tr>
<tr>
<td>Getname</td>
<td>• Make sure the returned name is</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>the correct user’s name.</td>
<td></td>
</tr>
<tr>
<td>Getpassword</td>
<td>• Make sure the returned password</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>is the user’s password.</td>
<td></td>
</tr>
<tr>
<td>Getemail</td>
<td>• Make sure the email address</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>returned is the correct user’s e-mail</td>
<td></td>
</tr>
<tr>
<td></td>
<td>address.</td>
<td></td>
</tr>
</tbody>
</table>

Table 8. Unit Test Results

4.2 Subsystem Testing

Subsystem testing is the next step up in the testing process where all related units from a subsystem do a certain task. Thus, the subsystem test process is useful for detecting interface errors and specific functions. Table 24 show subsystem test results in detail.
### Table 9. Subsystem Test Results

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Tests Performed</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Authorize subsystem</strong></td>
<td>• Test if it can get the error message.</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• Make sure the result of authorizing user is correct.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Verify the login user information is stored in session properly.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Check if the saving user login information function stores the user information in cookies for future login use.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Check if the login page can get the saved user login information saved before from cookies.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Verify the login page redirect to the correct browsing or editing page after the user logs in.</td>
<td></td>
</tr>
<tr>
<td><strong>Accounts management subsystem</strong></td>
<td>• Make sure all the existing users are listed in the user list.</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• Check if the subsystem can detect the error of creating of the user that exists in the subsystem.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Check if the user can update his/her own account properly.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Verify the created user information is the same as the information provided.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Verify the subsystem can delete a user account.</td>
<td></td>
</tr>
<tr>
<td>Subsystem</td>
<td>Tests Performed</td>
<td>Results</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Subsystem Tests Performed</td>
<td>Make sure the password query function can work properly.</td>
<td></td>
</tr>
<tr>
<td>Browsing subsystem</td>
<td>• Check if the subsystem checks for user privilege before showing pages.</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• Verify the page is showing properly after the user click on the page link.</td>
<td></td>
</tr>
<tr>
<td>Editing subsystem</td>
<td>• Make sure the subsystem checks the user privilege before forwarding to edit page.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Verify the subsystem check the user privilege before update the page information.</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• Verify if the subsystem shows the page properties is the users are the owner or the administrator.</td>
<td></td>
</tr>
<tr>
<td>Orphan pages subsystem</td>
<td>• Make sure all the orphan pages are shown on the list.</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>• Verify the owner the page or the administrator can delete the specific orphan page.</td>
<td></td>
</tr>
</tbody>
</table>

4.3 System Testing

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

System testing of begins with the following steps:

Table 10. System Test Results

<table>
<thead>
<tr>
<th>System Testing</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Install the system into server.</td>
<td>Pass</td>
</tr>
<tr>
<td>2. Start up all servers including Tomcat server and PostgreSQL database server.</td>
<td>Pass</td>
</tr>
<tr>
<td>3. Running testing by using real data on all forms and reports.</td>
<td>Pass</td>
</tr>
</tbody>
</table>
CHAPTER FIVE
MAINTENANCE MANUAL

The maintenance manual records any information that can be used to setup the system or backup the system. In order to make sure the system works smoothly and meets the expectation of the users, it is very important to follow the maintenance manual step by step carefully.

5.1 Software Installation

The system requires Linux RedHat as an operating system, PostgreSQL, JSDK, Ant, TOMCAT, and JDBC to run the programs. Following will detail the installation of those 6 softwares.

5.1.1 RedHat Installation

RedHat is a linux base operating system which is offered freely and be downloaded from internet. The reason we choose RedHat is it offers better performance than a Microsoft operating system. Following are the steps to install RedHat onto your machine.

2. Install the operating system by inserting CD 1 into the CD-ROM and start up the machine which is going to install the operating system.

3. The machine will startup via CD-ROM and start to install RedHat.

4. Follow the install wizard and sets up the required information such as network setting and the hardware environment.

5. After all the necessary files are copied into the computer and install it, the machine will restart and Redhat is installed.

5.1.2 PostgreSQL Installation

PostgreSQL is the database system we use in this project; it's free, and is included in RedHat by default. To install PostgreSQL, follow the following steps:

1. Because PostgreSQL may install on to RedHat when the operating system is installed, the first thing we have to do is to check if the PostgreSQL is already in the operating system. Using the command to check if PostgreSQL exist in the operating system:

```bash
rpm -q postgresql
```
If PostgreSQL is not installed in the operating system, then use rpm to install it.

2. In order to create a database user, "wiki," and database, have the following commands executed.

   su postgres
   initdb -D /var/bin/pgsql/data
   createuser wiki
   createdb wiki

   where at the first command, the postgres is the default user for PostgresSQL. Starting the database by using the command "initdb" with the directory "/var/bin/pgsql/data/" which is the default database directory will start up the database system. Login postgres as the supervisor and create a database user, "wiki," and the database "wiki."

3. There are still some steps needed to setup the environment values.

   In the user's environment setup file /etc/profile.d/*.sh, add the following line:

   export PGDATA=/var/lib/pgsql/data
Open the file /var/lib/pgsql/data/postgresql.conf and uncomment the line:

tcpip_socket = true

In order to have the database system startup at the system start, have the command executed:

/sbin/chkconfig --level 3 postgresql on

And, the last step is to startup the database system immediately now without restart the system:

/sbin/service postgresql start

After having the steps above executed, the database system is ready to go and now we have to install JAVA platform, JAVA 2 Platform, Standard Edition (J2SE).

5.1.3 JAVA 2 Platform, Standard Edition (J2SE)

J2SE is the compiler program for JSP and JAVA Servlet programs and it’s required in TOMCAT JAVA Container. Fist of all, we go to http://java.sun.com/j2se/1.4.1/download.html to download SDK Linux (all languages, including English) to the directory /usr/java, then execute the following commands:

    chmod +x j2sdk-1_4_2_01-linux-i586-rpm.bin
    ./j2sdk-1_4_2_01-linux-i586-rpm.bin
rpm -ivh j2sdk-1_4_2_01-linux-i586.rpm

And, set the environment variables in the file /etc/profile.d/* .sh:

JAVA_HOME=/usr/java/j2sdk1.4.2_01
PATH=${PATH}:${JAVA_HOME}/bin

Export JAVA_HOME

5.1.4 Tomcat

TOMCAT is one of the apache jakarta projects, which is a web container to process JSP and JAVA Servlet programs, and to serve static web pages. First of all, we go to the tomcat's official download ftp server at http://ftp.epix.net/apache/jakarta/tomcat-5/v5.0.12-beta/bin/ to download the file of tomcat server for linux jakarta-tomcat-5.0.12.tar.gz to /usr/java/ and extract it to the hard drive.

    tar -xzvf jarkata-tomcat-5.0.12.tar.gz

Also, we modify the file /usr/java/jarkata-tomcat-5.0.12/conf/server.xml by add the following setting in the file:

    <Context
      path="/wiki"
      docBase="/pub/wiki"
      debug="0"

swallowOutput="true" />

<Logger
    className="org.apache.catalina.logger.FileLogger"
    prefix="" 
    suffix=".log"
    directory="/pub/wiki/logs"
    timestamp="true" />

    <Parameter name="contextPath"
        value="/wiki" />

    <Parameter name="homePage" value="HomePage" />

    <Resource name="jdbc/postgres"
        auth="Container"
        type="javax.sql.DataSource" />

    <ResourceParams name="jdbc/postgres">
        <parameter>
            <name>factory</name>
            <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
        </parameter>

    </ResourceParams>
<parameter>
    <name>url</name>
    <value>jdbc:postgresql://127.0.0.1:5432/wiki</value>
</parameter>

<parameter>
    <name>username</name>
    <value>Jimmy</value>
</parameter>

<parameter>
    <name>password</name>
    <value></value>
</parameter>

<parameter>
    <name>maxActive</name>
    <value>10</value>
</parameter>

<parameter>
    <name>maxIdle</name>
    <value>2</value>
</parameter>

<parameter>
    <name>maxWait</name>
</parameter>
And, set the environment variable by adding the following lines in the file /etc/profile.d/*.sh

```
CATALINA_HOME=/usr/java/jarkata-tomcat-4.1.27
PATH=${PATH}:${JAVA_HOME}/bin:${CATALINA_HOME}/bin
export CATALINA_HOME
```

Add the following lines in the file /etc/rc.local to have the tomcat run when the system boots:

```
Export JAVA_HOME=/usr/java/j2sdk1.4.2_01
export CATALINA_HOME=/usr/java/jarkata-tomcat-4.1.27
${CATALINA_HOME}/bin/startup.sh
```

5.1.5 JAVA Database Connectivity (JDBC)

The API used to execute SQL statement is different for each database engine. Java programmers, however, are lucky and are freed from such database portability issues. They have a single API, the Java Database Connectivity API (JDBC), that's portable between database engines. The JDBC library provides an interface for executing SQL statements. It provides the basic functionality for data access. A number of drivers are
available for PostgreSQL, and information about this can be obtained at the PostgreSQL homepage at http://jdbc.postgresql.org/download/. Download pg73jdbc3.jar and copy the file to /usr/java/jakarta-tomcat-4.1.18/common/lib/.

5.2 Variables Modification

We have to change some environment variables in the linux system and server.xml in Tomcat server configuration directory.

5.2.1 System Variables

1. Open the file "server.xml" in the directory "/usr/java/jakarta-tomcat-4.1.18/conf" via "vi" or any other editor.

2. Scroll down until you see the context area we added in at chapter 7.4.1.

3. The variable "path" in Context indicates the context path of the web application. The default value would be "/wiki."

4. The variable "docBase" in Context is the files directory for the web application. The default value would be "/pub/wiki."
5. The variable "variable" in Logger is the absolute or relative pathname of a directory in which log files created by this logger will be placed. The default value would be "/pub/wiki/logs."

6. Now, let's look down at the parameter setting.

7. The parameter "contextPath" indicate the context path for the system which would be the same as the value of path.

8. The parameter "homepage" sets the name of the home page of wiki system. The default value would be "HomePage."

9. The parameter "username" is the user name who can access the database system. Usually, this value would be the administrator.

10. The parameter "password" is the password corresponding to the user name at 9. If there is no special setting in database system, leave the value to be empty.
5.3 Installation/Migration

1. All the JSP programs and HTML programs are stored in
   \pub\wiki

2. All the classes are stored in
   \pub\wiki\WEB-INF\classes

3. Place the web.xml for in
   \pub\wiki

5.4 Backup and Restore

Backup is a very important action needed for any system to prevent losing data. No one can say a system works very well and will never have a problem. There are two steps to back up WOACC. One is to backup the system files. The other step is to backup the database which is used by the system.

5.4.1 System Backup

All the files are located in the directory "\pub/wiki" and all its subdirectory. Thus, in order to backup the system files, all we need to do is to backup the files in the directory. The method here I suggest is to compress the directory of "\pub/wiki" including its subdirectory to compress files for future use by the
compress program "tar." Using the following command to backup the system files:

```
tar -cf wiki.tar /pub/wiki
```

### 5.4.2 Database Backup

To backup the database system, we use `pg_dump` command. The following command is used to backup the database:

```
pg_dump wiki | gzip wiki.zip
```

After executing the backup command above, the file Wiki.zip would be the backup file of the database.

### 5.4.3 System Restore

To restore the system file, simply extract the backup file by using the following command:

```
tar -xzvf wiki.tar /
```

By the command above, all the files will restore into the directory /pub/wiki and complete the restore system process.

### 5.4.4 Database Restore

To restore the database needed for the system, go to the directory where your database backup file is in, and execute the following commands:

```
createdb wiki

gunzip -c wiki.zip | psql wiki
```
After the commands are executed, the database is restored to the database system. Then, restart tomcat, the wiki will be completely restored.
CHAPTER SIX

CONCLUSION AND FUTURE DIRECTIONS

6.1 Conclusion

This project shows a successful implementation of extending the wiki paradigm to serve a whole department of instructors involved in teaching information technology. The system has been designed and implemented with highly effective contemporary tools and technologies, which makes it flexible for more enhancements and improvements. The system is capable of handling sets of courses offered by set of instructors, where students can be enrolled in more than one class. Moreover, the wiki part works with XHTML validation, which provides an excellent learning environment for students.

This system will run as a communication interface between students and instructors. Using this interface, pages are created and modified quickly and easily. An instructor will be able to publish course materials, post announcements, presentations timetable, etc into the web. If the student is enrolled to the course, she will be able to edit these pages and can add for example her
presentation title to the presentations timetable, or her paper abstract to the term paper page. Moreover, all the pages can be used as a public or private repository, so that students can post her homework and then limit access to instructor.

This project consists of various state-of-the-art java web technologies. The project has provided me with the experience in server side programming and persistent object frameworks, designing extensible web applications, and debugging real world applications.
REFERENCES


54