California State University, San Bernardino
CSUSB ScholarWorks

Theses Digitization Project | John M. Pfau Library

2010

The college of extended learning online registration system

Ching-yi Wang

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

Part of the Graphics and Human Computer Interfaces Commons

Recommended Citation

Wang, Ching-yi, "The college of extended learning online registration system" (2010). *Theses Digitization Project*. 3724.
https://scholarworks.lib.csusb.edu/etd-project/3724

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.

THE COLLEGE OF EXTENDED LEARNING ONLINE REGISTRATION SYSTEM

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

Ching-yi Wang

March 2010

THE COLLEGE OF EXTENDED LEARNING ONLINE REGISTRATION SYSTEM

A Project

Presented to the

Faculty of

California State University,

San Bernardino

by

Ching-yi Wang

March 2010

Approved by:

[redacted]

David Turner, Chair, Department of Computer Science and Engineering

3/11/2010
Date

[redacted]

Kerstin Voigt

[redacted]

Arturo I. Concepcion

© 2010 Ching-yi Wang

ABSTRACT

The College of Extended Learning Online Registration System (CELORS) is a client-server Web application used by the staff of the College of Extended Learning (CEL) and students from campus and off-campus. The CELORS provides the staff of the College of Extended Learning a user-friendly graphic interface to manage courses and process student registrations and payments. The CELORS also provides students a secure and convenient platform, 24 hours per day, seven days per week, to view and register for courses offered by the College of Extended Learning.

The CELORS was newly written in Java and integrated with the Spring and Hibernate frameworks, which helped to reduce the lines of code needed to implement desired functionality, and provided a means to develop the user interface following a small set of well-defined patterns. The CELORS solves orphan records by restructuring the database. The CEL staff has more control in changing the parking fees, Osher membership fees, and other quarterly-based settings. The CELORS also provides the college a secure, PCI-compliant online payment system by integrating with Paypal.

ACKNOWLEDGEMENTS

I would like to thank all the people with whom I have worked while pursuing my master's degree at California State University, San Bernardino (CSUSB). I wish I could list all their names but the list would be too long and I would still probably leave some people out. Studying in the Computer Science and Engineering department at CSUSB has been a tremendous learning experience, both personally and professionally.

Thank you to the following faculty of the Computer Science and Engineering department for their invaluable guidance, advice, support, help, and patience during this project's long gestation: Dr. David Turner, Dr. Arturo Concepcion and Dr. Kerstin Voigt.

Special thanks to Christine Plattner in the College of Extended Learning. Without her valuable suggestions and support, the Online Registration System wouldn't be a solid system and this master's project document wouldn't be published without her proofreading it.

I would also like to thank my friends and family members who have patiently helped me during these past few years. Finally, thanks to my parents who encouraged me to explore my interest in the computer field.

TABLE OF CONTENTS

Abstract iii

Acknowledgements iv

List of Tables ix

List of Figures xi

1. Introduction 1

1.1 Background 1

1.2 Purpose 1

1.3 Project Scope 2

1.4 Project Limitations 3

1.5 Definitions, Acronyms, and Abbreviations 3

2. Architecture 8

2.1 Project Design 8

2.2 The Model-view-controller Architecture 8

2.3 Database Mapping 9

2.4 ORS Controllers 10

2.4.1 OrsParameterizableViewController Class 11

2.4.2 OrsAbstractWizardFormController Class 11

2.4.3 OrsSimpleFormController Class 11

2.5 Product Perspective 11

2.5.1 System Interfaces 11
2.5.2 User Interfaces 13
2.5.3 Software Interfaces 13
2.5.4 Communication Interface 13
2.5.5 Memory Constraints 13
2.5.6 Operations 13
2.5.7 Site Adaptation Requirements 14
2.5.8 Product Function 14
2.5.9 User Characteristics 14
2.5.10 Constraints 18
2.5.11 Assumptions and Dependencies 19
2.5.12 Apportioning of Requirements 19

3. Database Design 20
3.1 Data Analysis 22
3.2 Database Specification 23
3.2.1 Database Schema Logical Model - Relational Schema 23
3.2.2 Database Design 25
3.2.3 Admin Table 27
3.2.4 Category Table 28
3.2.5 Coupon Table 29
3.2.6 Course Table 30
3.2.7 Course_category Table 32
3.2.8 Course_discount Table 33
3.2.9 Course_multiple_discount Table 34
3.2.10 Course_registration Table 35
3.2.11 Discount Table 36
3.2.12 Discount_item Table 37

3.2.13 Header Table 39
3.2.14 Header_category Table 39
3.2.15 Marketing Table 40
3.2.16 Multiple Discount Table 42
3.2.17 Quarter Table 44
3.2.18 Receipt Table 46
3.2.19 Registration Table 48
3.2.20 Staff Table 51

4. Project Implementation 54
4.1 User Interface Design 54
4.1.1 Administrator Pages 54
4.1.2 Staff Pages 60
4.1.3 Student Pages 97

5. Software Validation 109
5.1 Unit Testing 109
5.2 System Testing 111
5.3 System Integration Testing 112

6. Maintenance Manual 113
6.1 Operation System Installation 113
6.1.1 Downloading and Installing CentOS 113
6.1.2 Disk Partition Setup 114
6.1.3 Network Configuration 114
6.1.4 Others 115
6.1.5 Post Installation 115
6.1.6 Java Installation 118
6.1.7 Tomcat Installation 119

6.1.8 MySQL Installation and Configuration 126
6.1.9 JDBC Connector Installation 127
6.2 CELORS Installation 128
6.2.1 Database Installation 128
6.2.2 Software Installation 131
6.3 System Backup and Restore 131
6.3.1 Database Backup 131
6.3.2 Database Restore 132

7. Conclusion and Future Direction 133
7.1 Conclusion 133
7.2 Future Direction 134

References 135

LIST OF TABLES

3.1 Staff Table Metadata 21

3.2 Database Relational Schema 23

3.3 Database Relational Schema - Cont. 24

3.4 Database Relational Schema - Cont. 25

3.5 hibernate.cfg.xml File 26

3.6 Structure of Admin Table 27

3.7 Structure of Category Table 28

3.8 Structure of Coupon Table 29

3.9 Structure of Course Table 31

3.10 Structure of Course Table - Cont. 32

3.11 Structure of Course_Category Table 33

3.12 Structure of Course_Discount Table 34

3.13 Structure of Course_Multiple_Discount 35

3.14 Structure of Course_Registration 36

3.15 Structure of Discount Table 37

3.16 Structure of Discount_Item Table 38

3.17 Structure of Header Table 39

3.18 Structure of Header_Category Table 40

3.19 Structure of Marketing Table 41

3.20 Structure of Multiple_Discount Table 43

3.21 Structure of Quarter Table 45

3.22 Structure of Receipt Table 47

3.23 Structure of Registration Table 49

3.24 Structure of Registration Table - Cont. 50

3.25 Structure of Staff Table 52

5.1 Unit Testing Results 110

5.2 System Testing Results 111

5.3 System Integration Testing 112

6.1 Disk Partition Specification 114

6.2 Network Configuration 115

LIST OF FIGURES

2.1 Page Request Processing In The MVC Architecture 9

2.2 Deployment Diagram 12

2.3 Use Case Diagram 14

2.4 Student Use Case Diagram 15

2.5 Admin Use Case Diagram 16

2.6 Catalog Manager Use Case Diagram 17

2.7 Registration Manager Use Case Diagram 18

3.1 Entity Relational Diagram 22

3.2 Entity Relational Diagram - Admin Table 27

3.3 Entity Relational Diagram - Category Table 29

3.4 Entity Relational Diagram - Coupon Table 30

3.5 Entity Relational Diagram - Course Table 32

3.6 Entity Relational Diagram - Course_Category Table 33

3.7 Entity Relational Diagram - Course_Discount Table 34

3.8 Entity Relational Diagram - Course_Multiple_Discount Table 35

3.9 Entity Relational Diagram - Course_Registration Table 36

3.10 Entity Relational Diagram - Discount Table 37

3.11 Entity Relational Diagram - Discount_Item Table 38

3.12 Entity Relational Diagram - Header Table 39

3.13 Entity Relational Diagram - Header_Category Table 40

3.14 Entity Relational Diagram - Marketing Table 42

3.15 Entity Relational Diagram - Multiple_Discount Table 43
3.16 Entity Relational Diagram - Quarter Table 46
3.17 Entity Relational Diagram - Receipt Table 48
3.18 Entity Relational Diagram - Registration Table 51
3.19 Entity Relational Diagram - Staff Table 53

4.1 Administrator Login Page 55
4.2 List Staff Page 56
4.3 Create Staff Page 57
4.4 Edit Staff Page 58
4.5 Delete Staff Page 59
4.6 Change Password Page - Administrator 59
4.7 Staff Login Page 60
4.8 List Quarters Page 61
4.9 Clone Quarter Page 62
4.10 Edit Quarter Page 63
4.11 View Category List Page 64
4.12 Create Category Page 65
4.13 Edit Category Page 66
4.14 Delete Category Page 67
4.15 View Course List Page 68
4.16 Create Course Page 69
4.17 Create Course Page - Cont. 70
4.18 Edit Course Page 71
4.19 Edit Course Page - Cont. 72
4.20 Delete Course Page 73
4.21 View Header List Page 74
4.22 Create Header Page 75

4.23 Edit Header Page 76
4.24 Delete Header Page 77
4.25 View Discount List Page 78
4.26 Create Discount Page 79
4.27 Edit Discount Page 80
4.28 Delete Discount Page 81
4.29 View Coupon List Page 82
4.30 Create Coupon Page 83
4.31 Edit Coupon Page 84
4.32 Delete Coupon Page 85
4.33 View Multiple Discount List Page 86
4.34 Create Multiple Discount Page 87
4.35 Edit Multiple Discount Page 88
4.36 Delete Multiple Discount Page 89
4.37 Search Registration Records Page 90
4.38 List Registration Records Page 91
4.39 View Registration Record Page 92
4.40 Edit Registration Record Page 93
4.41 Delete Registration Record Page 94
4.42 Print Registration Record Page 95
4.43 Change Password Page - Staff 96
4.44 Remote Login Page 97
4.45 Index Page Without Specified Quarter 98
4.46 View Bulletin Page 99
4.47 View Course List Page 100
4.48 View Course Page 101
4.49 Register By Schedule Number Page 102

4.50 View Cart Page 102
4.51 Registration Information Page 103
4.52 Registration Discount/Parking Pass Page 104
4.53 Registration Marketing Page 105
4.54 Registration Review Page 106
4.55 Registration Payment Page 107
4.56 Registration Confirmation Page 108

6.1 Download The Latest Tomcat Package 120

1. INTRODUCTION

This chapter gives a brief introduction of the project background.

1.1 Background

The College of Extended Learning (CEL) helps learners achieve their education goals by providing access to University degrees, certificate programs, professional development, and personal enrichment programs. As the Internet became a more useful tool in our daily lives, today's learners expect not only high-quality classes, but fast, student-friendly service. In February 2005, the first generation of the College of Extended Learning online registration system (CELORS) was launched. The CELORS is accessible 24 hours per day, seven days per week through a secure mechanism. As time went by, a lot of features were added or modified based on CEL's requirements. That made the CELORS more and more complicated and difficult to maintain from the developer and staff points of view. Therefore, a staff- and developer-friendly online registration system was needed to make the daily work easier and more efficient.

1.2 Purpose

This system was initially built so that the College of Extended Learning could offer its students an online method to register for classes, in addition to the traditional methods of registering by phone, by fax, by mail, or in person. Over the last few years, the system has been routinely modified to meet the needs of the College to suit their

ever-changing business rules. Also, as Internet and computer usage have increased, students have come to expect course information at their fingertips and the ability to register immediately for classes. The inefficient performance of the CELORS and its associated database, as well as the inflexibility of managing multiple course catalogs became issues. Updating the previous version of the CELORS became more difficult and time-consuming each time a CEL business rule changed or a new business rule was implemented. Therefore, this project involved re-structuring, redesigning and rewriting the old ORS based on CELs desired functionalities and changing business rules. In this new version of the ORS, all users' registration-related information is now stored in a MySQL database and, since the CELORS is a 24/7 application, the system was made flexible and maintainable so that, as business rules (such as parking fees) are revised, the CEL staff now have the ability to make the necessary changes.

1.3 Project Scope

Students can view multiple course catalogs online, read course descriptions, and select and register for courses through the CELORS. The CEL staff can maintain quarterly course catalogs as well as process student registrations through the CELORS.

There are two intended users of the CELORS: CEL staff and CEL students. The CELORS has the following functionalities for its users:

- Registration
- Manage Staff
- Manage Courses
- Manage Registrations

The CEL staff has two available privileges: Manage Courses and/or Manage Registrations. Privileges are assigned to staff members depending upon their job functions.

For the Managing Courses privilege, staff can manage course catalog information by quarter: input course descriptions, apply discounts, adjust fees, etc. For the Manage Registrations privilege, staff can manage registration records and payments: download and approve or disapprove registrations.

For CEL students, the CELORS provides 24/7 catalog access, up-to-date course descriptions, schedules and fees, and a secure, PCI-compliant platform for submitting course registrations and payments.

1.4 Project Limitations

- Due to the implementation of the PCI DSS at CSUSB, payment information cannot be stored in any format.
- The CELORS is an official Web site that represents California State University, San Bernardino. Therefore, it must comply with the CSUSB Web page accessibility standards and guidelines.

1.5 Definitions, Acronyms, and Abbreviations

The definitions, acronyms, and abbreviations used in the document are described in this section.

- API: Application Program Interface is a set of routines that an application uses to request and carry out low-level services performed by a computer's operating system; also, a set of calling conventions in programming that defines how a service is invoked through the application.
- CEL: The College of Extended Learning at California State University, San Bernardino.
- CELORS: The College of Extended Learning Online Registration System.

- CentOS: A freely available operating system that is based on Red Hat Enterprise Linux.
- CSUSB: California State University, San Bernardino.
- DHCP: Dynamic Host Configuration Protocol is a computer networking protocol used by devices (DHCP clients) which dynamically distributes the IP address to the destination host.
- DNS: The Domain Name System is a hierarchical naming system for computers, services, or any resource connected to the Internet or a private network.
- Hibernate: An object-relational mapping (ORM) library for the Java language which provides a framework for mapping an object-oriented domain model to a traditional relational database.
- HTML: HyperText Markup Language is the authoring language used to create documents on the World Wide Web.
- HTTPS: Hypertext Transfer Protocol Secure is a combination of the Hypertext Transfer Protocol and a network security protocol. This security protocol operates at a lower sublayer, encrypting an HTTP message prior to transmission and decrypting a message upon arrival.
- IEP: International Extension Programs is a unit of the College of Extended Learning.
- J2EE: Java 2 Platform Enterprise Edition is a platform-independent, Java-centric environment from Sun Microsystems, Inc., for developing, building, and deploying Web-based enterprise applications online. The J2EE platform consists of a set of services, APIs, and protocols that provide the functionality for developing multitiered, Web-based applications.

- Java: Java is a object-oriented, cross-platform programming language from Sun Microsystems.
- Java Servlet: Java Servlet technology provides Web developers with a simple, consistent mechanism for extending the functionality of a Web server and for accessing existing business systems.
- JDBC: Java Database Connectivity is an API for the Java programming language that defines how a client may access a database. It provides methods for querying and updating data in a database. JDBC is oriented towards relational databases.
- Join Table (also known as a "Link Table" or "Junction Table"): A Join Table is table that contains common fields from two tables. It is on the many side of a one-to-many relationship with the other table. It is employed when dealing with many-to-many relationships in a database.
- JSP: Java Servlet Page is a server-side technology. JSPs have dynamic scripting capabilities that work in tandem with HTML code, separating the page logic from the static elements to help make the HTML more functional (i.e., dynamic database queries).
- MVC: Model-View-Controller is an architectural pattern used in software engineering to isolate business logic from user interface considerations.
- Osher: Osher Lifelong Learning Institute is a unit of the College of Extended Learning.
- MySQL: MySQL is a relational database management system which runs as a server providing multi-user access to a number of databases.
- Paypal: Paypal is an e-commerce business allowing payments and money transfers to be made through the Internet. Paypal serves as an electronic alternative to traditional paper payment methods such as checks and money orders.

- PCI: Payment Card Industry.
- PCI DSS: Payment Card Industry Data Security Standard. The standard was created to help organizations that process card payments to prevent credit card fraud through increased controls around data and its exposure to compromise.
- Section 508: In 1998, the U.S. Congress amended the Rehabilitation Act to require Federal agencies to make their electronic and information technology accessible to people with disabilities. Section 508 was enacted to eliminate barriers in information technology, to make available new opportunities for people with disabilities, and to encourage development of technologies that will help achieve those goals.
- Session: A session is a collection of related clients which can exchange data via defined communication paths. The session maintains the state-associated communication paths and may interact with an object which encapsulates a defined session-management policy.
- Socket: An endpoint for communication between two machines.
- Spring Framework: (Spring, for short) is an open-source application framework for the Java platform.
- TCP/IP: Transmission Control Protocol on top of the Internet Protocol provides a reliable, point-to-point communication channel that client-server applications on the Internet use to communicate with each other. To communicate over TCP, a client program and a server program establish a connection to one another. Each program binds a socket to its end of the connection. To communicate, the client and the server each read from and write to a socket bound to the connection.
- Tomcat: Apache Tomcat is a servlet container. Tomcat implements the Java

Servlet and the Java Server Pages (JSP) specifications and provides a "pure Java" HTTP Web server environment for Java code to run.

- UML: The Unified Modeling Language is the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

2. ARCHITECTURE

2.1 Project Design

The College of Extended Learning Online Registration System (CELORS) implements a client-server Web-based system. The front-end pages are written in JSP. On the server side, the whole project is written in Java. In addition, this project uses Spring Framework to manage the life cycle and configuration of application objects, and Hibernate Framework to handle database mapping and accessing. Apache Tomcat is used as the Web server.

2.2 The Model-view-controller Architecture

The model-view-controller (MVC) is a pattern for the architecture of a software application. It separates an application into the following components:

- Models, for handling data and business logic
- Controllers, for handling the user interface and application logic
- Views, for handling graphical user interface objects and presentation logic

This separation results in user requests being processed as follows:

1. The browser, on the client, sends a request for a page to the controller on the server.
2. The controller retrieves the data it needs from the model in order to respond to the request.

3. The controller renders the page and sends it to the view.

4. The view sends the page back to the client for the browser to display.

Figure 2.1 illustrates the MVC process in the CELORS.

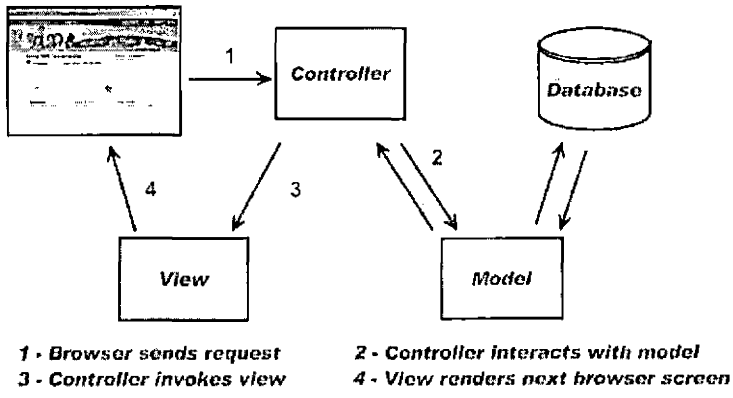


Fig. 2.1: Page Request Processing In The MVC Architecture

2.3 Database Mapping

The CELORS project uses database mapping to represent the following relationships between objects: one-to-one, many-to-many, one-to-many, and many-to-one. For each of the tables that are involved in these mappings, the field in the corresponding .java source file has been set, as well as the setter and getter methods for that field. Here is the list of tables for each mapping type.

For one-to-one mapping:

- Registration to marketing and receipt
- Quarter to registration

For many-to-many mapping:

- Header to category
- Category to course
- Course to multiple discount

For one-to-many mapping:

- Quarter to course, discount, multiple discount, coupon, category and header
- Registration to course

For many-to-one mapping:

- Course to quarter
- Category to quarter
- Header to quarter
- Course to registration

2.4 ORS Controllers

The Spring Framework 1.5 provides a base controller interface, representing a component that receives HttpServletRequest and HttpServletResponse like an HttpServlet but is able to participate in an MVC workflow 1.5. It also provides several abstract controllers that contain functionalities for typical use cases in Web applications. Hence, the CELORS implements its customized controllers by extending abstract controller class instead of directly writing the controller interfaces. They are "ParameterizableViewController", "AbstractWizardFormController", and "SimpleFormController".

2.4.1 *OrsParameterizableViewController Class*

This class extends the ParameterizableViewController. It offers an alternative to sending a request straight to a view such as a JSP. This is mainly used for the view or list function.

2.4.2 *OrsAbstractWizardFormController Class*

As its name states, this controller provides wizard-like workflows. This controller has more than one form view page. Therefore, there are various actions instead of one single submit action. This controller is only used in the student checkout procedure. If a student does not finish the entire process, the registration will not be stored and the payment will not be charged.

2.4.3 *OrsSimpleFormController Class*

This is the most widely used controller in this system. It provides configurable form and success views, and an onSubmit chain for convenient overriding. It automatically submits to the form view in case of validation errors, and renders the success view in case of a valid submission.

2.5 *Product Perspective*

The CELORS is currently being used by the College of Extended Learning to manage their course catalogs and registration submissions. A performance improvement along with new functionalities, made this system more user-friendly, productive, and stable.

2.5.1 *System Interfaces*

The CELORS consists of three components as shown in Figure 2.2. These are the client machines, the Web/database server and the Paypal payment server. When a

user accesses the system, the client machine connects to the Web server through the HTTPS protocol. The Web server then pulls out catalog information or verifies login information by connecting to the database. When a student registers for course(s) through the CELORS, the Web server transmits payment information to the Paypal payment server. When staff verifies a registration and payment, staff can login to the CELORS to the finish registration process, such as approving or rejecting registrations and printing out registration records as references.

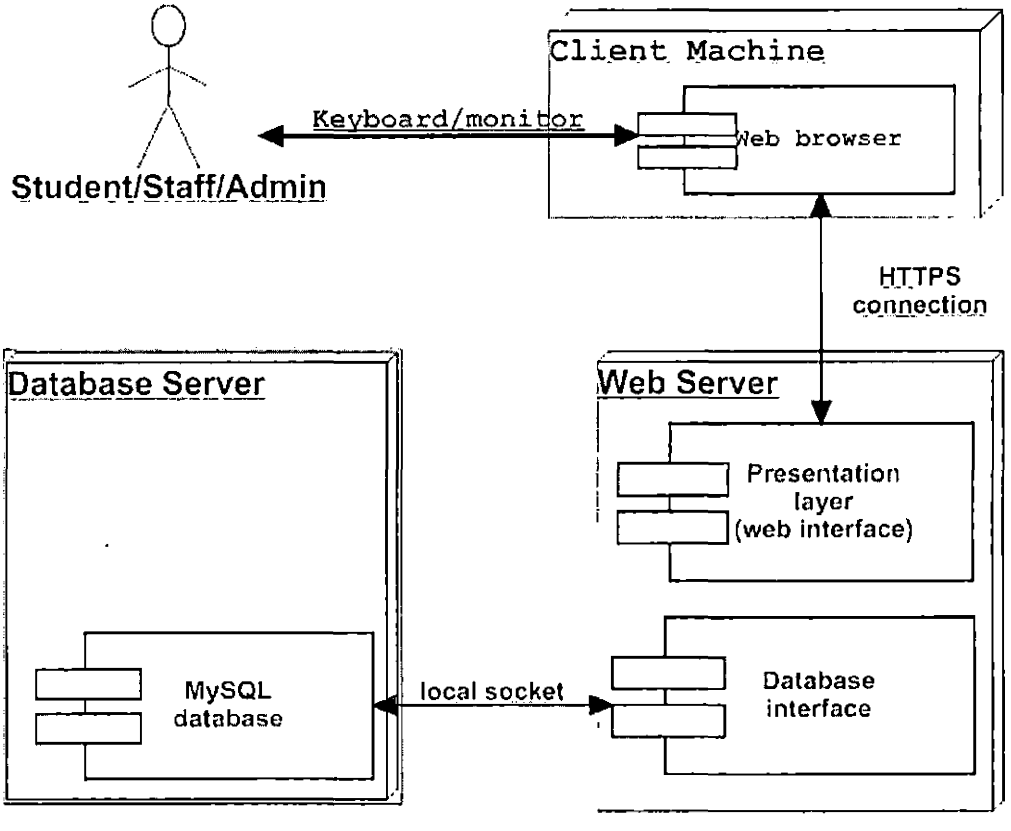


Fig. 2.2: Deployment Diagram

2.5.2 User Interfaces

The CELORS categorizes users into three major types: student, staff, or administrator. All user interfaces, except student, include a login interface page. Staff own different roles: managing catalogs and/or processing payments. The Staff main page shows different functionalities based on his/her role.

2.5.3 Software Interfaces

The project software interface is viewed over the Web. It can run on any Web browser for Windows, Linux, or Mac OS. The language and applications used in writing this project are Java Servlet Page (JSP), Java, and CSS. The operating system running on the server is CentOS 5.2.

2.5.4 Communication Interface

The CELORS uses Spring framework and the operating system to manage communication between the client and the server. The Hibernate framework handles the communication between the MySQL database and Java through JDBC.

2.5.5 Memory Constraints

The minimum memory requirement for running the CELORS server is 512 MB. For the client machine, at least 256 MB is desired.

2.5.6 Operations

The CELORS operates 24/7. Database backup can be done every night locally and remotely through cron job. Maintenance is done on call, and mostly done remotely.

2.5.7 Site Adaptation Requirements

The CELORS does not have any specific site adaptation requirements but at least 1024 x 768 screen resolution is encouraged.

2.5.8 Product Function

The typical users of the CELORS are shown in Figure 2.3: system administrator, student, the College of Extended Learning staff (catalog manager and registration manager). This figure also contains the actions that the project is expected to perform for the users.

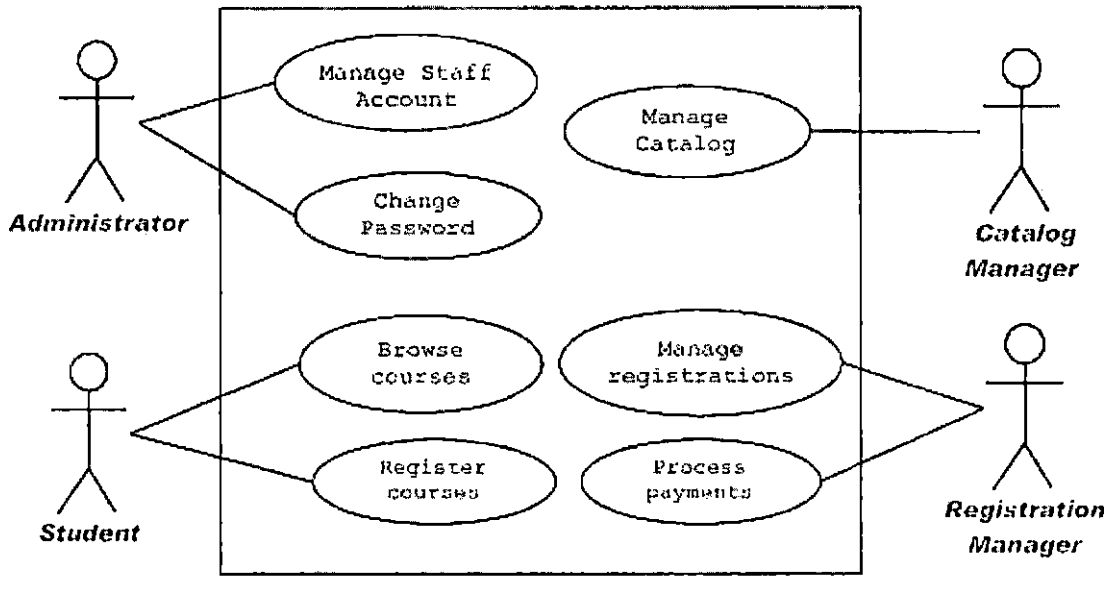


Fig. 2.3: Use Case Diagram

2.5.9 User Characteristics

The CELORS system facilitates the processing of online course registrations for students and the managing of course catalogs and registrations for staff through different

roles. These roles include staff account administration, catalog management, and registration and payment management.

A student (shown in Figure 2.4) adds courses to the shopping cart that he/she is interested in taking and follows the steps to fill in the registration information and pay fees through a secured Paypal payment system. Once registered successfully, he/she will receive a confirmation E-mail.

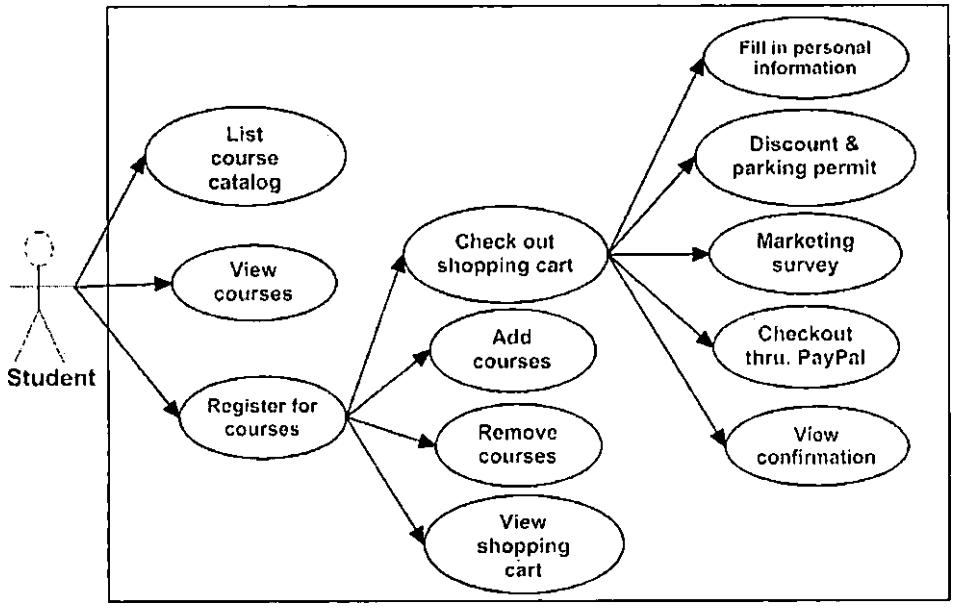


Fig. 2.4: Student Use Case Diagram

The staff account administrator (shown in Figure 2.5) is responsible for managing CEL staff members through various operations. These operations include viewing a list of CEL staff members, viewing the detailed information of a specific staff member, creating a new account, editing information or designated roles for a specific member, and deleting a staff member account from the CELORS.

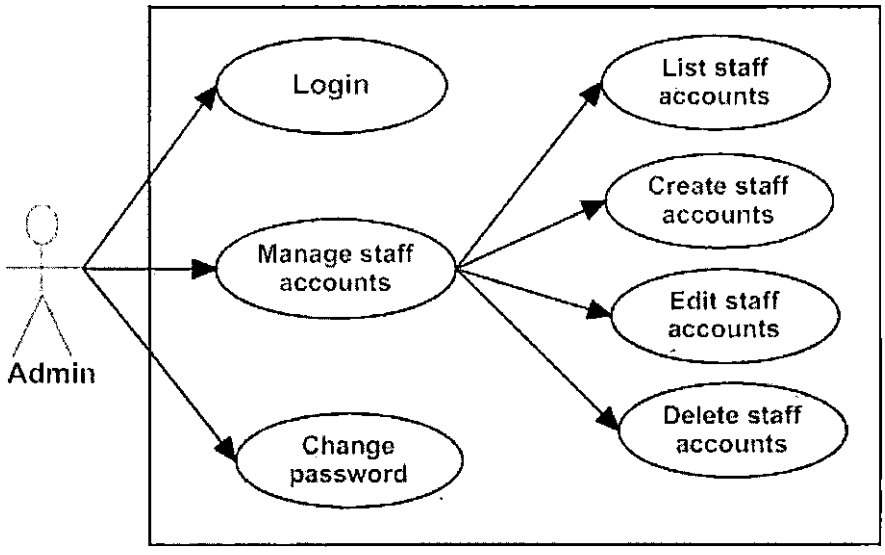


Fig. 2.5: Admin Use Case Diagram

The catalog manager maintains quarterly course information, parking and membership fees, discounts, etc. As shown in Figure 2.6, the manager can view, edit, create, or edit a specific category.

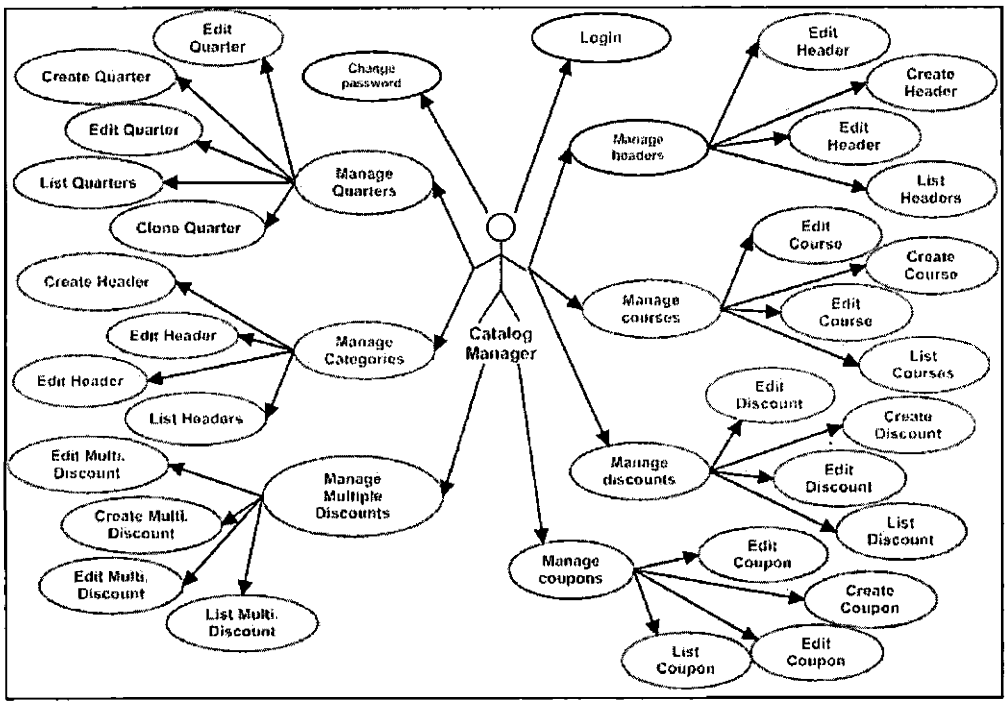


Fig. 2.6: Catalog Manager Use Case Diagram

The registration and payment manager (shown in Figure 2.7) manages registration records and the post-payment process. The registration and payment manager can view, delete, print, or edit a specific registration. This manager also can approve or deny a specific unprocessed registration, or disapprove an approved payment. If a registration is approved by the manager, a payment confirmation E-mail is automatically sent to the student by the CELORS.

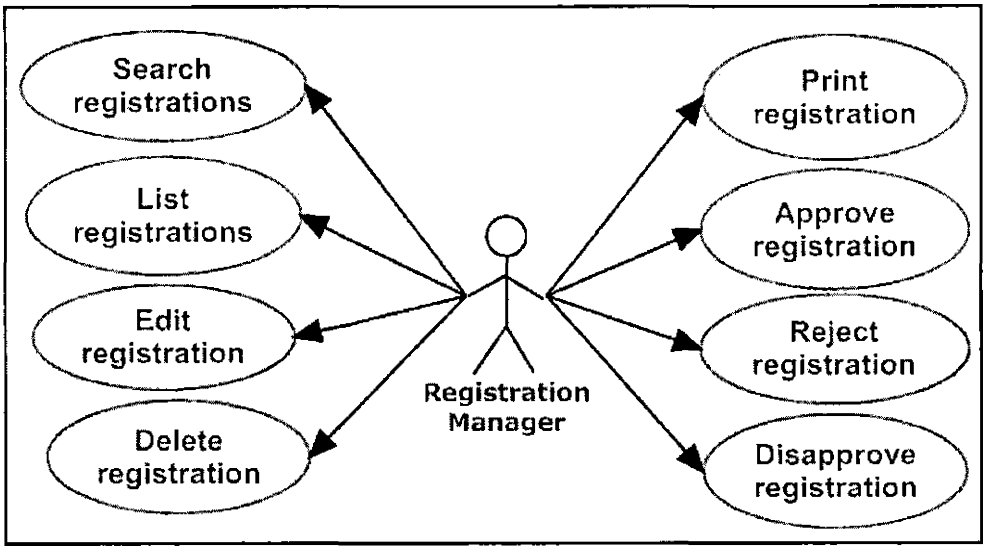


Fig. 2.7: Registration Manager Use Case Diagram

2.5.10 Constraints

In 2009, CSUSB adapted PCI DSS for enhancing payment account data security. There were two major issues in the current CELORS that needed to be resolved.

First, a CSUSB-signed certificate needed to be used instead of a self-signed certificate. The CELORS requested a "CSU San Bernardino Root Certificate" through the campus Information Security Office. The CELORS uses this CSUSB root certificate to generate the CELORS's own certificate used in the https secure CELORS Web site.

Second, payments needed to be processed through a third-party Web site. Previously, the CELORS stored encrypted payment information temporarily (usually no longer than 72 hours) in a database. The payment information was removed permanently from the database once the payment was processed. In this version, the CELORS sends the payment information to the Paypal payment server for processing payment. There is no more payment information stored in the CELORS in any

format.

The following constraints are set for users based on CEL's business rules. For Osher courses, one membership is good for up to a certain number of courses, based on quarterly settings made by the catalog manager. Only the catalogs set to "viewable" can be viewed by students.

2.5.11 Assumptions and Dependencies

The project is built using Spring and Hibernate frameworks. The dependencies of this project are the dependencies of the frameworks that we used.

2.5.12 Apportioning of Requirements

There are no apportioning of requirements.

3. DATABASE DESIGN

The CELORS implements an object/relational mapping (ORM) framework through Hibernate. Hibernate provides the bridge between the database, which is MySQL, and the Java application by storing application objects in the database, rather than writing and maintaining an abundance of code to store and retrieve objects. In short, object/relational mapping (ORM) is the automated persistence of objects to the tables in a relational database. Hibernate uses required metadata to describe the mapping between the objects and the database. Table 3.1 is a mapping metadata for the staff table.

Tab. 3.1: Staff Table Metadata

```
<?xml version='1.0'?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
  <class name="cel.bus.Staff" table="staff" lazy="true">
     <cache usage="read-write" />
     <id name="id" column="id" type="long">
        <generator class="increment"/>
     </id>
     <property name="username" type="string" unique="true"/>
     <property name="passwordDigest" type="string" column="password" />
     <property name="firstName" type="string"/>
     <property name="lastName" type="string"/>
     <property name="email" type="string" />
     <property name="regNotify" type="boolean"/>
     <property name="specialRegNotify" type="boolean" />
     <property name="manageCoursePermission"
             column="manage_course_permission" type="boolean"/>
     <property name="processStudentPermission"
             column="process_student_permission" type="boolean"/>
     <property name="manageReportPermission"
             column="manage_report_permission" type="boolean"/>
  </class>
</hibernate-mapping>
```

3.1 Data Analysis

To effectively store all necessary data, a database with 18 tables was reconstructed. Six of them are join tables (also called association tables) and five tables were dropped from the previous version because they were unused or CEL's business rules had changed. All tables used in the CELORS project store plain text data. The Entity Relational (ER) diagram for the CELORS system is shown in Figure 3.1.

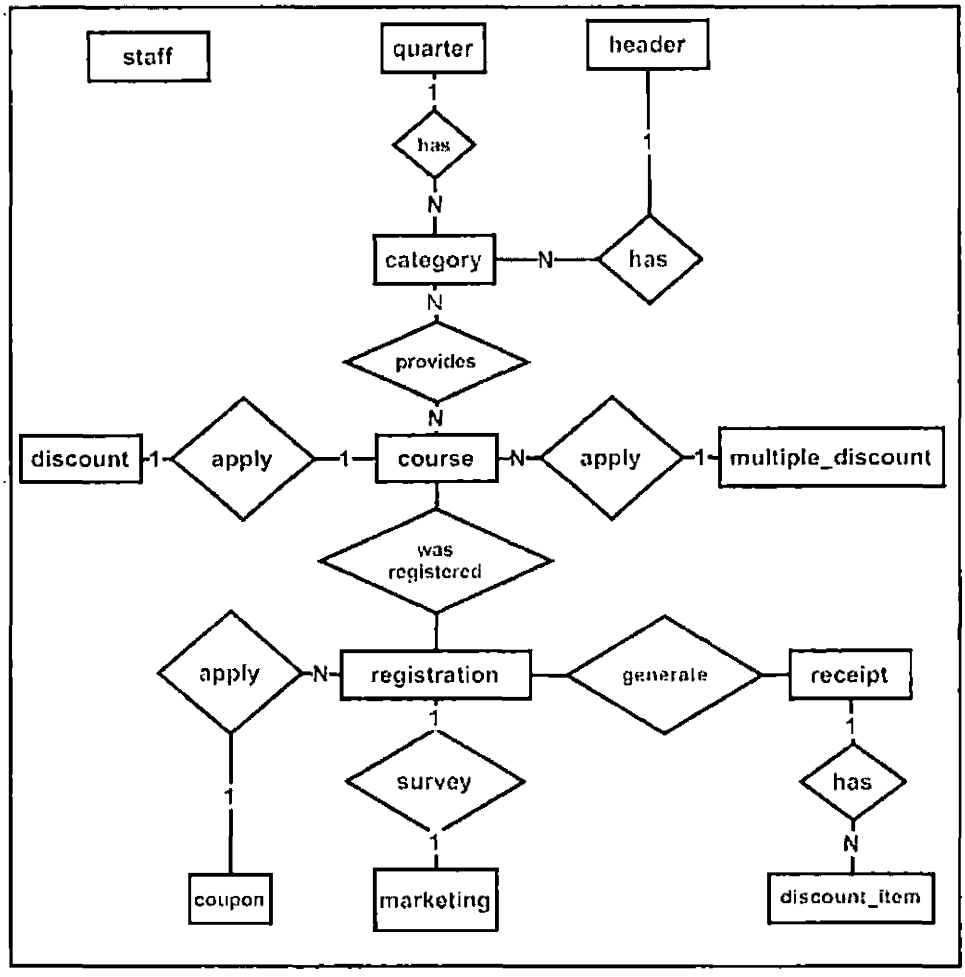


Fig. 3.1: Entity Relational Diagram

The Key field indicates whether the column is indexed. A value of PRI indicates that the column is part of the table's primary key. UNI indicates that the column is part of a UNIQUE index. The MUL value indicates that multiple occurrences of a given value are allowed within the column.

3.2.1 Database Schema Logical Model - Relational Schema

The database relational schema is shown below in Table 3.2. The primary key for each table is underlined.

Tab. 3.2: Database Relational Schema

Admin

id	username	password	

Category

id	name	bulltin_id	visible
view_order			

Category_course

id	category_id	course_id	

Coupon

id	name	discount	quarter_id

Course

id	creditSchedule	nonCreditSchedule	schedule2
schedule3	name	discription	number
units	creditFee	nonCreditFee	fee2
fee3	desc0	desc1	desc2
desc3	daysDates	classTimes	location
regDeadline	instructor	quarter_id	seatSize
seatsTaken	view	orsRegDeadline	

Tab. 3.3: Database Relational Schema - Cont.

Course_discount

id	course_id	discount_id	

Course_multiple_discount

id	course_id	multiple_discount_id	

Course_registration

id	course_id	registration_id	feeType

Discount

id	name	discount	quarter_id

Discount_item

id	name	discount	value
courseNumber	receipt_id		

Header

id	name	view	

Header_category

id	header_id	category_id	

Marketing

id	classLevel	degree	bulletin
flyer	newspaper	radioStation	wordOfMouth
phoneCall	web	other	otherWeb
ad	attend	extendedClasses	registration_id

Multiple_discount

id	name	numCourses	discount
program	quarter_id	minNumCourses	

Quarter

id	name	startTime	stopTime
showQuarter	parking_fee	osher_parking_pass	osher_membership_fee
max_osher_courses_allowed			

Receipt

id	registration_id	coupon_id	couponAmount
parkingPass	ParkingOsher	parkingPassQuantity	parkingPassAmount
subTotal	total	osherMembership	

Tab. 3.4: Database Relational Schema - Cont.

Registration

id	lastName	firstName	middleInitial
ssn	otherName	address	city
state	zip	email	workEmail
homeEmail	employer	dayphone	workDayPhone
homeDayPhone	otherDayPhone	nightPhone	workNoghtPhone
homeNightPhone	otherNightPhone	dob	sex
status	rules	quarter_id	dayPhoneExt
rules	ethnicity	apt	poBox
country	enrolledDate	correlationId	transactionId
avsCode	cvv2Match	payerId	orderTime

Staff

id	username	password	firstName
lastName	email	regNotify	specifyRegNotify
manage_course_permission	process_student_permission	manage_report_permission	

3.2.2 Database Design

In the CELORS, the database stores course categories for students to view and register, and student registration records and payment information for CEL staff to access and retrieve. As mentioned earlier, the CELORS uses Hibernate framework to handle connections between MySQL database and its application.

All CELORS database functionalities are stored in the Database Access Object (DAO). The DAO files extend the ParentDAO file, which calls on a HibernateUtil class to handle all of the queries to the database. Table 3.5 is the XML configuration file.

Tab. 3.5: hibernate.cfg.xml File

```
<?xml version='1.0'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
 <session-factory>
  <!-- Database connection settings -->
  <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
  <property name="connection.url">
    jdbc:mysql://127.0.0.1/ors2?autoReconnect=true</property>
  <property name="connection.username">CELORSDBUSER</property>
  <property name="connection.password">CELORSPASSWORD</property>
  <!-- JDBC connection pool -->
  <property name="c3p0.min_size">2</property>
  <property name="c3p0.max_size">5</property>
  <property name="c3p0.timeout">1800</property>
  <property name="c3p0.max_statements_per_connection">80</property>
  <!-- SQL dialect -->
  <property name="dialect">org.hibernate.dialect.MySQLInnoDBDialect</property>
  <!-- Enable Hibernate's automatic session context management -->
  <property name="current_session_context_class">thread</property>
  <!-- Set JDBC isolation level -->
  <property name="cache.provider_class">org.hibernate.cache.EhCacheProvider</property>
  <!-- Echo all executed SQL to stdout -->
  <property name="show_sql">true</property>
  <property name="use_sql_comments">true</property>
 </session-factory>
</hibernate-configuration>
```

3.2.3 *Admin Table*

The admin table contains the CELORS administrator's login information. The default administrator username is "admin" and the CEL-designated administrator can create, delete, or modify staff user accounts. See Table 3.6 and Figure 3.2 for details.

Tab. 3.6: Structure of Admin Table

Field	Type	Null	Key	Default	Extra
id	int(11)		PRI	NULL	auto_increment
username	varchar(255)			NULL	
password	varchar(255)	Yes		NULL	

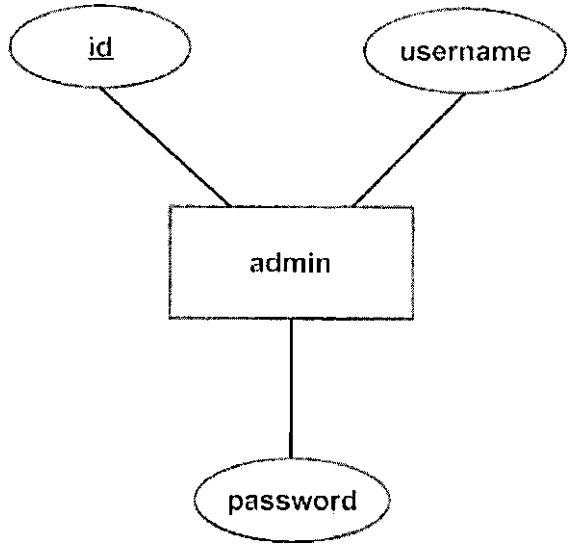


Fig. 3.2: Entity Relational Diagram - Admin Table

3.2.4 *Category Table*

The category table contains its name and other essential attributes. It is also associated with the quarter table.

A category can be viewable by setting the "visible" field as TRUE or 1. The default display sequence is sorted by name in alphabetical order. However, it can be changed by setting the "view_order" attribute to a larger number (default is 0). See Table 3.7 and Figure 3.3 for details.

Tab. 3.7: Structure of Category Table

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(255)	NO		NULL	
bulletin_id	int(11)	NO		NULL	
visible	char(1)	NO		NULL	
view_order	int	NO		NULL	

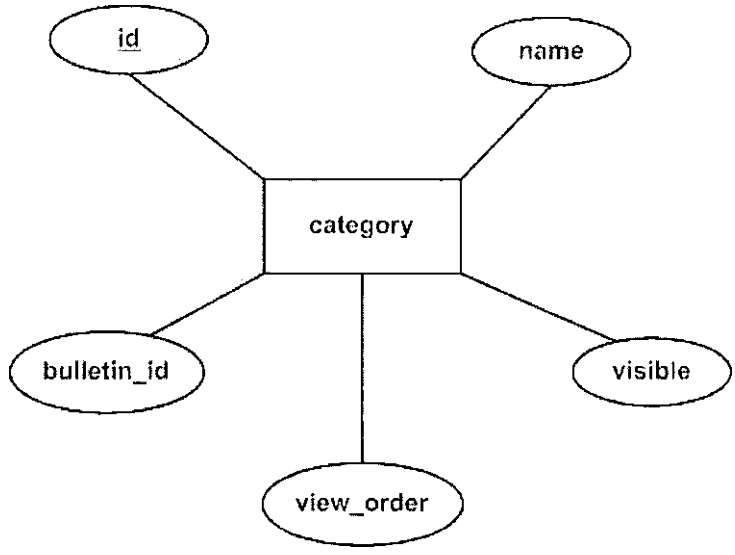


Fig. 3.3: Entity Relational Diagram - Category Table

3.2.5 Coupon Table

The coupon table stores coupon names and discount rates and is associated with the quarter table. The College of the Extended Learning hasn't provided any coupon promotions for several years. This table may be dropped in the future. See Table 3.8 and Figure 3.4 for details.

Tab. 3.8: Structure of Coupon Table

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(255)	NO	MUL		
discount	double	YES		NULL	
quarter_id	int(11)	YES		NULL	

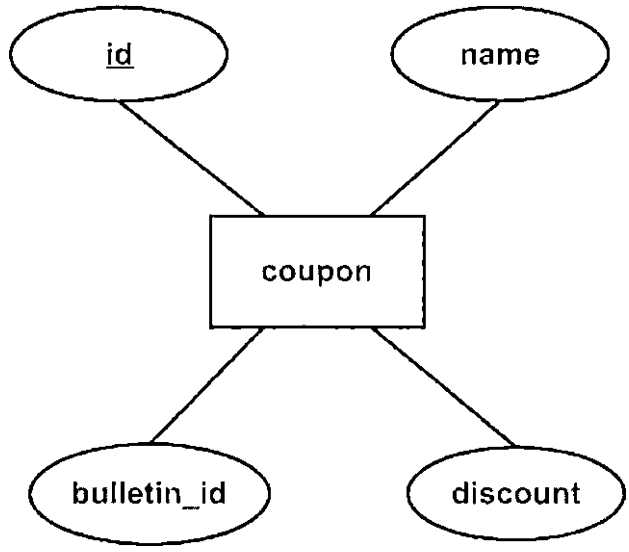


Fig. 3.4: Entity Relational Diagram - Coupon Table

3.2.6 Course Table

This table contains course-related information and is associated with the quarter table. In CEL's business rules, a class may be offered for credit or non-credit or for different fees are distinguished by different schedule numbers. In other words, the same course number might have more than one schedule number. Therefore, the course table is capable of storing up to four schedule numbers and fees. The "regDeadline" field is used to control whether the class is available to register online. If the registration deadline for a class has passed, the student must contact CEL by phone in order to register for it. See Table 3.9, Table 3.10 and Figure 3.5 for details.

Tab. 3.9: Structure of Course Table

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
creditSchedule	varchar(255)	YES		NULL	
nonCreditSchedule	varchar(255)	YES		NULL	
schedule2	varchar(255)	YES		NULL	
schedule3	varchar(255)	YES		NULL	
name	varchar(255)	YES		NULL	
description	text	YES		NULL	
number	varchar(255)	NO		NULL	
units	varchar(255)	YES		NULL	
creditFee	double	YES		NULL	
nonCreditFee	double	YES		NULL	
fee2	double	YES		NULL	
fee3	double	YES		NULL	
desc0	varchar(255)	YES		NULL	
desc1	varchar(255)	YES		NULL	
desc2	varchar(255)	YES		NULL	
desc3	varchar(255)	YES		NULL	
daysDates	varchar(255)	YES		NULL	
classTimes	varchar(255)	YES		NULL	
location	varchar(255)	YES		NULL	
regDeadline	date	YES		NULL	
instructor	varchar(255)	YES		NULL	
quarter_id	int(11)	YES		NULL	
seatSize	int(11)	NO		0	
seatsTaken	int(11)	NO		0	

Tab. 3.10: Structure of Course Table - Cont.

Field	Type	Null	Key	Default	Extra
view	char(1)	YES		NULL	
orsRegDeadline	date	YES		NULL	

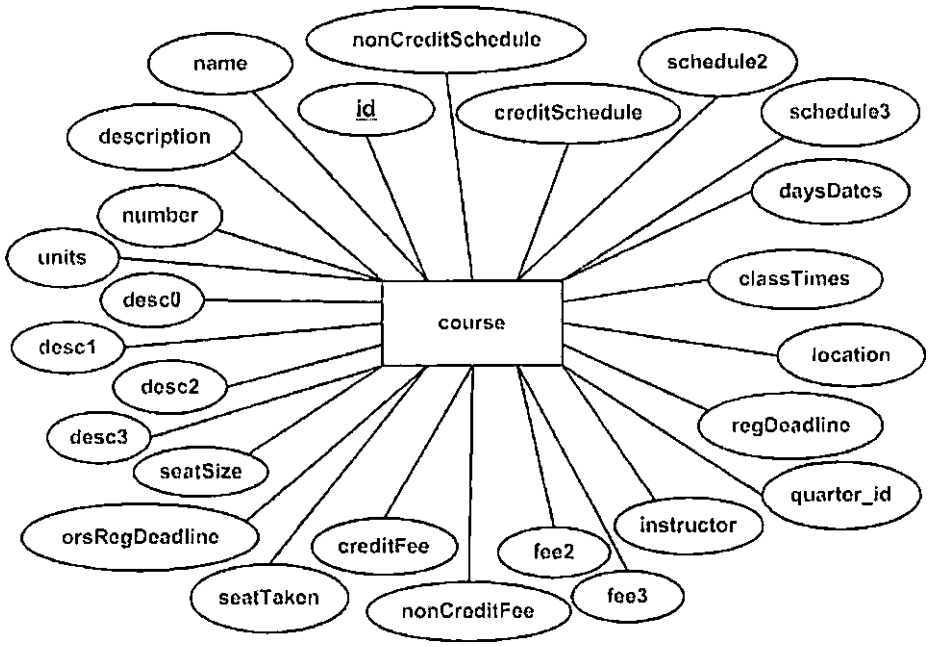


Fig. 3.5: Entity Relational Diagram - Course Table

3.2.7 Course_category Table

This is a junction table for associating course and category tables in order to represent their many-to-many relationship. See Table 3.11 and Figure 3.6 for details.

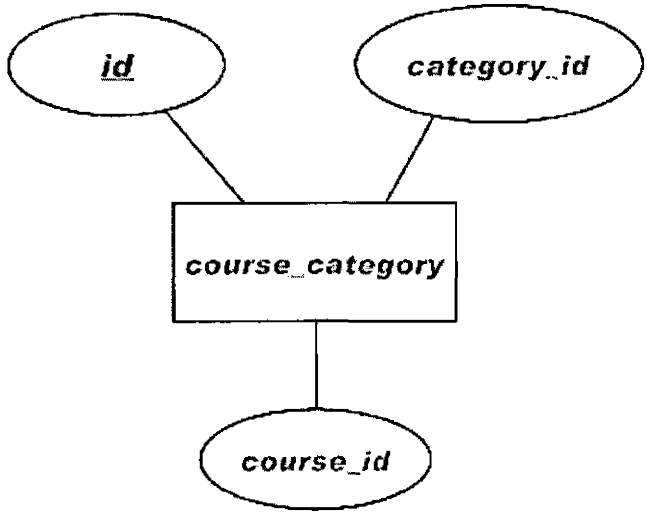


Fig. 3.6: Entity Relational Diagram - Course_Category Table

Tab. 3.11: Structure of Course_Category Table

Field	Type	Null	Key	Default	Extra
id	int(11)	YES		NULL	
category_id	int(11)	YES		NULL	
course_id	int(11)	YES		NULL	

3.2.8 Course_discount Table

This is a junction table for associating course and discount tables in order to representing their many-to-many relationship. See Table 3.12 and Figure 3.7 for details.

Tab. 3.12: Structure of Course_Discount Table

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	NULL	auto_increment
course_id	int(11)	YES	MUL	NULL	
discount_id	int(11)	YES	MUL	NULL	

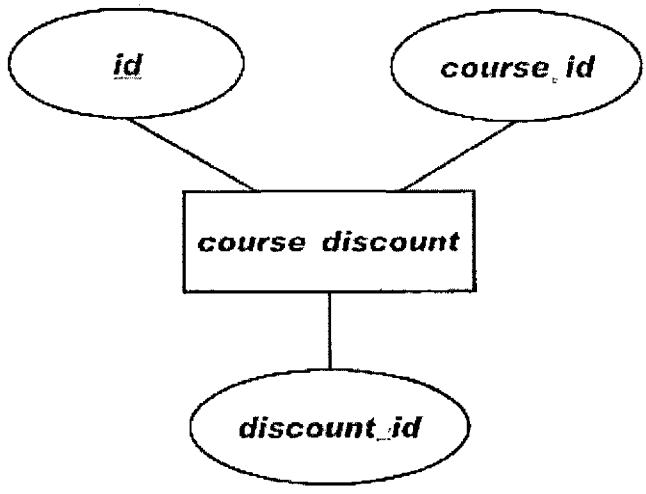


Fig. 3.7: Entity Relational Diagram - Course_Discount Table

3.2.9 Course_multiple_discount Table

This is a junction table for associating course and multiple discount tables to represent their many-to-many relationship. See Table 3.13 and Figure 3.8 for details.

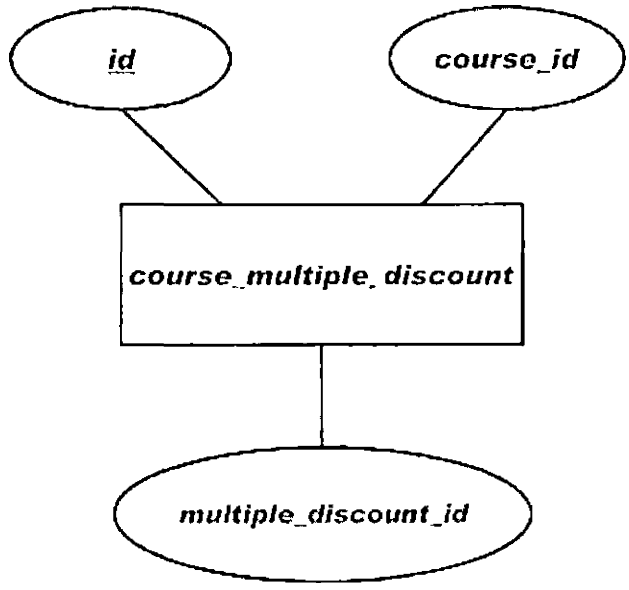


Fig. 3.8: Entity Relational Diagram - Course_Multiple_Discount Table

Tab. 3.13: Structure of Course_Multiple_Discount

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	NULL	auto_increment
course_id	int(11)	YES	MUL	NULL	
multiple_discount_id	int(11)	YES	MUL	NULL	

3.2.10 Course_registration Table

This is a junction table for associating course and registration tables in order to represent their many-to-many relationship. Through this table, we can easily query courses registered per registration, or registered students per course. See Table 3.14 and Figure 3.9 for details.

Tab. 3.14: Structure of Course_Registration

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	NULL	auto_increment
course_id	int(11)	YES	MUL	NULL	
registration_id	int(11)		MUL	NULL	
feeType	int(11)	YES		NULL	

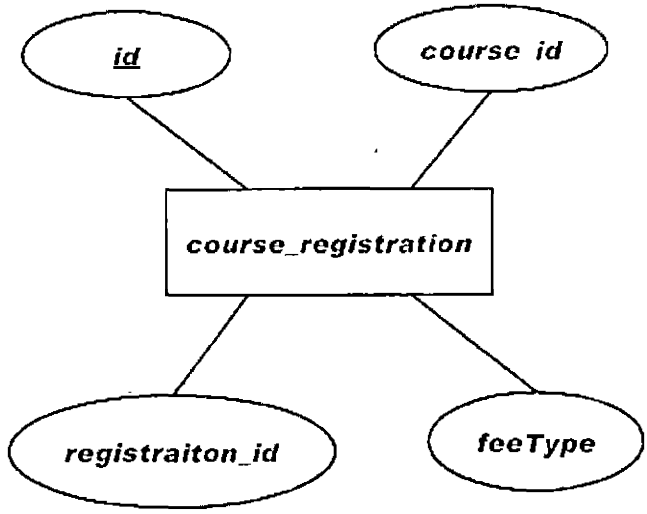


Fig. 3.9: Entity Relational Diagram - Course_Registration Table

3.2.11 Discount Table

The Discount table contains the discount name, discount rate, and is associated with a quarter id. See Table 3.15 and Figure 3.10 for details.

Tab. 3.15: Structure of Discount Table

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(255)	YES		NULL	
discount	double	YES		NULL	
quarter_id	int(11)	YES		NULL	

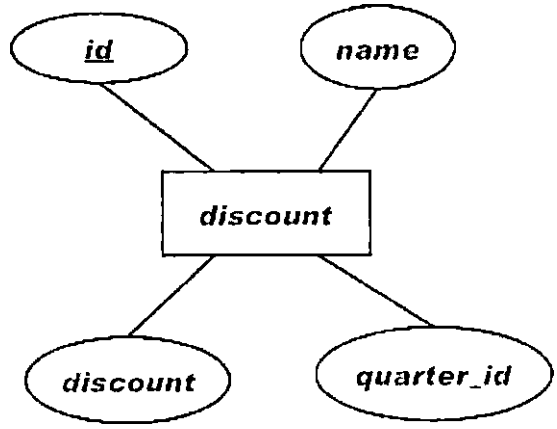


Fig. 3.10: Entity Relational Diagram - Discount Table

3.2.12 Discount_item Table

The Discount_item table collects discounted course details during registration. Every record is associated with a receipt record. This table keeps track of the discounted course name, course number, discount rate (in percentage), and the applied discount amount. See Table 3.16 and Figure 3.11 for details.

Tab. 3.16: Structure of Discount_Item Table

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(255)	YES		NULL	
discount	double	YES		NULL	
value	double	YES		NULL	
courseNumber	varchar(255)	YES		NULL	
receipt_id	int(11)	YES	MUL	NULL	

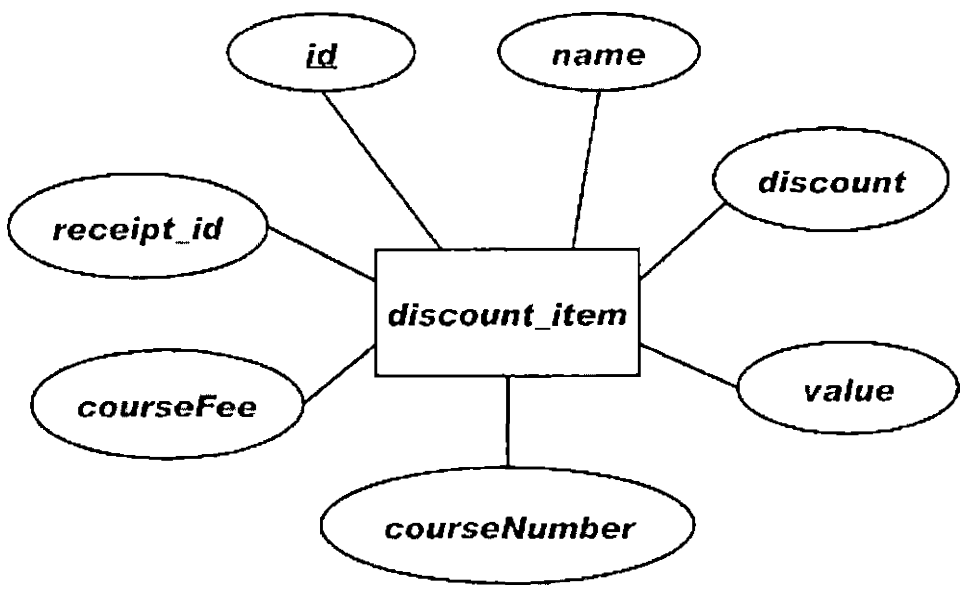


Fig. 3.11: Entity Relational Diagram - Discount_Item Table

3.2.13 Header Table

This table contains the header name and viewing status. A header acts as a title representing a group of categories. See Table 3.17 and Figure 3.12 for details.

Tab. 3.17: Structure of Header Table

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(255)	YES		NULL	
view	char(1)	YES		NULL	

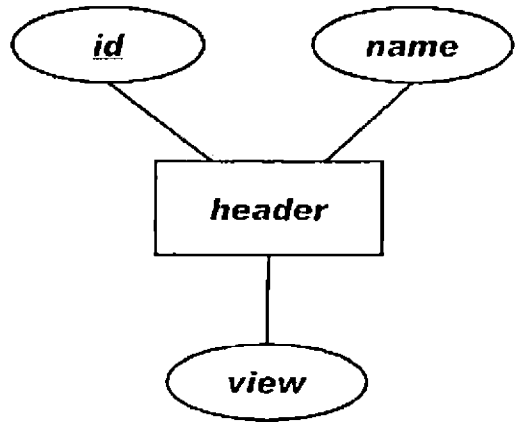


Fig. 3.12: Entity Relational Diagram - Header Table

3.2.14 Header_category Table

This is a junction table for Header and Category tables in representing many-to-many relationship. See Table 3.18 and Figure 3.13 for details.

Tab. 3.18: Structure of Header_Category Table

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	NULL	auto_increment
header_id	int(11)	YES		NULL	
category_id	int(11)	YES		NULL	

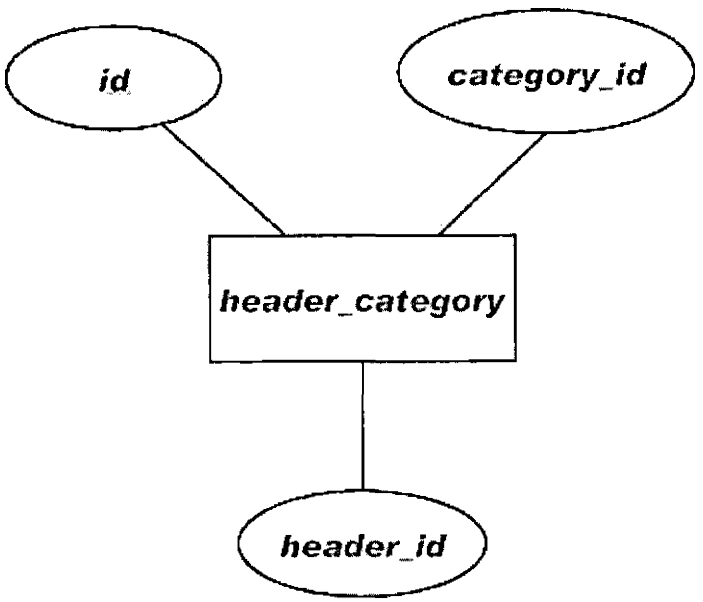


Fig. 3.13: Entity Relational Diagram - Header_Category Table

3.2.15 Marketing Table

This table is used to collect registered students' information for marketing purposes. See Table 3.19 and Figure 3.14 for detail.

Tab. 3.19: Structure of Marketing Table

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
classLevel	varchar(255)	YES		NULL	
degree	varchar(255)	YES		NULL	
bulletin	int(11)	YES		NULL	
flyer	int(11)	YES		NULL	
newspaper	varchar(255)	YES		NULL	
radioStation	varchar(255)	YES		NULL	
wordOfMouth	int(11)	YES		NULL	
phoneCall	int(11)	YES		NULL	
web	int(11)	YES		NULL	
other	varchar(255)	YES		NULL	
otherWeb	varchar(255)	YES		NULL	
ad	varchar(255)	YES		NULL	
attend	int(11)	YES		NULL	
extendClasses	int(11)	YES		NULL	
registration_id	int(11)	YES	MUL	NULL	

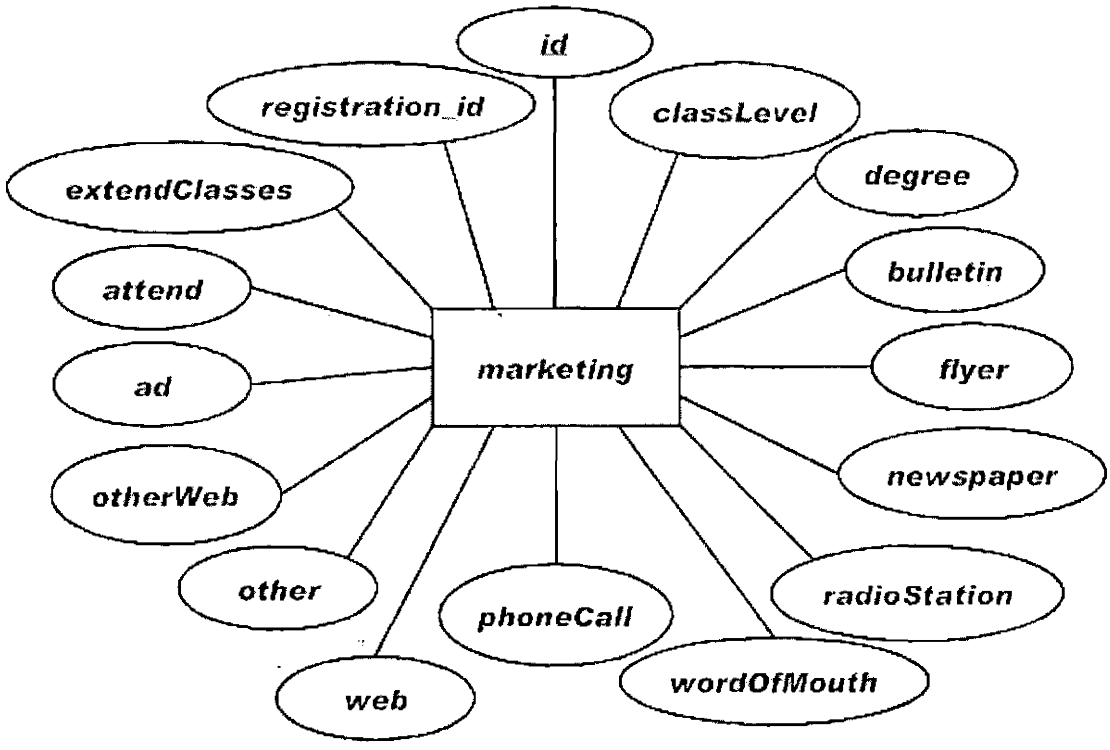


Fig. 3.14: Entity Relational Diagram - Marketing Table

3.2.16 Multiple Discount Table

The Multiple discount table is targeted to discount designated courses or categories. In the CELORS, the "numCourse" field relates to a discount which is applied when a minimum number of designated courses is selected. For example, when the "numCourses" field is set to "3", a discount is applied to the third and subsequent designated courses. The "minNumCourses" field is a newly added field. When this field is set, the discount will apply to all applicable courses when registering for the minimum number of designated courses.For example, if the "minNumCourses" field is set to "2", a discount will apply to all designated courses as long as a minimum of two courses are selected. "numCourses" and "minNumCourses" fields cannot be used at the same time. See Table 3.20 and Figure 3.15 for details.

Tab. 3.20: Structure of Multiple_Discount Table

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(255)	NO		NULL	
numCourses	int(11)	YES		NULL	
discount	double	YES		NULL	
program	int(11)	YES		NULL	
quarter_id	int(11)	YES		NULL	
minNumCourses	int(11)	YES		NULL	

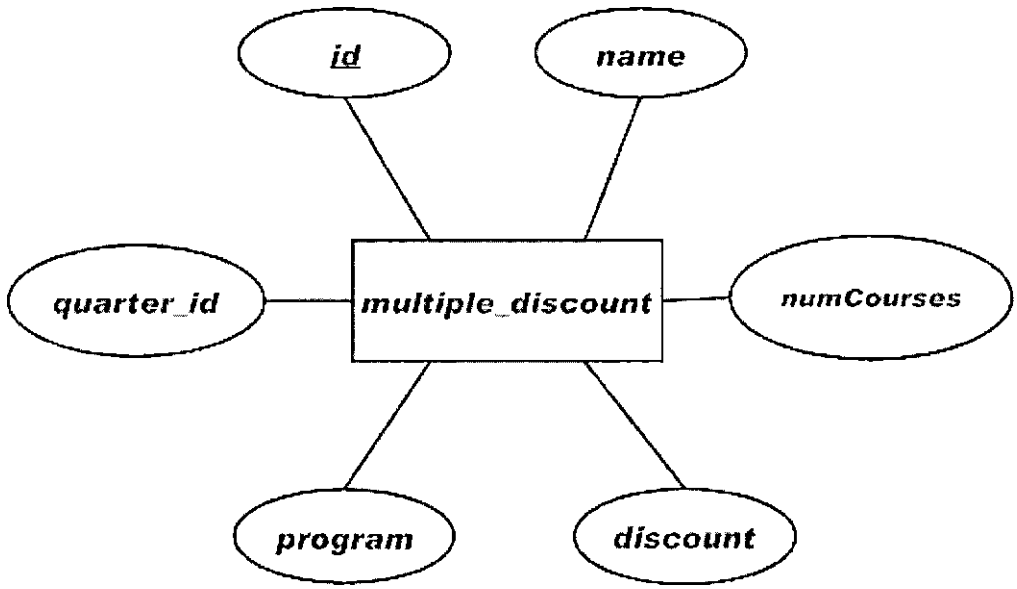


Fig. 3.15: Entity Relational Diagram - Multiple_Discount Table

3.2.17 Quarter Table

This table contains all the information relating to a particular quarter. In the CELORS, the parking_fee and parking_pass tables were dropped from the previous version of the CELORS and more attributes were added to this version. Now there are parking_fee, osher_parking_fee, osher_membership_fee, and max_osher_course_allowed. The showQuarter attribute controls the viewability by students of the course categories of the quarter. The parking_fee attribute refers to the daily parking fee for the CSUSB campus.

In the past, all Osher-related fees were hard-coded. If any fee changed, the course manager didn't have control to modify it. The system administrator had to check every single line of code, recompile, and update the production environment. Hence, those fees were added in this table and the CEL staff who have privileges to maintain courses can easily modify these fees. The osher_parking_fee is the quarterly parking fee for people taking Osher courses. The osher_membership_fee is charged to people who register for Osher courses. The max_osher_courses_allowed is the maximum number of Osher courses that a student can register per quarter. See Table 3.21 and Figure 3.16 for details.

Tab. 3.21: Structure of Quarter Table

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(255)	NO	UNI		
startTime	varchar(255)	YES		NULL	
stopTime	varchar(255)	YES		NULL	
showQuarter	int(11)	YES		NULL	
parking_fee	double	YES		0	
osher_parking_pass	double	YES		0	
osher_membership_fee	double	YES		0	
max_osher_courses_allowed	int(11)	YES		0	

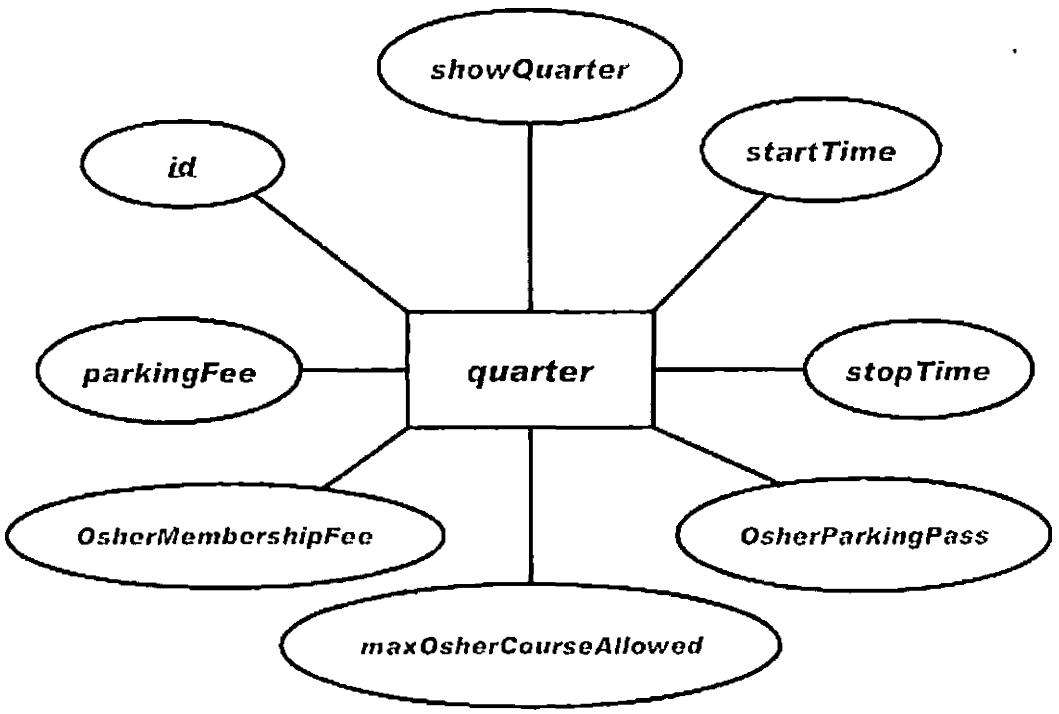


Fig. 3.16: Entity Relational Diagram - Quarter Table

3.2.18 Receipt Table

This table contains the total fees of courses taken, parking fee, membership fee, and applied coupon discount. Every receipt corresponds to a registration record. See Table 3.22 and Figure 3.17 for details.

Tab. 3.22: Structure of Receipt Table

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
registration_id	int(11)	YES		NULL	
coupon_id	int(11)	YES		NULL	
couponAmount	double	YES		NULL	
parkingPass	varchar(255)	YES		NULL	
parkingOsher	varchar(255)	YES		NULL	
parkingPassQuantity	varchar(255)	YES		NULL	
parkingPassAmount	double	YES		NULL	
subTotal	double	YES		NULL	
total	double	YES		NULL	
osherMembership	varchar(255)	YES		NULL	

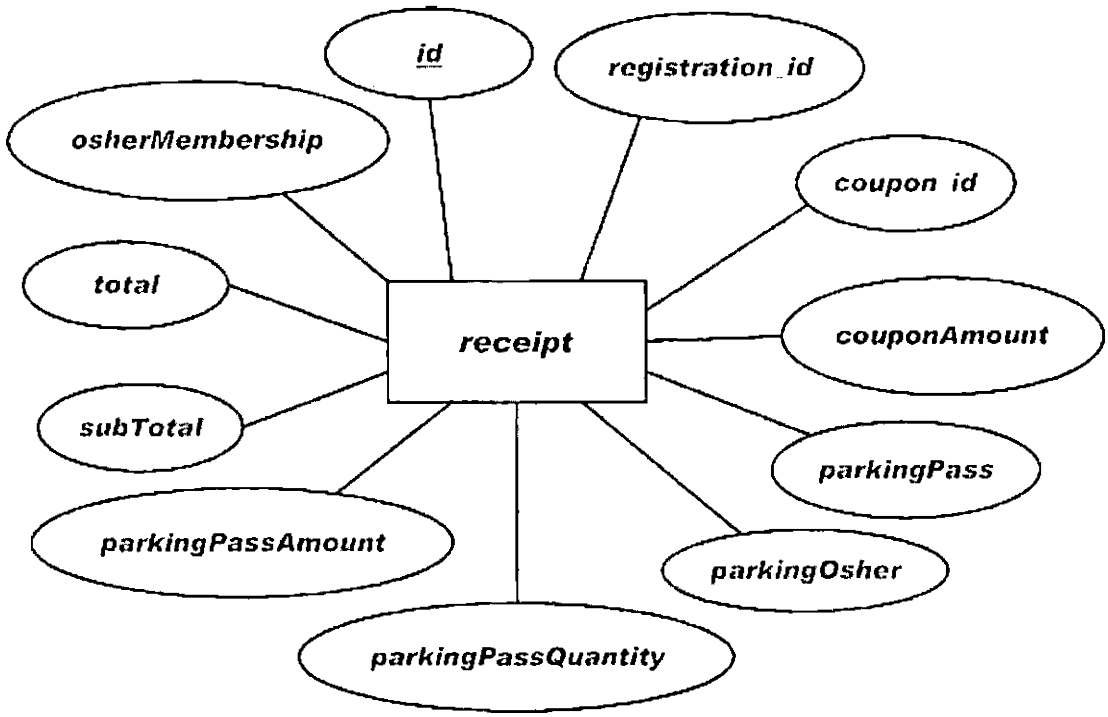


Fig. 3.17: Entity Relational Diagram - Receipt Table

3.2.19 Registration Table

This table stores students' contact information. See Table 3.23 and Figure 3.18 for details.

In the past, students were required to enter their social security numbers. Currently, we use student IDs (CSUSB MyCoyote ID) instead. "EnrolledDate" is added here for this purpose, which stores the date and time that the registration occurred. In 2009, a new online payment regulation, PCI compliance, was implemented at CSUSB. PayPal was selected for the CELORS's payment processing system. In order to easily trace payment records, six (6) new fields were added to this table: correlationId, transactionId, avsCode, cvv2Match, payerId, and orderTime.

Tab. 3.23: Structure of Registration Table

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
lastName	varchar(255)	YES		NULL	
firstName	varchar(255)	YES		NULL	
middleInitial	varchar(255)	YES		NULL	
ssn	text	YES		NULL	
otherName	varchar(255)	YES		NULL	
address	varchar(255)			NULL	
city	varchar(255)			NULL	
state	varchar(255)			NULL	
zip	varchar(255)			NULL	
email	varchar(255)			NULL	
workEmail	int(1)			NULL	
homeEmail	int(1)			NULL	
employer	varchar(255)			NULL	
dayphone	varchar(255)			NULL	
workDayPhone	int(1)			NULL	
homeDayPhone	int(1)			NULL	
otherDayPhone	int(1)			NULL	
nightPhone	varchar(255)			NULL	
workNightPhone	int(1)			NULL	
homeNightPhone	int(1)			NULL	
otherNightPhone	int(1)			NULL	
dob	date				

Tab. 3.24: Structure of Registration Table - Cont.

Field	Type	Null	Key	Default	Extra
sex	int(1)	YES		NULL	
status	int(11)	YES		NULL	
rules	int(1)	YES		NULL	
quarter_id	int(11)	NO		NULL	
dayPhoneExt	varchar(255)	YES		NULL	
rules	int(1)	YES		NULL	
ethnicity	varchar(255)	YES		NULL	
apt	varchar(255)	YES		NULL	
poBox	varchar(255)	YES		NULL	
country	varchar(255)	YES		NULL	
enrolledDate	date			NULL	
correlationId	varchar(255)	YES		NULL	
transactionId	varchar(255)	YES		NULL	
avsCode	varchar(255)	YES		NULL	
cvv2Match	varchar(255)	YES		NULL	
payerId	varchar(255)	YES		NULL	
orderTime	varchar(255)	YES		NULL	

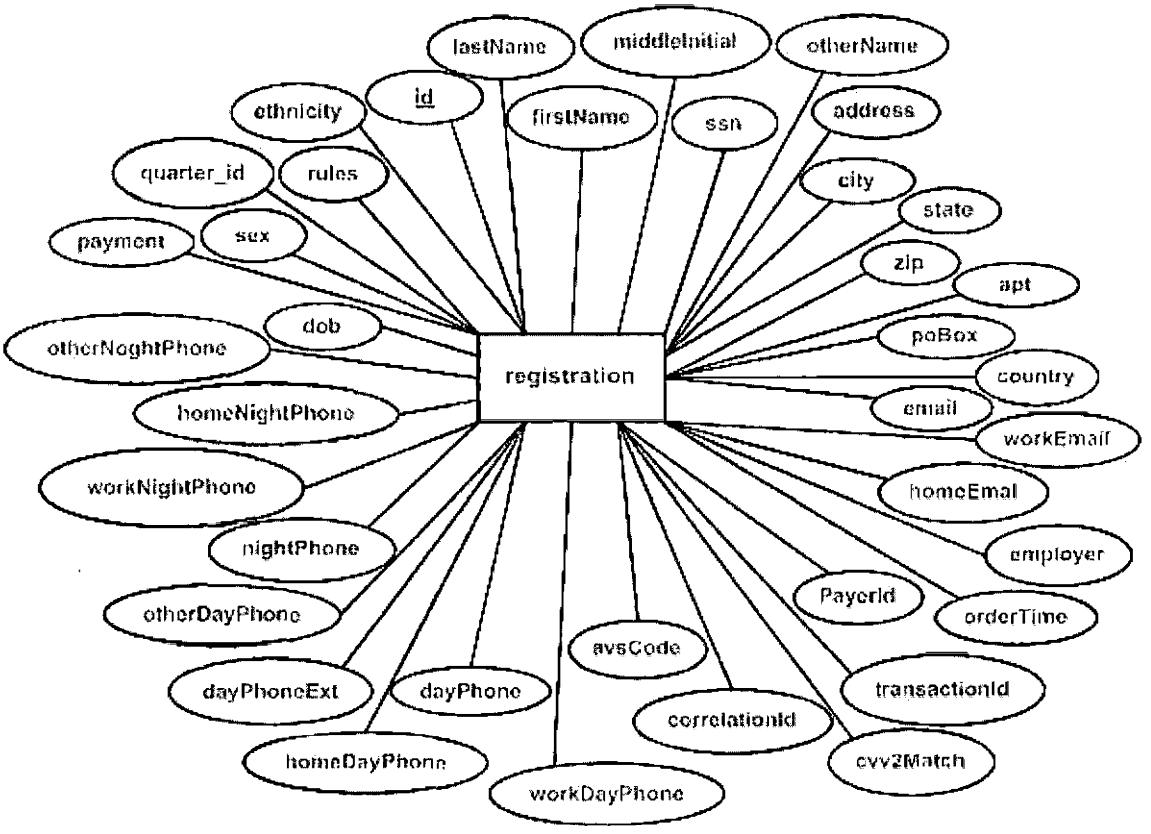


Fig. 3.18: Entity Relational Diagram - Registration Table

3.2.20 Staff Table

The Staff table contains the staff member's name, E-mail, login, and password for accessing the CELORS staff pages. Staff receive notification by E-mail if the regNotify or specialRegNotify is set to on. The specialRegNotify is used for special registration types, like cohort courses. The staff roles are defined here. These are course manager (manage_course_permission), registration manager (process_student_permission), and marketing report manager (manage_report_permission). The functions of the report manager are not implemented at this stage. It is created upon request of CEL staff. See Table 3.25 and Figure 3.19 for details.

Tab. 3.25: Structure of Staff Table

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	
username	varchar(255)	NO	UNI		
password	varchar(255)	Yes		NULL	
firstname	varchar(255)	YES		NULL	
lastname	varchar(255)	YES		NULL	
email	varchar(255)	YES		NULL	
regNotify	char(1)	YES		NULL	
specialRegNotify	char(1)	YES		NULL	
manage_course_permission	tinyint(1)	YES		0	
process_student_permission	tinyint(1)	YES		0	
manage_report_permission	tinyint(1)	YES		0	

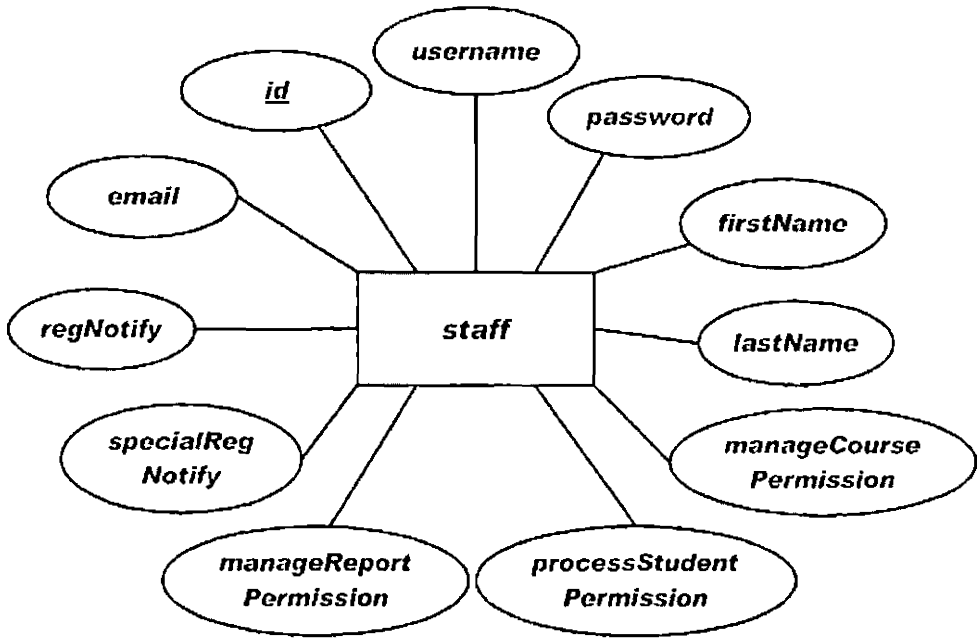


Fig. 3.19: Entity Relational Diagram - Staff Table

4. PROJECT IMPLEMENTATION

4.1 User Interface Design

The CELORS is a pure Web-based application. Therefore its user interfaces are designed to support most popular Web browsers, such as Microsoft Internet Explorer and Mozilla Firefox and to be accessible by all kinds of users. All interfaces are generated dynamically on the server side using JSP, based on the user's role or selections.

4.1.1 Administrator Pages

Admin Login Page

This is the starting page for administrators. In order to use these administration functionalities, an authorized CELORS administrator must login first to execute the CELORS admin tasks. (Figure 4.1).

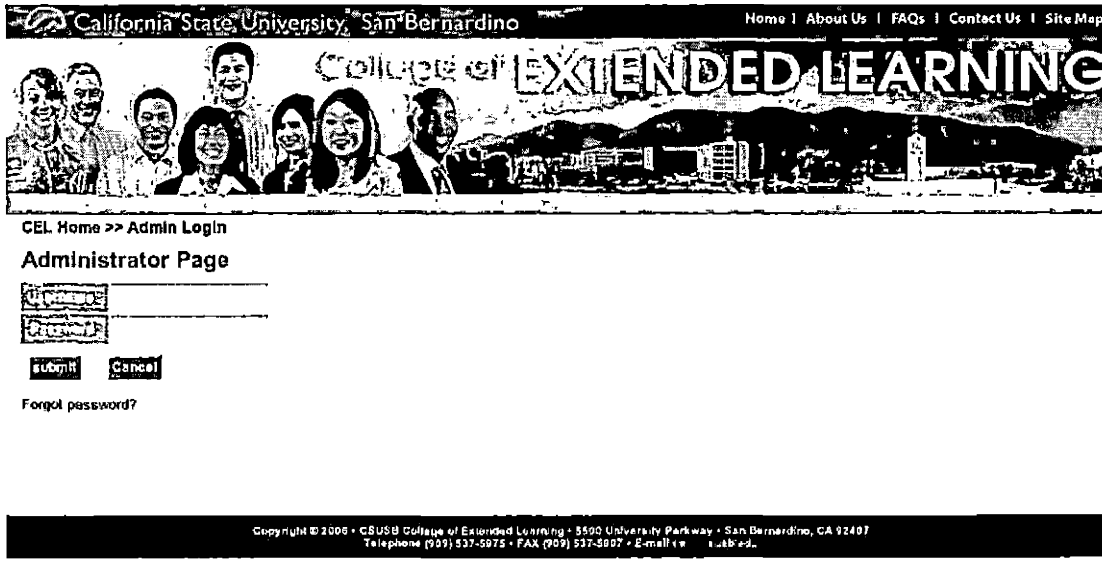


Fig. 4.1: Administrator Login Page

List Staff Page

This page lists all available accounts sorted by the creation time. It shows staff's name, E-mail, username, E-mail notification status, and permissions (Figure 4.2).

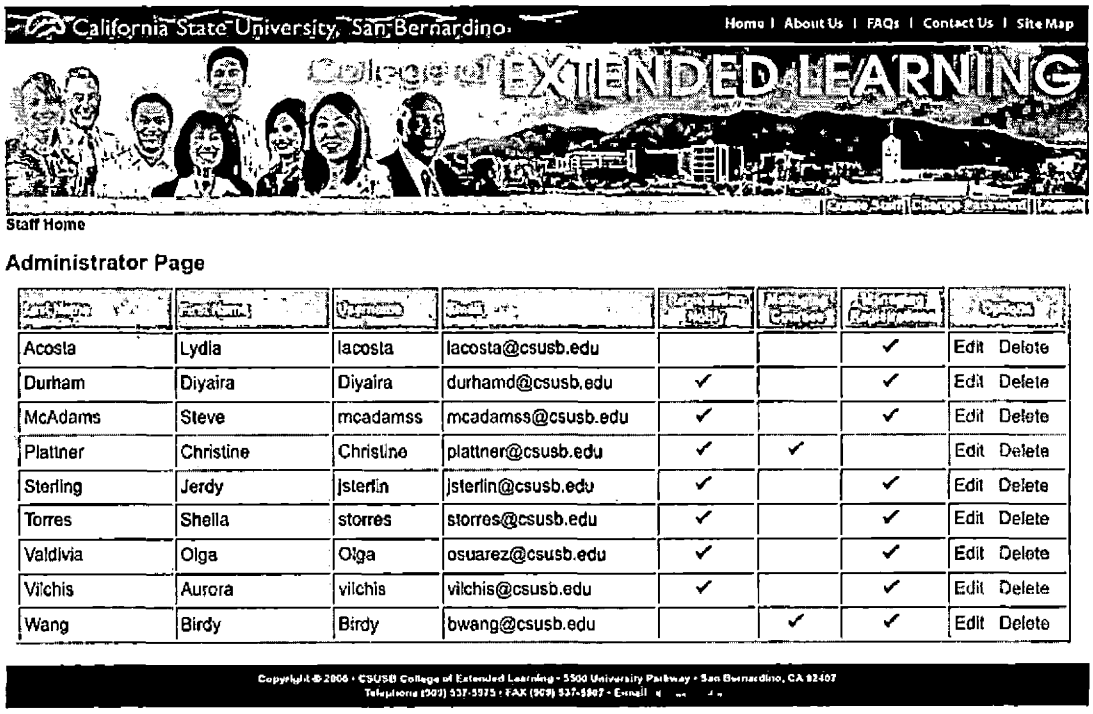


Fig. 4.2: List Staff Page

Create Staff Page

This page is designed to create an account for a staff person who is responsible for any CELORS tasks but doesn't have an account yet. The username has to be unique and there is no length limit. The E-mail field has to be a valid e-mail format (Figure 4.3).

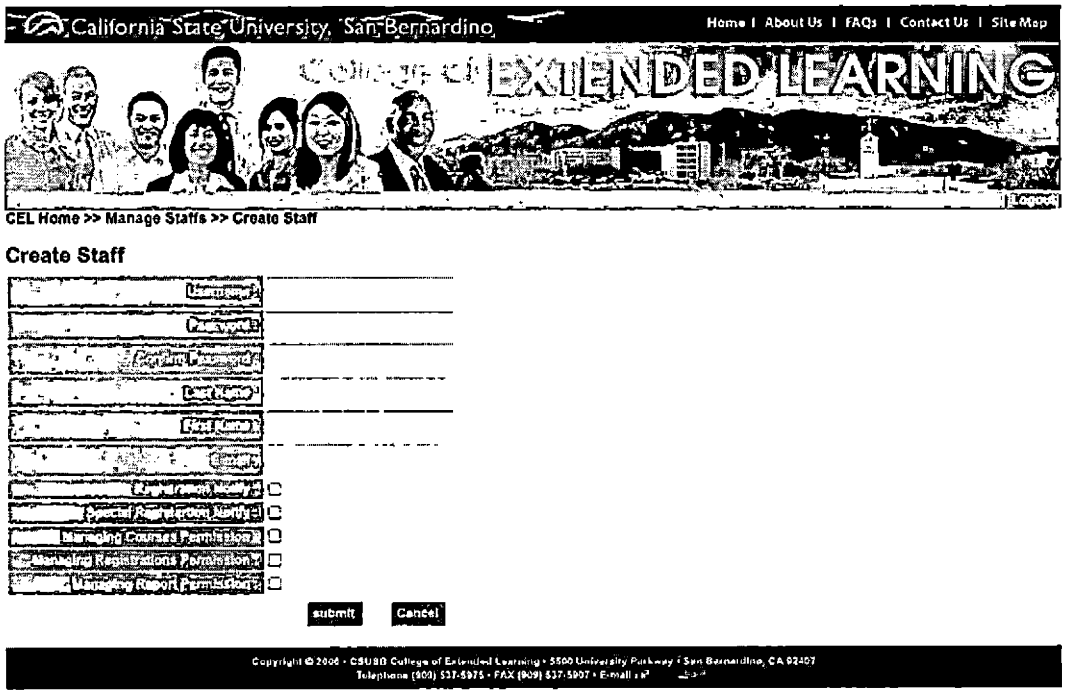


Fig. 4.3: Create Staff Page

Edit Staff Page

The administrator can reset the staff person's password, modify E-mail address and notification settings, or enable/disable privileges through this page (Figure 4.4).

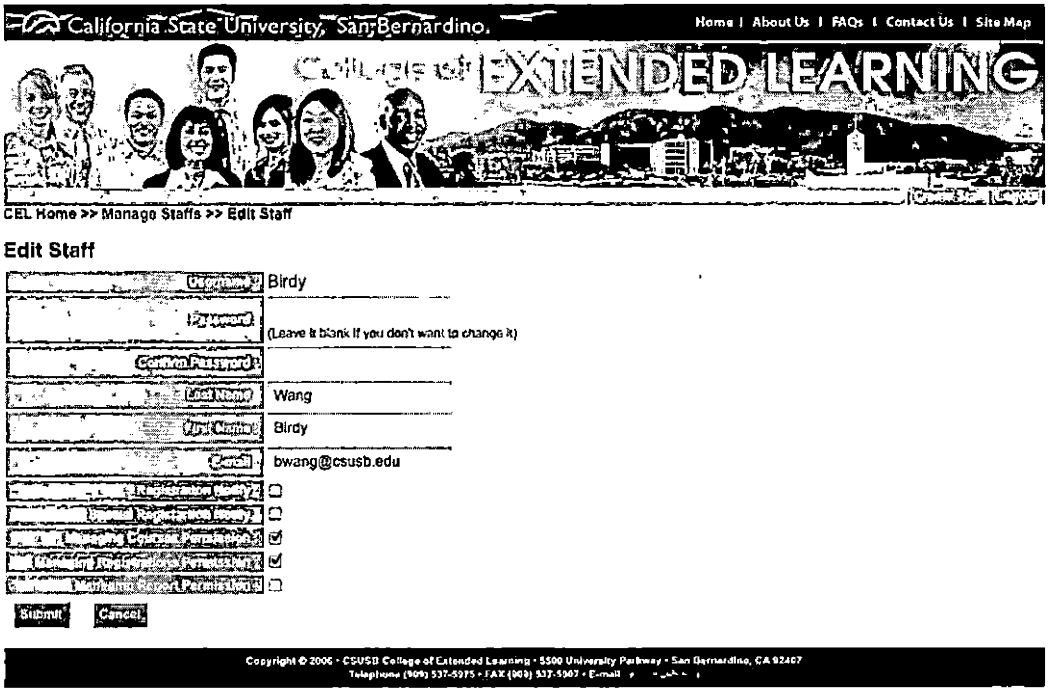


Fig. 4.4: Edit Staff Page

Delete Staff Page

This is a confirmation page to eliminate the chance of deleting a staff record by accident (Figure 4.5).

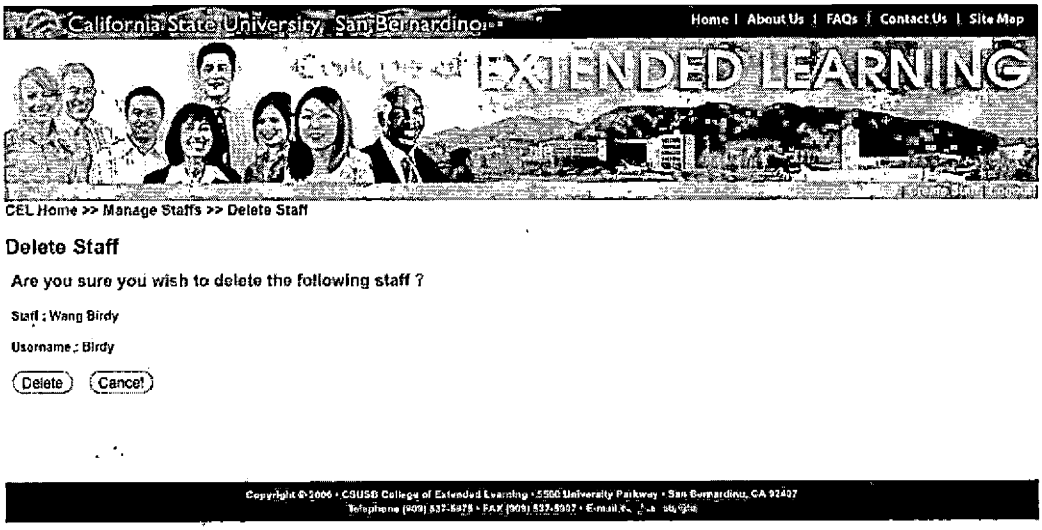


Fig. 4.5: Delete Staff Page

Change Password Page

This page is to change the administrator's own password (Figure 4.6).

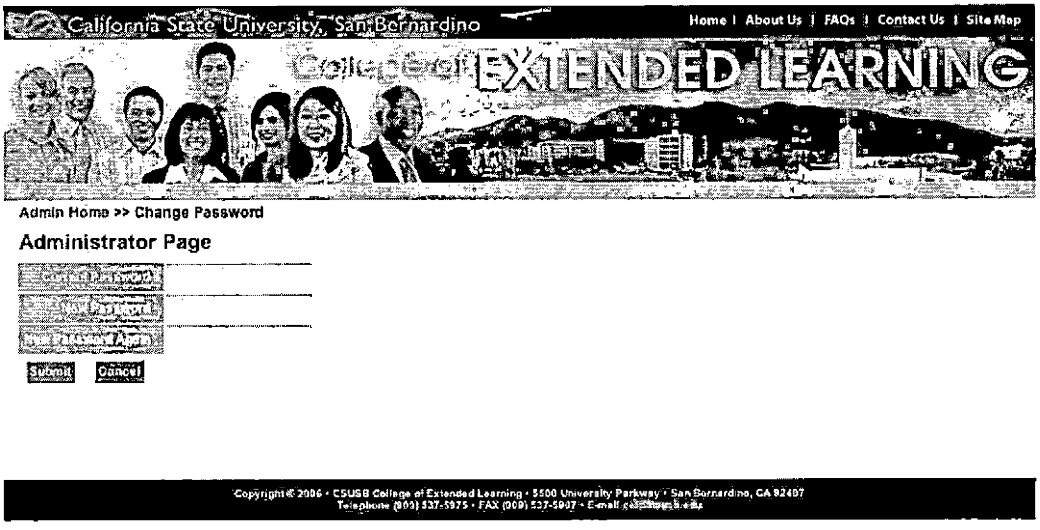


Fig. 4.6: Change Password Page - Administrator

4.1.2 *Staff Pages*

Login Page

This is the staff authentication page. Every authorized staff member is assigned a unique username and a default password for his/her first login. Once logged in, he/she can change his/her password through the Change Password page anytime. It is recommended that staff change their passwords after their first login (Figure 4.7).

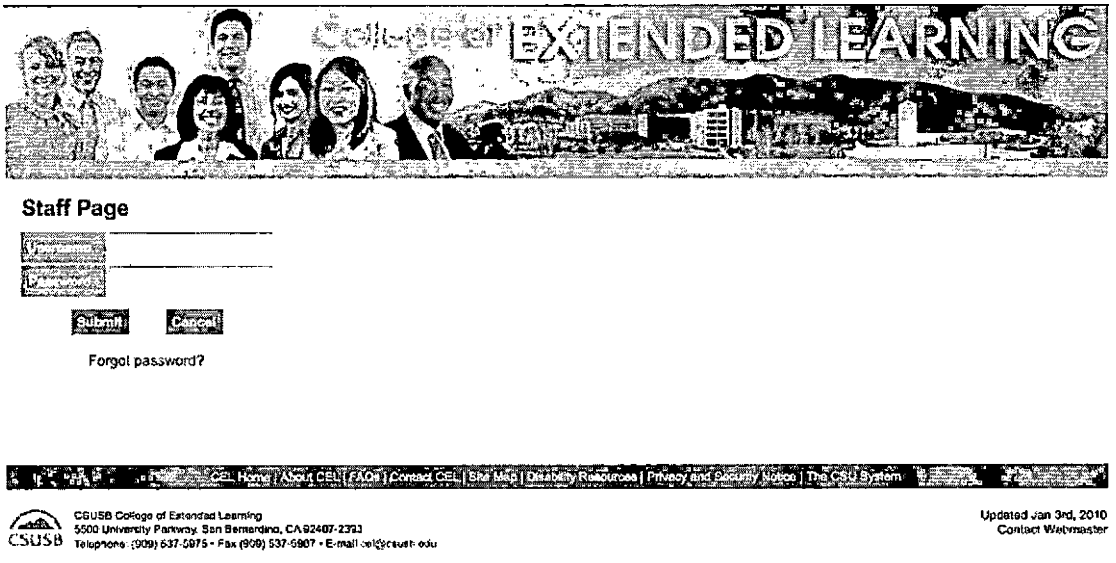
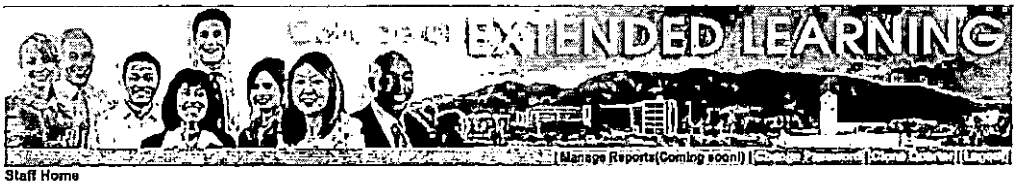


Fig. 4.7: Staff Login Page

List Quarters Page

This is the first page staff sees once logged in successfully. It lists past and current quarters in chronological order. It also shows the numbers of new or unprocessed registrations (Figure 4.8).



Staff Home

1	Fall 2004	Sep 10, 2004	Dec 11, 2004	0	0	Process Students	Manage Courses	Not Viewable
2	Winter 2005	Jan 1, 2005	Mar 31, 2005	0	0	Process Students	Manage Courses	Not Viewable
3	Spring 2005	Apr 1, 2005	Jun 30, 2005	0	0	Process Students	Manage Courses	Not Viewable
4	Summer 2005	Jun 1, 2005	Sep 1, 2005	0	0	Process Students	Manage Courses	Not Viewable
5	Fall 2005	Sep 1, 2005	Dec 1, 2005	0	0	Process Students	Manage Courses	Not Viewable
6	Winter 2006	Jan 1, 2006	Mar 31, 2006	0	0	Process Students	Manage Courses	Not Viewable
7	Spring 2006	Apr 1, 2006	Jun 30, 2006	0	0	Process Students	Manage Courses	Not Viewable
8	Summer 2006	Jun 17, 2006	Sep 23, 2006	0	0	Process Students	Manage Courses	Not Viewable
9	Fall 2006	Sep 8, 2006	Dec 13, 2006	0	0	Process Students	Manage Courses	Not Viewable
10	Winter 2007	Jan 1, 2007	Apr 1, 2007	0	0	Process Students	Manage Courses	Not Viewable
11	Spring 2007	Mar 30, 2007	Jun 16, 2007	0	0	Process Students	Manage Courses	Not Viewable
12	Summer 2007	Jun 17, 2007	Aug 31, 2007	0	0	Process Students	Manage Courses	Not Viewable
13	Fall 2007	Sep 1, 2007	Dec 31, 2007	0	0	Process Students	Manage Courses	Not Viewable

Fig. 4.8: List Quarters Page

Clone Quarter Page

Instead of typing in hundreds of courses, categories, and other attributes every quarter, the CELORS provides a clone function to simplify the course manager's quarterly tasks. All fees will be the same as the source quarter if left blank. Cloning a quarter might take a little bit longer to finish depending upon the size of the source quarter. Access to this attribute is limited to course managers only (Figure 4.9).

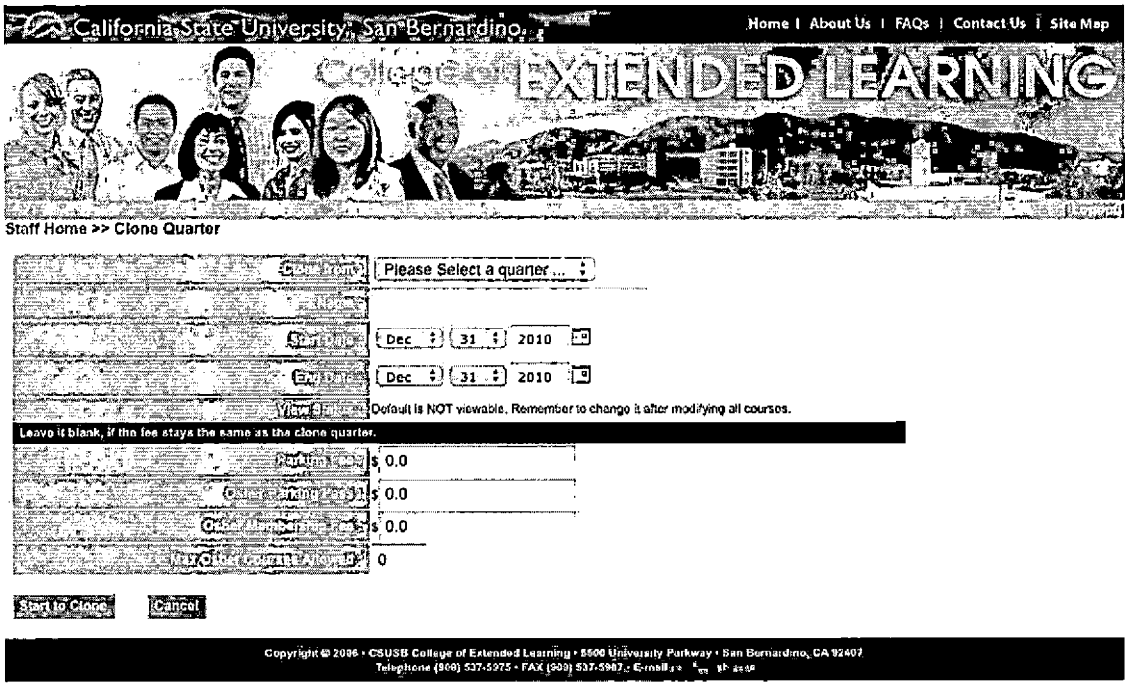


Fig. 4.9: Clone Quarter Page

Edit Quarter

This page provides course managers the ability to set up a variety of attributes associated with a specific quarter such as the parking pass fee, the Osher membership fee, etc. Access to this attribute is limited to course managers only (Figure 4.10).

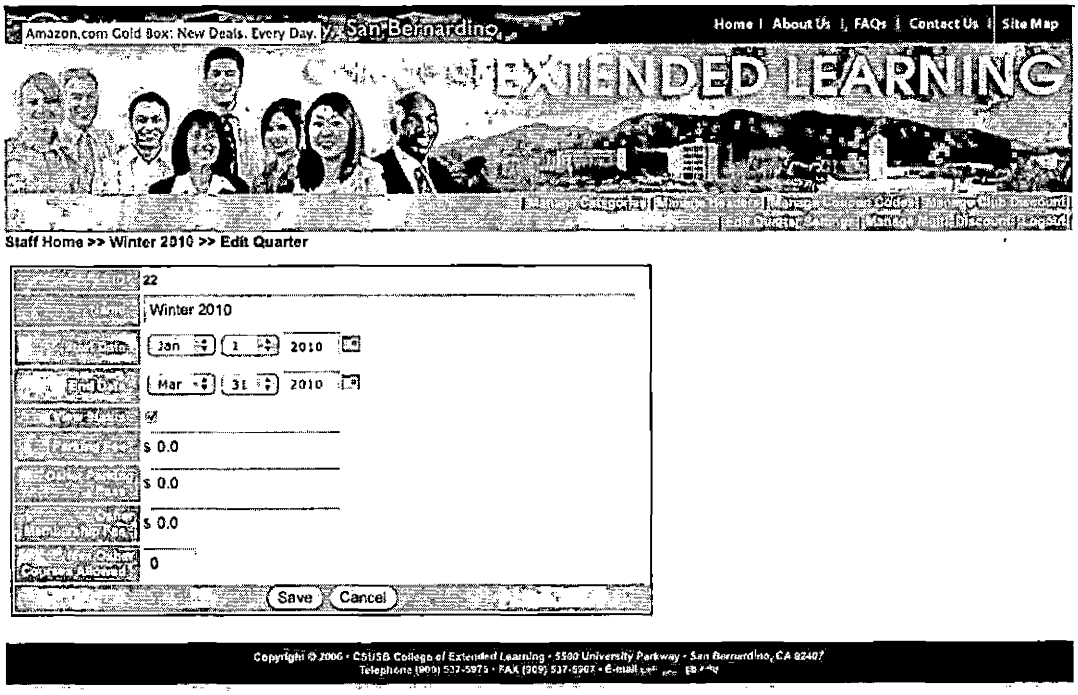


Fig. 4.10: Edit Quarter Page

View Category List Page

This page lists all categories for a selected quarter. Access to this attribute is limited to course managers only (Figure 4.11).

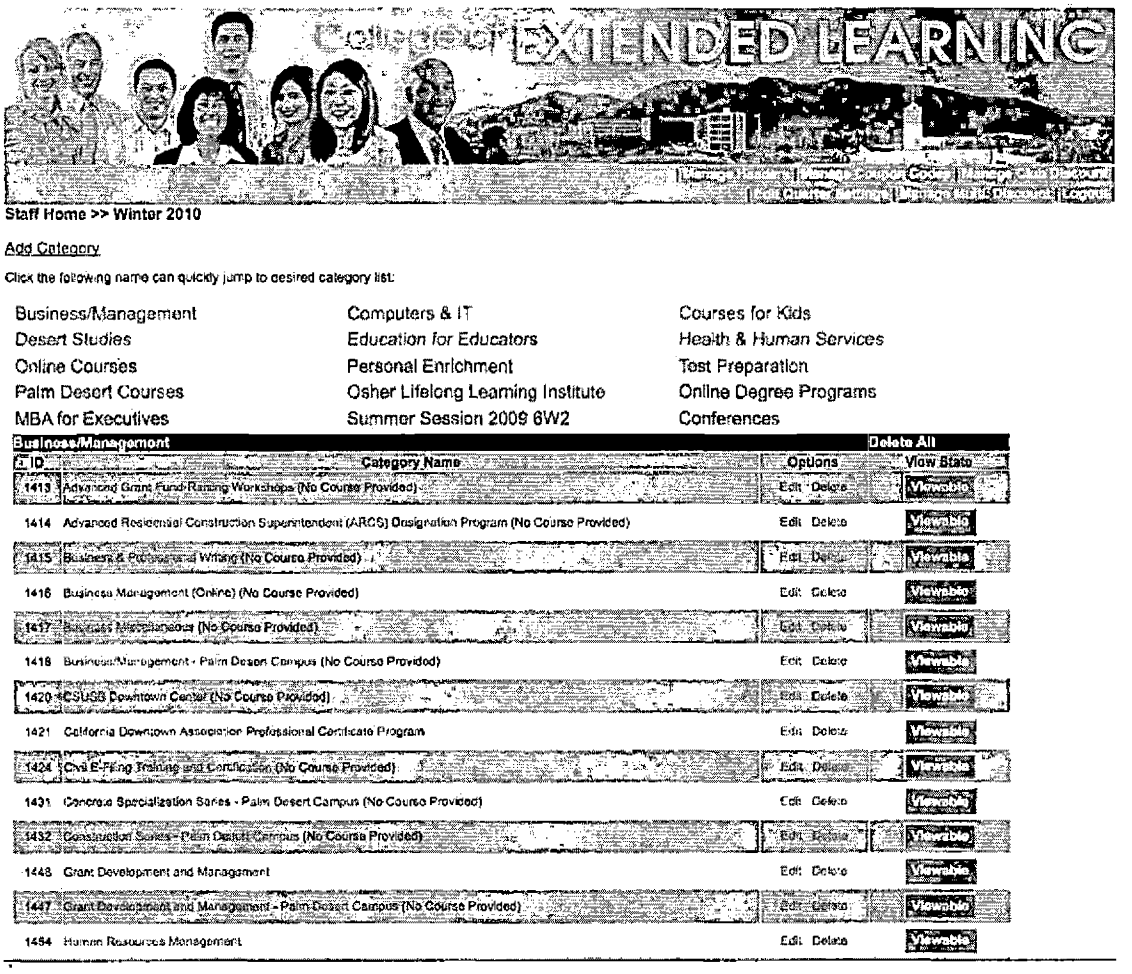


Fig. 4.11: View Category List Page

Create Category Page

This page is used to create a new category. A category can be set to be associated with more than one header. Default category is not viewable by the public. Access to this attribute is limited to course managers only (Figure 4.12).

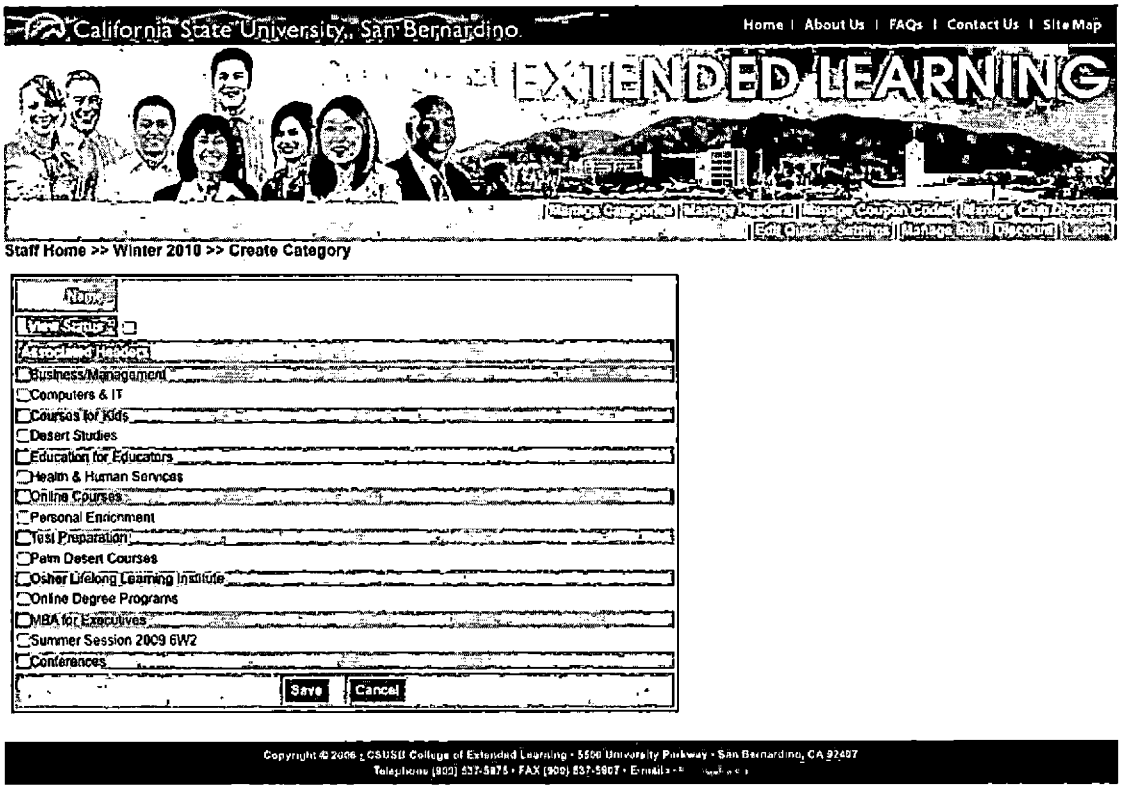


Fig. 4.12: Create Category Page

Edit Category Page

This page (Figure 4.13) allows the course manager to edit the category name, visibility, and its associated headers. Access to this attribute is limited to course managers only.

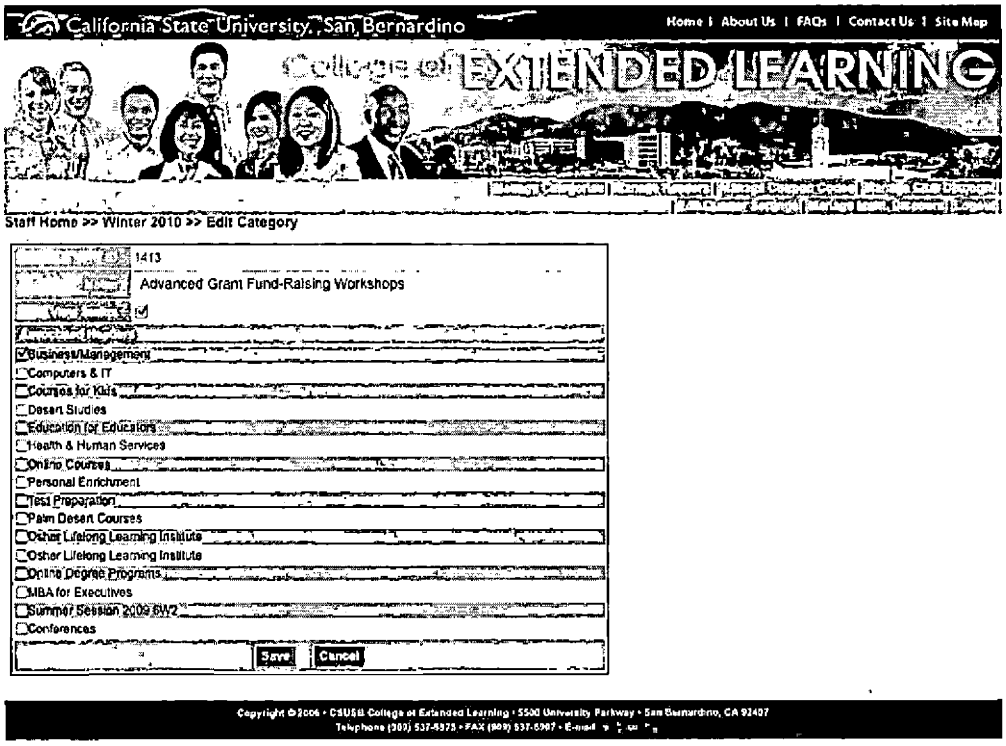


Fig. 4.13: Edit Category Page

Delete Category Page

This is a confirmation page (Figure 4.14) before a category is permanently removed from the database. The category cannot be deleted if any courses are associated with it. Access to this attribute is limited to course managers only.

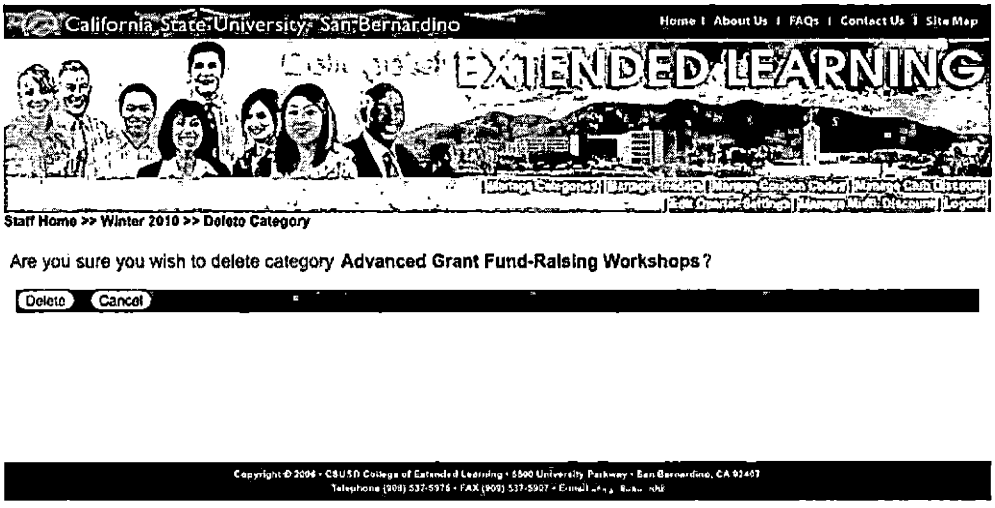


Fig. 4.14: Delete Category Page

View Course List Page

This page shows courses associated with a specific quarter and category. Access to this attribute is limited to course managers only (Figure 4.15).

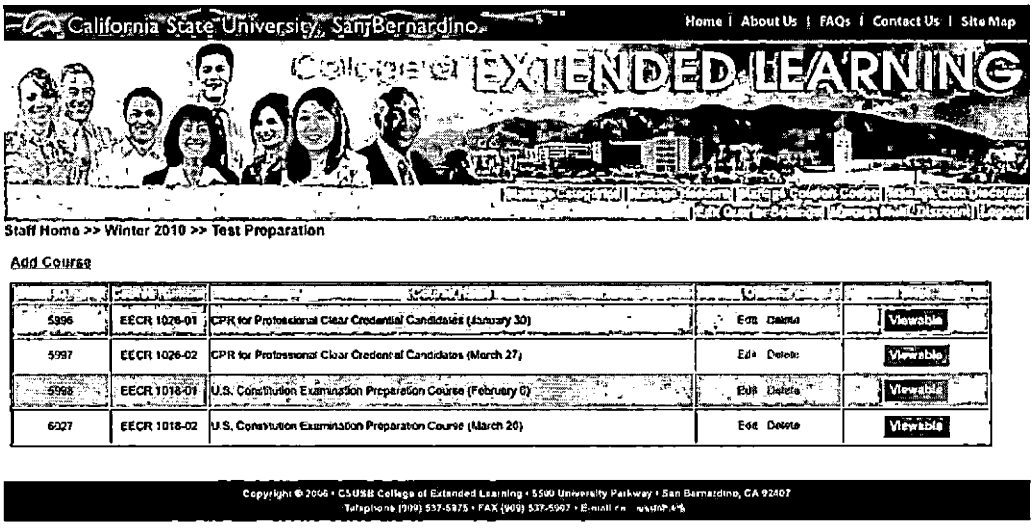


Fig. 4.15: View Course List Page

Create Course Page

This page will generate a new course associated with a designated quarter and category. If the registration deadline is set, students cannot register through the CELORS after that date. For online courses, the "No Registration Deadline Applied" box is checked. Access to this attribute is limited to course managers only (Figure 4.16 and Figure 4.17).

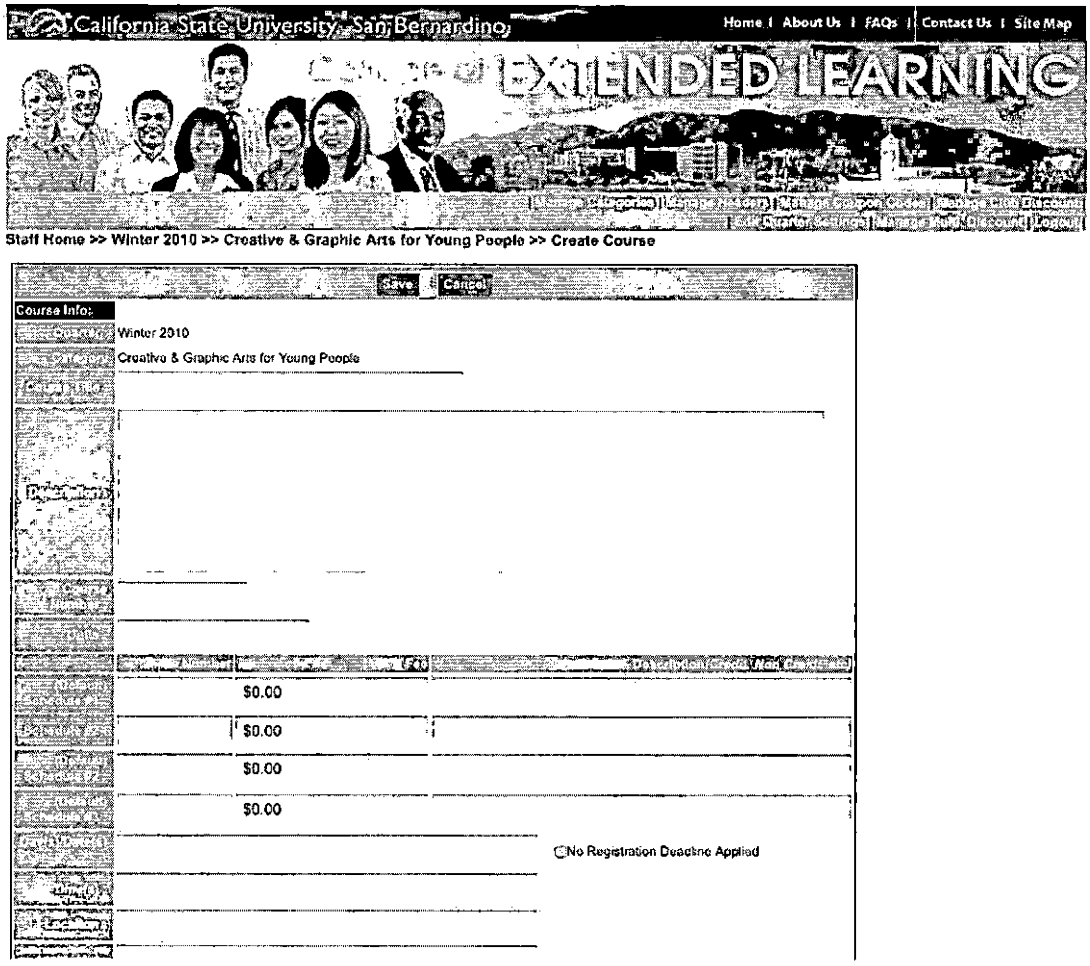


Fig. 4.16: Create Course Page

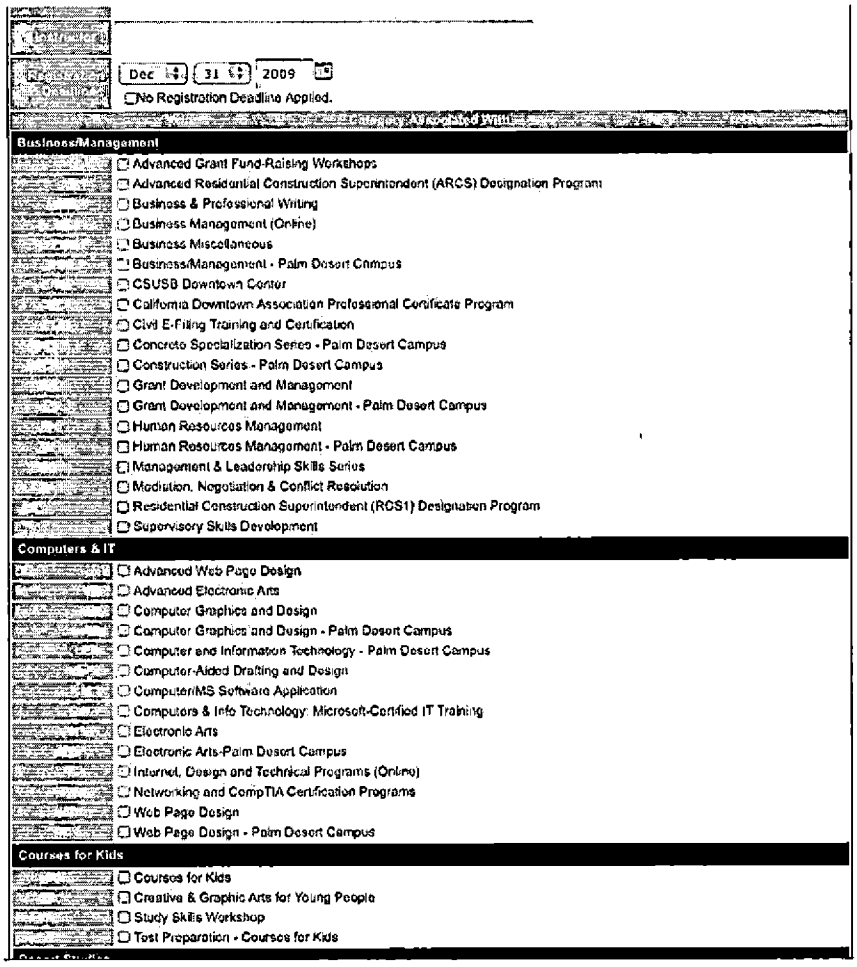


Fig. 4.17: Create Course Page - Cont.

Edit Course Page

Course attributes can be modified through this page (Figure 4.18 and Figure 4.19). Access to this attribute is limited to course managers only.

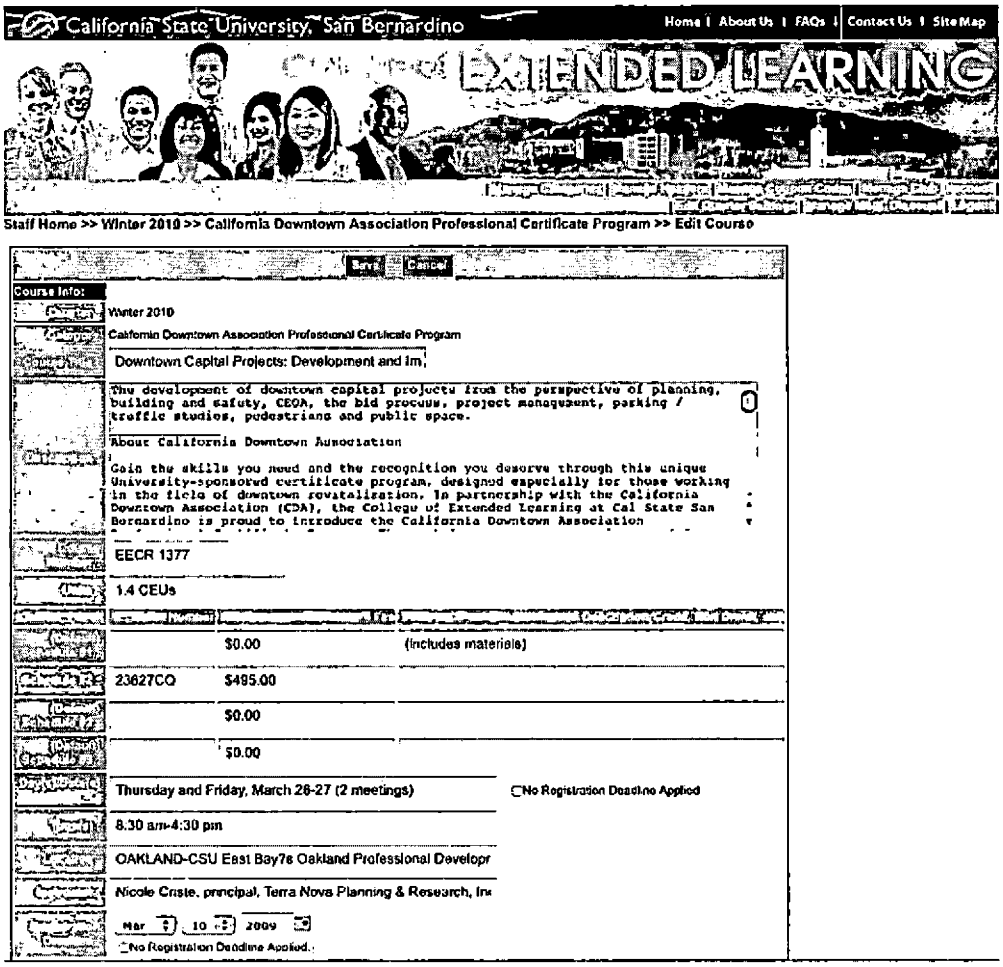


Fig. 4.18: Edit Course Page

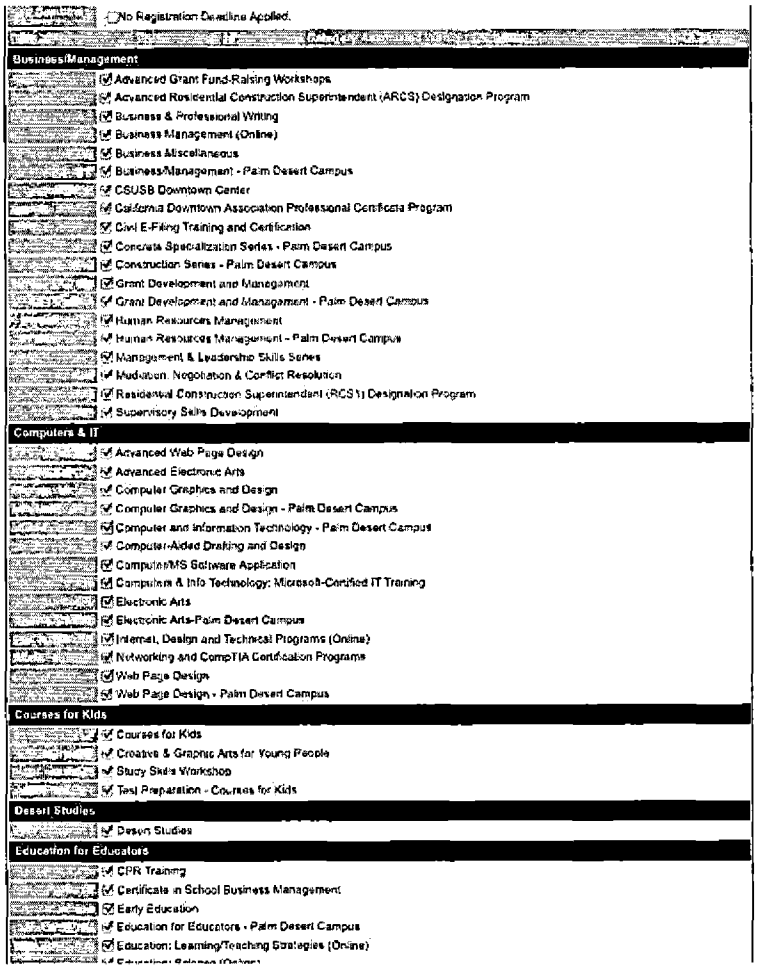


Fig. 4.19: Edit Course Page - Cont.

Delete Course Page

This is a confirmation page before a course is permanently removed from the database. The course cannot be deleted if any registration is associated with. Access to this attribute is limited to course managers only (Figure 4.20).

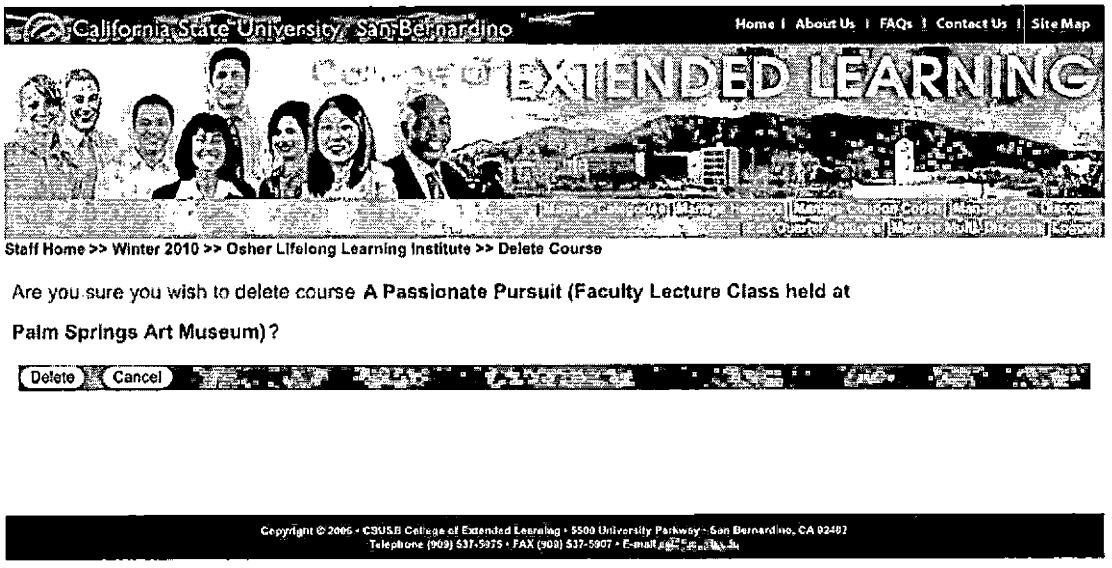


Fig. 4.20: Delete Course Page

View Header List Page

This page lists all headers in the CELORS and their viewable status. Access to this attribute is limited to course managers only (Figure 4.21).

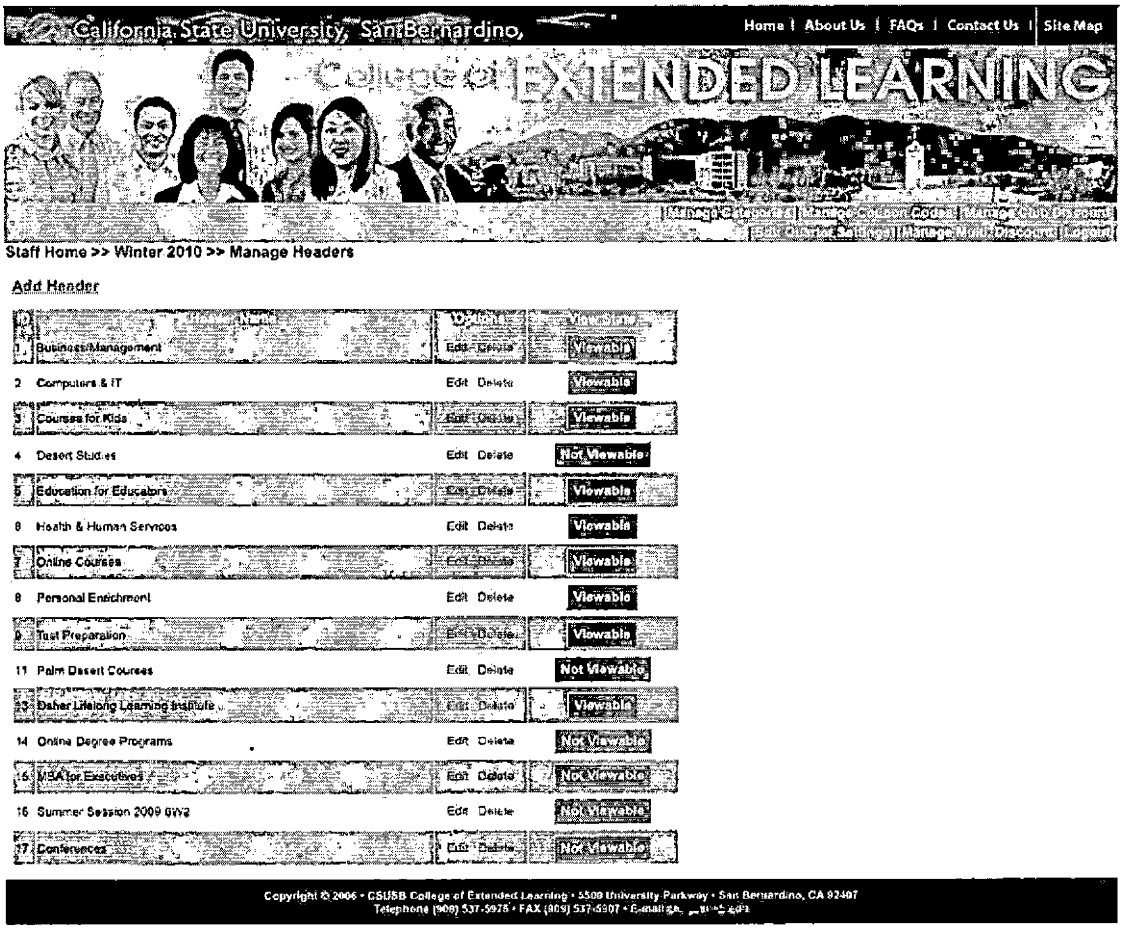


Fig. 4.21: View Header List Page

Create Header Page

This is header creation page. An empty header name or duplicate header name is not allowed. Access to this attribute is limited to course managers only (Figure 4.22).

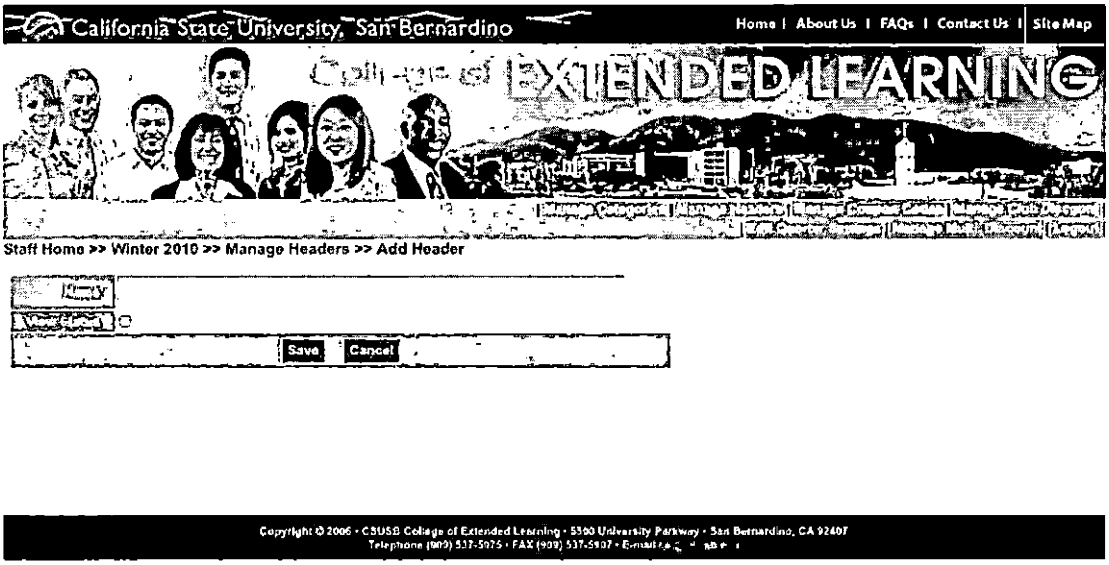


Fig. 4.22: Create Header Page

Edit Header Page

The attributes of a header can be changed through this page (Figure 4.23). Access to this attribute is limited to course managers only.

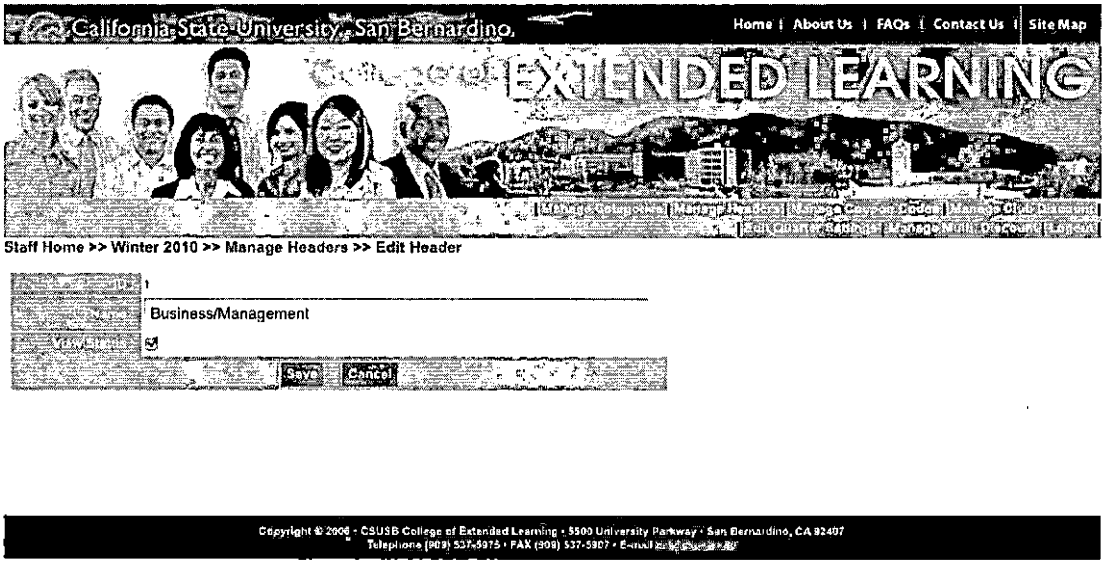


Fig. 4.23: Edit Header Page

Delete Header Page

This is a confirmation page to remind the course manager that the header record is going to be removed permanently from the database. The header cannot be deleted if it is associated with any categories. Access to this attribute is limited to course managers only (Figure 4.24).

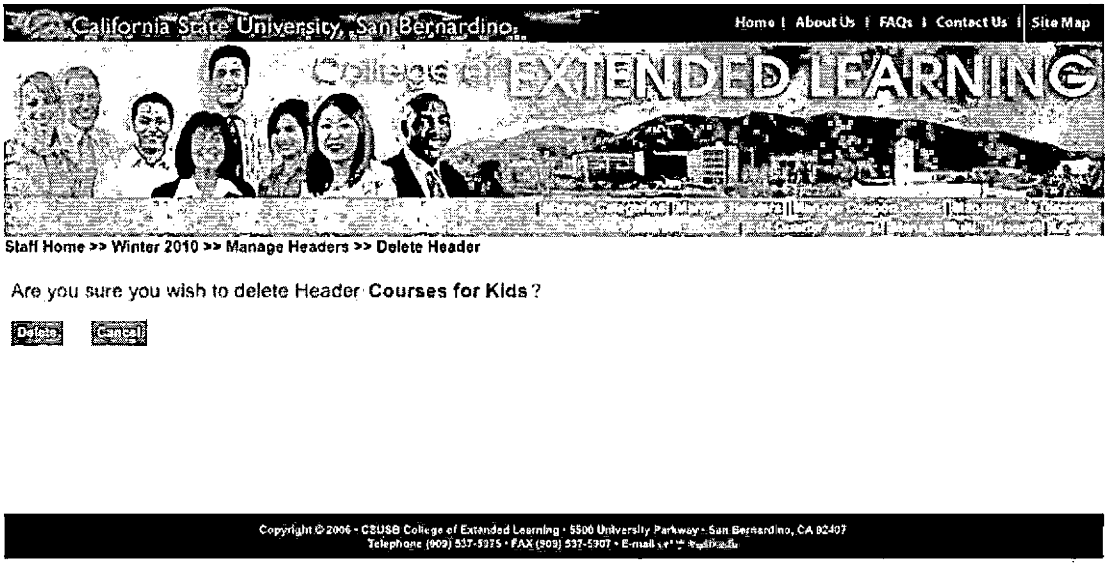


Fig. 4.24: Delete Header Page

View Discount List Page

This page (Figure 4.25) shows all offered discounts in the quarter. Access to this attribute is limited to course managers only.

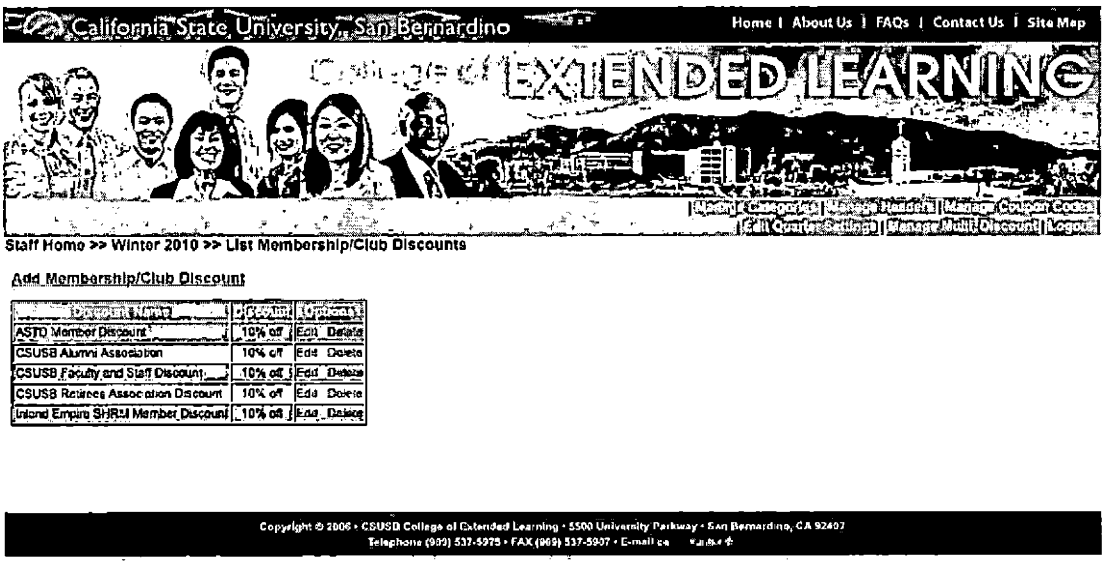


Fig. 4.25: View Discount List Page

Create Discount Page

A new discount can be generated through this page (Figure 4.26). The discount name must be unique and cannot be empty. A valid discount rate is any percentage more than 0 but less than 100. Access to this attribute is limited to course managers only.

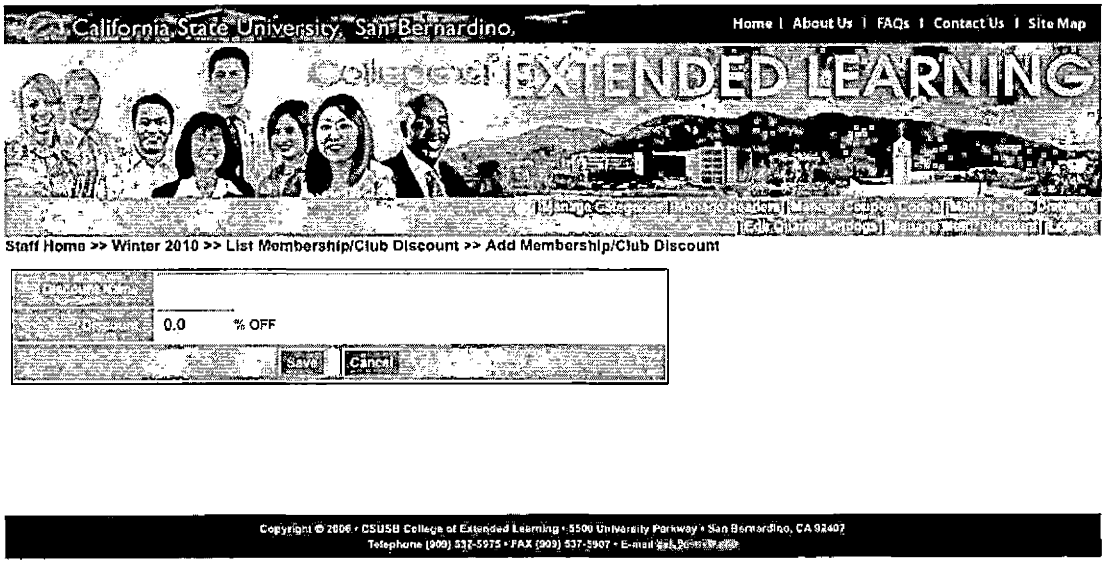


Fig. 4.26: Create Discount Page

Edit Discount Page

This page provides an interface to modify the discount rate and name (Figure 4.27). The same restrictions apply here as in creating a new discount. Access to this attribute is limited to course managers only.

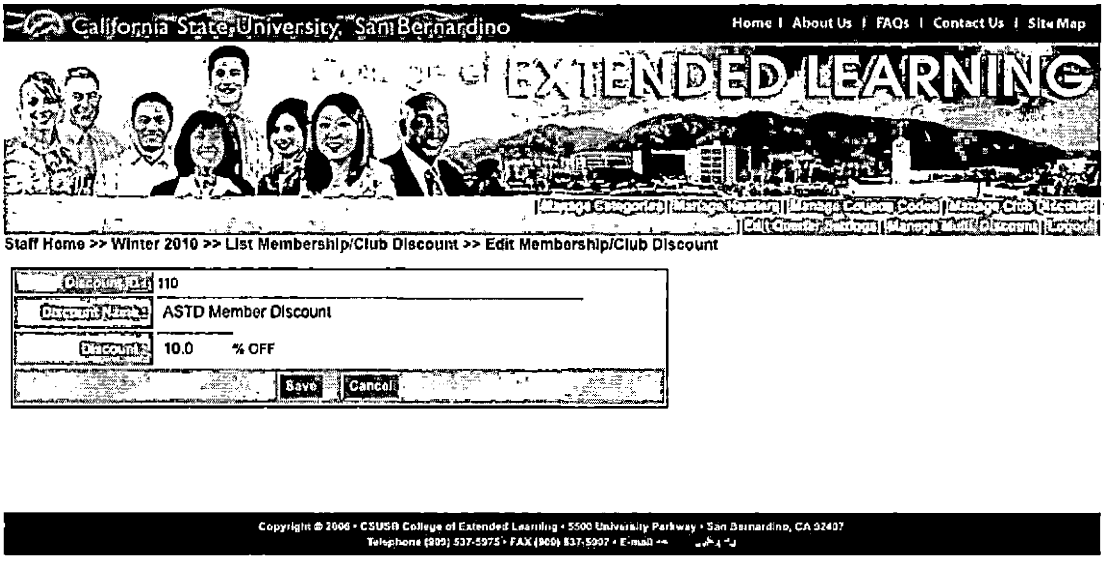


Fig. 4.27: Edit Discount Page

Delete Discount Page

This is a warning page for deleting a discount. The discount cannot be deleted if it is applied to any registrations. Access to this attribute is limited to course managers only (Figure 4.28).

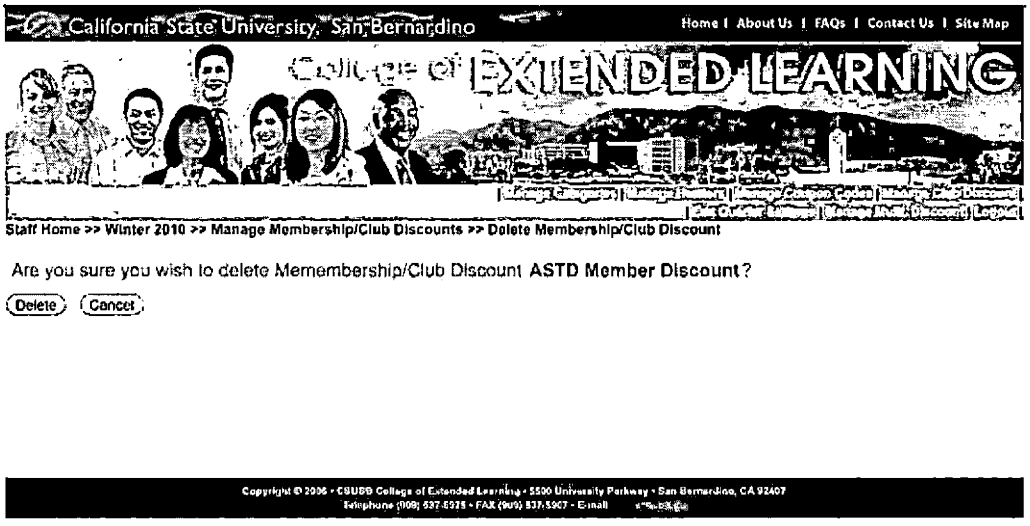


Fig. 4.28: Delete Discount Page

View Coupon List Page

This page (Figure 4.29) lists all advertising coupon code discounts for the quarter. Access to this attribute is limited to course managers only.

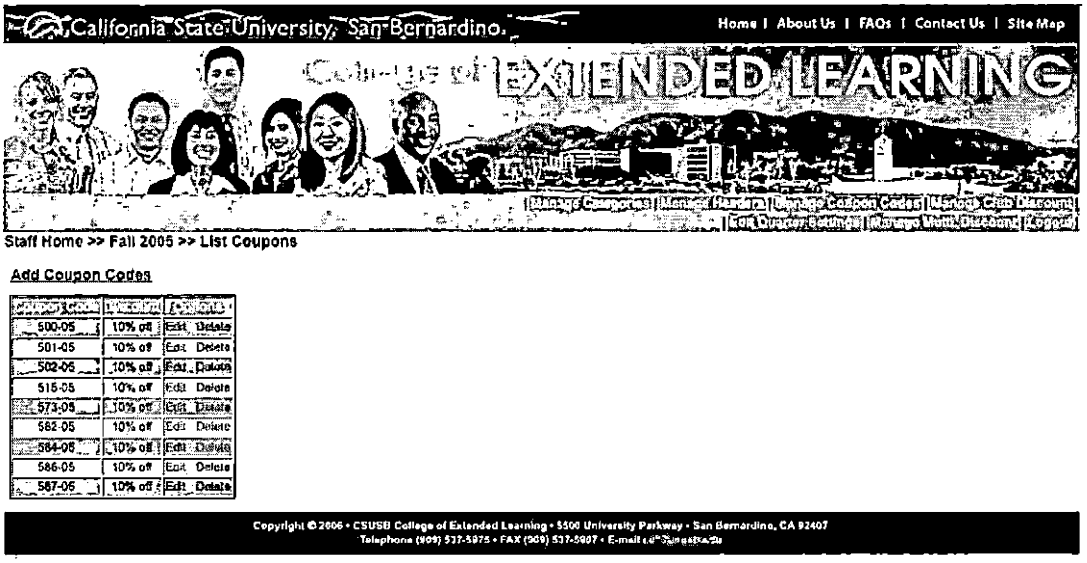


Fig. 4.29: View Coupon List Page

Create Coupon Page

A new advertising coupon code discount can be created here (Figure 4.30). The coupon code must be unique to the quarter and cannot be empty. The discount rate must be entered in numbers more than 0 but less than 100. Access to this attribute is limited to course managers only.

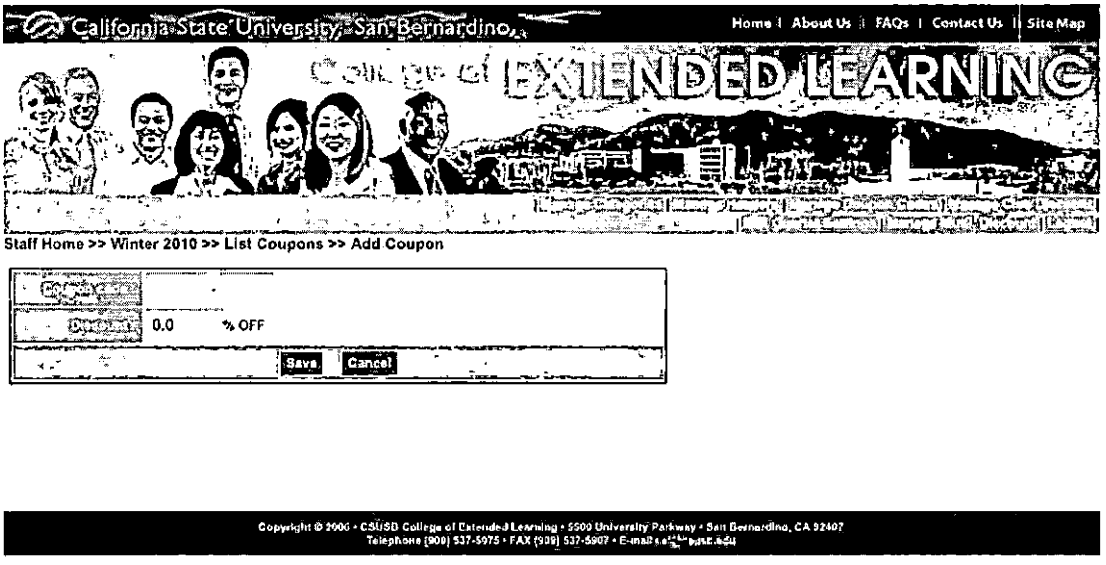


Fig. 4.30: Create Coupon Page

Edit Coupon Page

The coupon code and discount rate can be modified on this page (Figure 4.31). The same restrictions are applied as creating a coupon code. Access to this attribute is limited to course managers only.

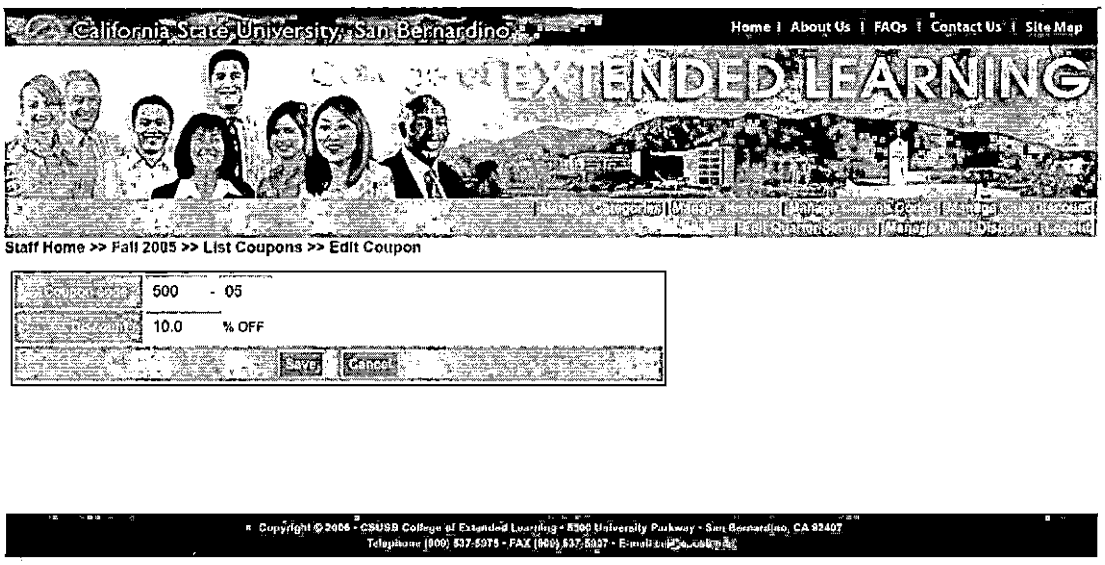


Fig. 4.31: Edit Coupon Page

Delete Coupon Page

This is a warning page (Figure 4.32) for deleting a coupon code. The coupon code cannot be deleted if used in that quarter. Access to this attribute is limited to course managers only.

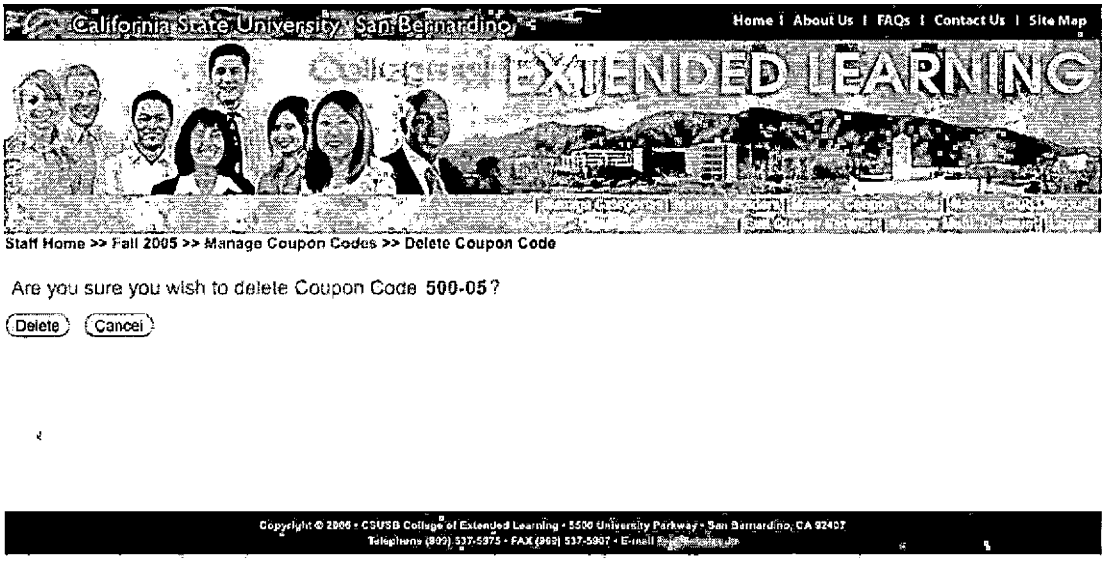


Fig. 4.32: Delete Coupon Page

View Multiple Discount List Page

This page (Figure 4.33) lists all multiple coupon programs offered during a specific quarter. Access to this attribute is limited to course managers only.

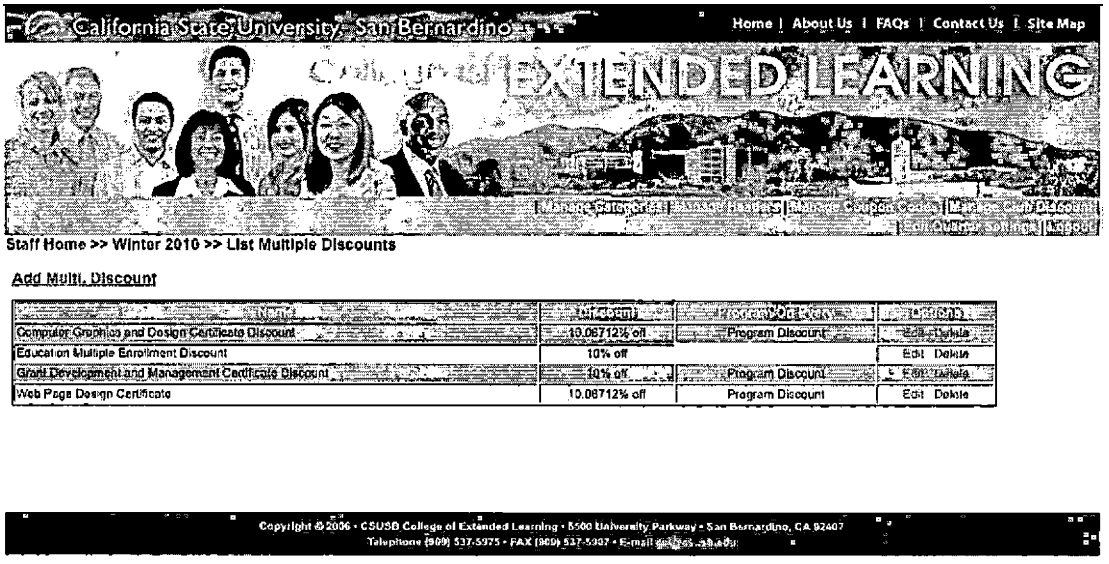


Fig. 4.33: View Multiple Discount List Page

Create Multiple Discount Page

This page is for creating a new multiple courses discount (Figure 4.34). If the "Program Discount" box is checked, two types of discount may be applied: If "Deduct on courses" is selected, the discount only applies to qualified courses which exceed the designated number. For instance, if "Deduct on courses" was set to "3", and a student registered for five qualified courses, only two qualified courses would receive the discount. If "Deduct on all qualified" is specified, the discount applies to all qualified courses as long as the minimum number of qualified courses is registered. For example, if "Deduct on all qualified" was set to 3, and a student registered for four qualified courses, all four courses would receive the discount. However, if only two qualified courses were selected, none of them would be discounted. If the "Program Discount" box is checked, a discount is applied to all courses as long as the student registers for every course in a designated program in the same quarter. Only one option can be set at the same time. Access to this attribute is limited to course

managers only.

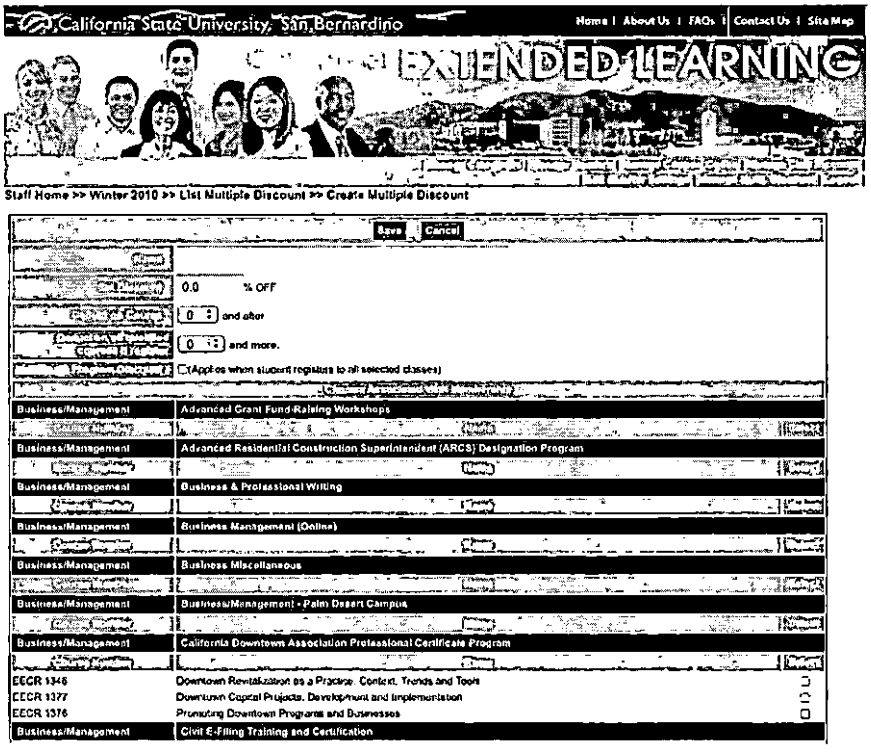


Fig. 4.34: Create Multiple Discount Page

Edit Multiple Discount Page

This is a multiple courses discount editing page (Figure 4.35). See the applied restrictions in the previous section. Access to this attribute is limited to course managers only.

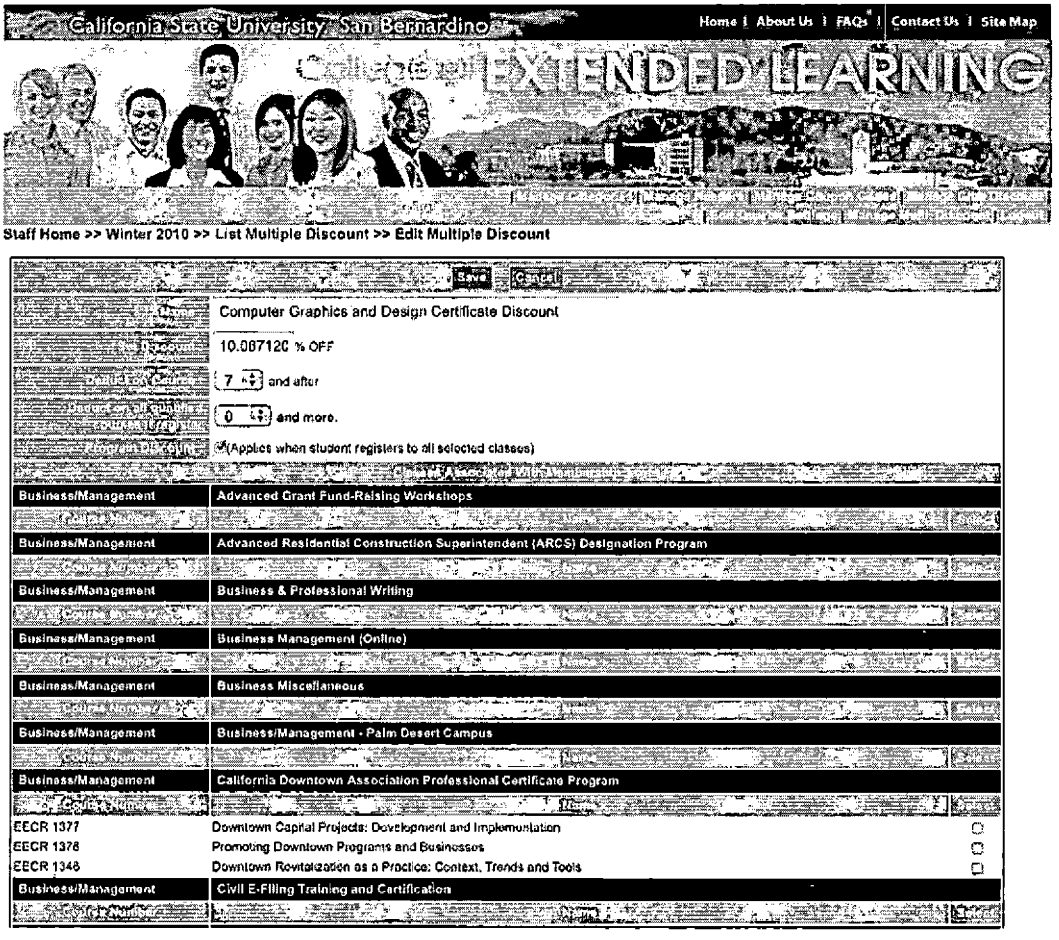


Fig. 4.35: Edit Multiple Discount Page

Delete Multiple Discount Page

This is a warning page for deleting a multiple course discount (Figure 4.36). Access to this attribute is limited to course managers only.

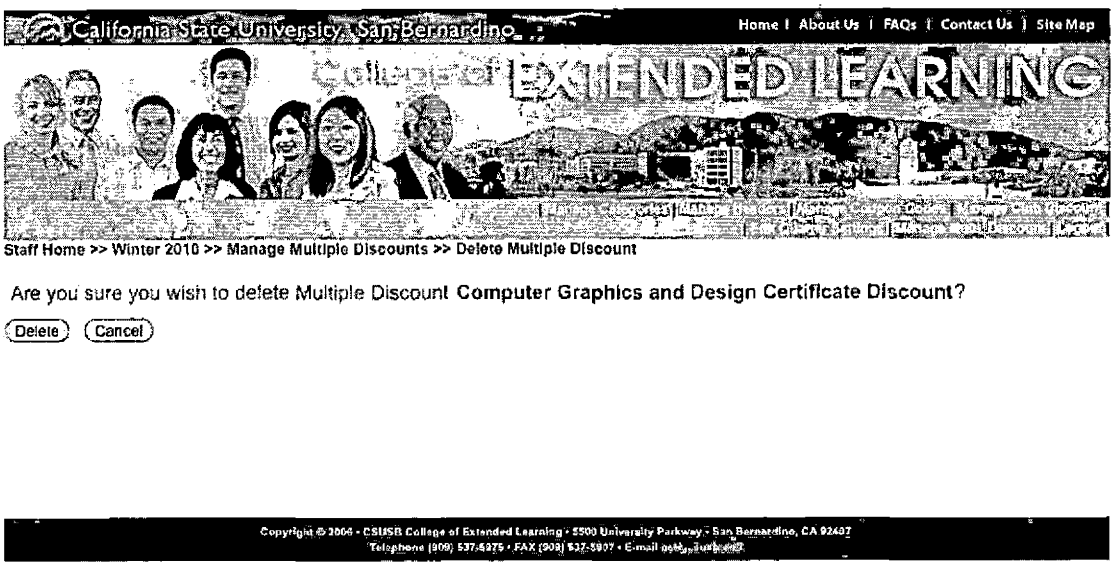


Fig. 4.36: Delete Multiple Discount Page

Search Registration Records Page

This is the starting page (Figure 4.37) when the registration manager clicks the "Process Students" button on the list quarter page (Figure 4.8). Staff can search registration records by course, first name, or last name. Those selections are optional. Staff must pick one of the mandatory selections before proceeding. The mandatory selections are pending, accepted, or rejected. Access to this page is limited to registration managers only.

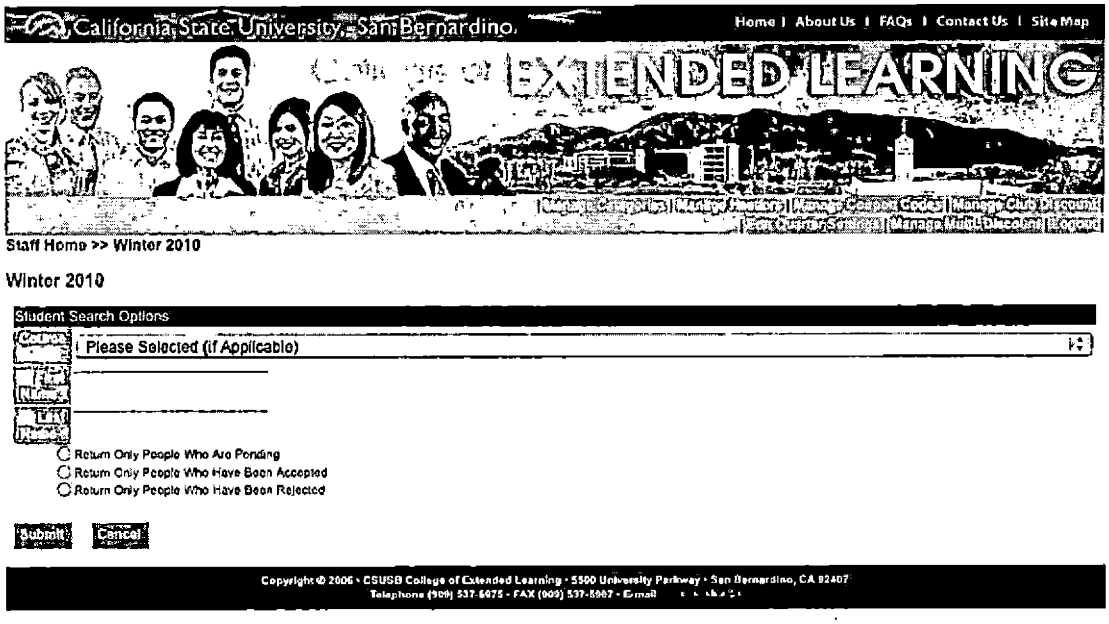


Fig. 4.37: Search Registration Records Page

List Registration Records Page

The search result based on the criteria given from the previous section (Figure 4.38). Access to this page is limited to registration managers only.

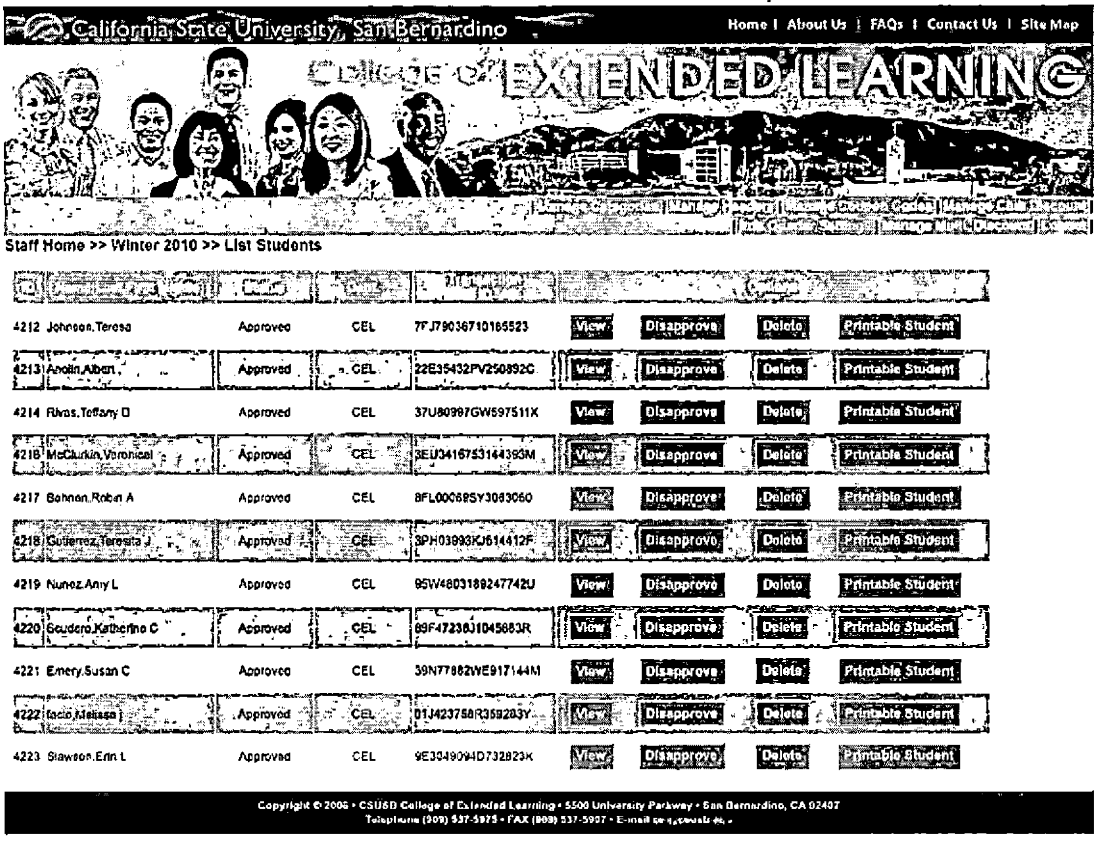


Fig. 4.38: List Registration Records Page

View Registration Record Page

This page shows the student's registration details, including marketing information and registered courses (Figure 4.39). The registration manager must verify the fees that the student payed through Paypal before approving the student's payment on the Paypal Payment Web site and approving or rejecting the registration on the CELORS. Access to this page is limited to registration managers only.

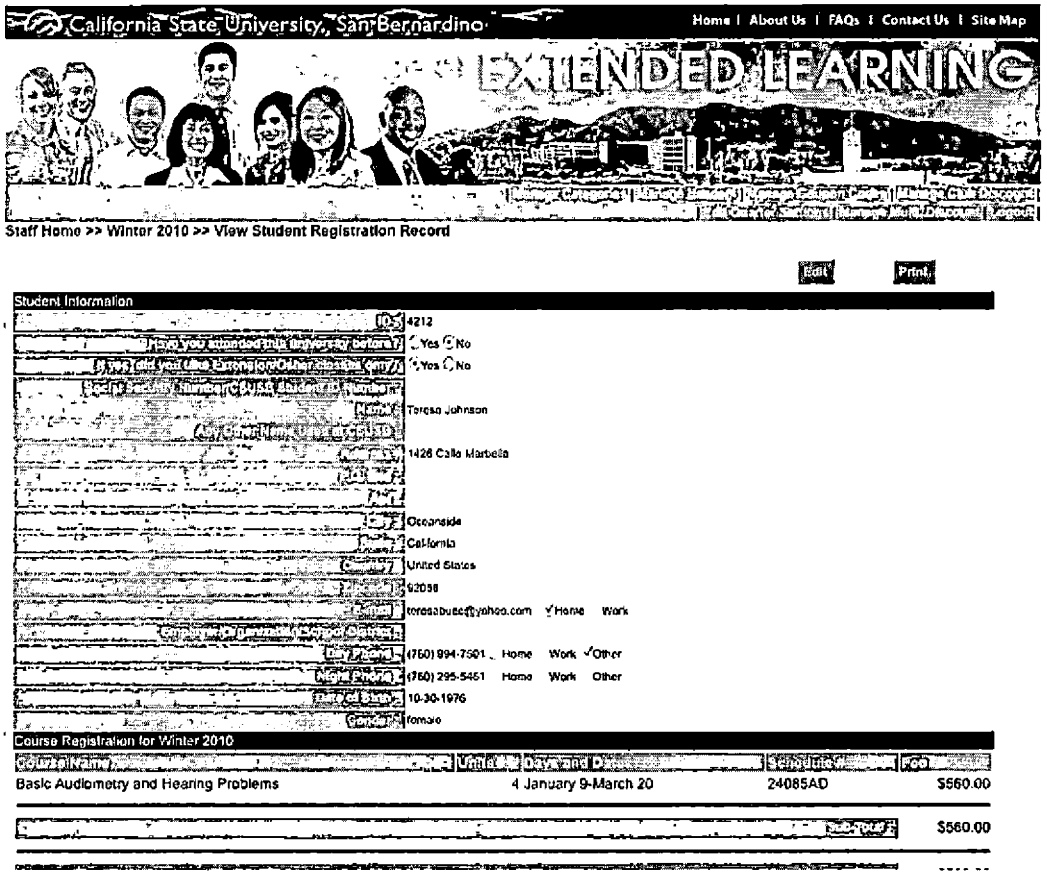


Fig. 4.39: View Registration Record Page

Edit Registration Record Page

This page is used to make changes on a student's registration record under certain circumstances (Figure 4.40). For example, if the registered course was cancelled due to low enrollments, or the student entered the incorrect daily parking pass quantity, or selected the wrong parking pass, this feature allows the CELORS staff to correct online registrations. Access to this page is limited to registration managers only.

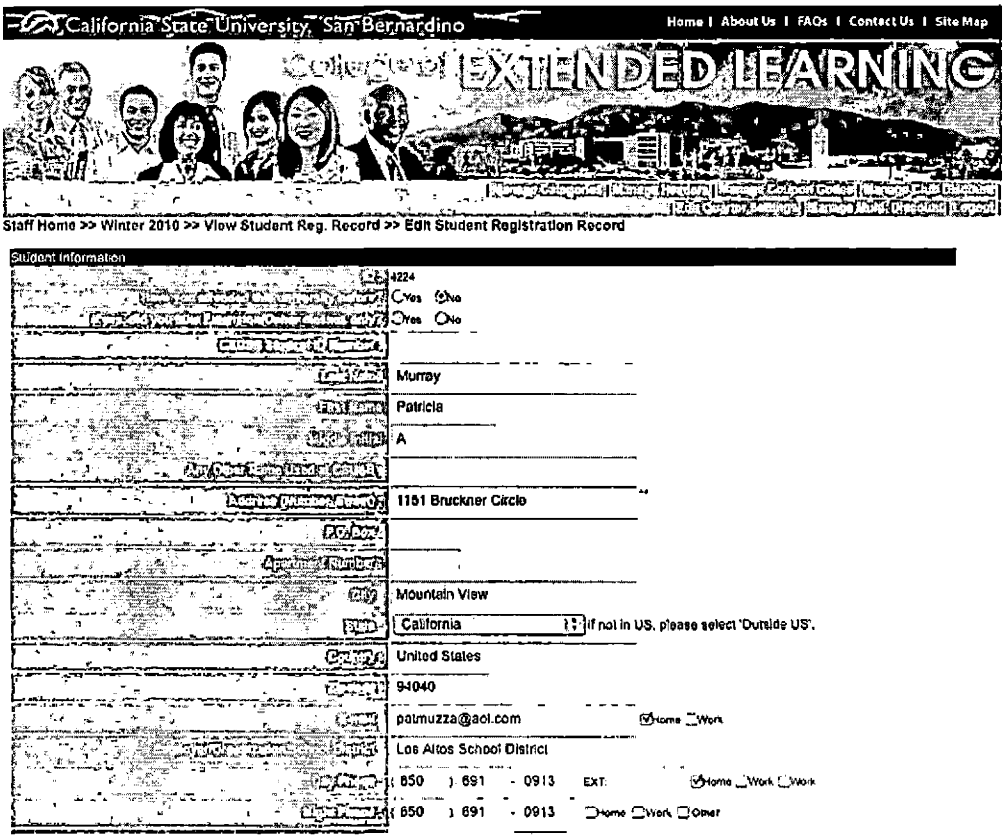


Fig. 4.40: Edit Registration Record Page

Delete Registration Record Page

This page (Figure 4.41) asks the registration manager to confirm whether the student's registration really should be removed from the database. Access to this page is limited to registration managers only.

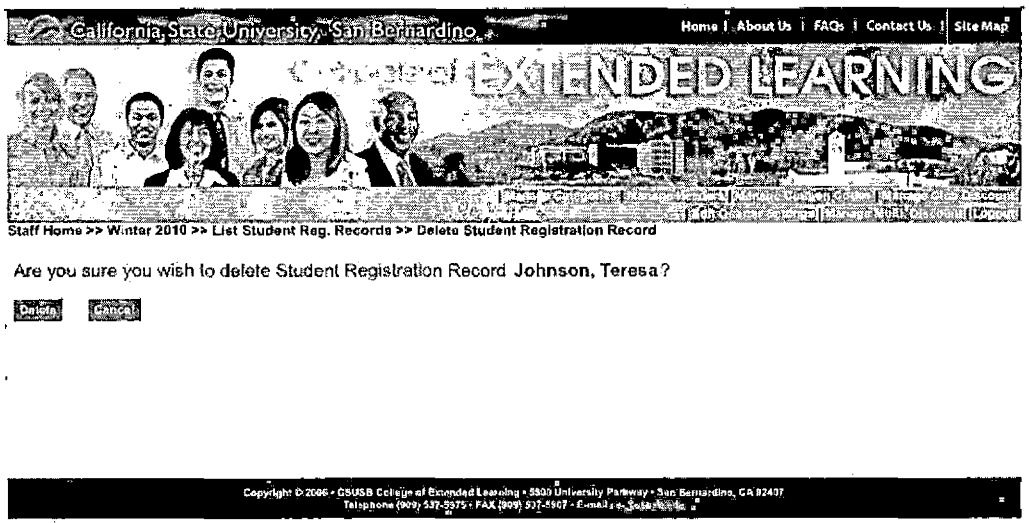


Fig. 4.41: Delete Registration Record Page

Print Registration Record Page

This is a printer-friendly version of the student registration record used for record filing purposes (Figure 4.42). Access to this page is limited to registration managers only.

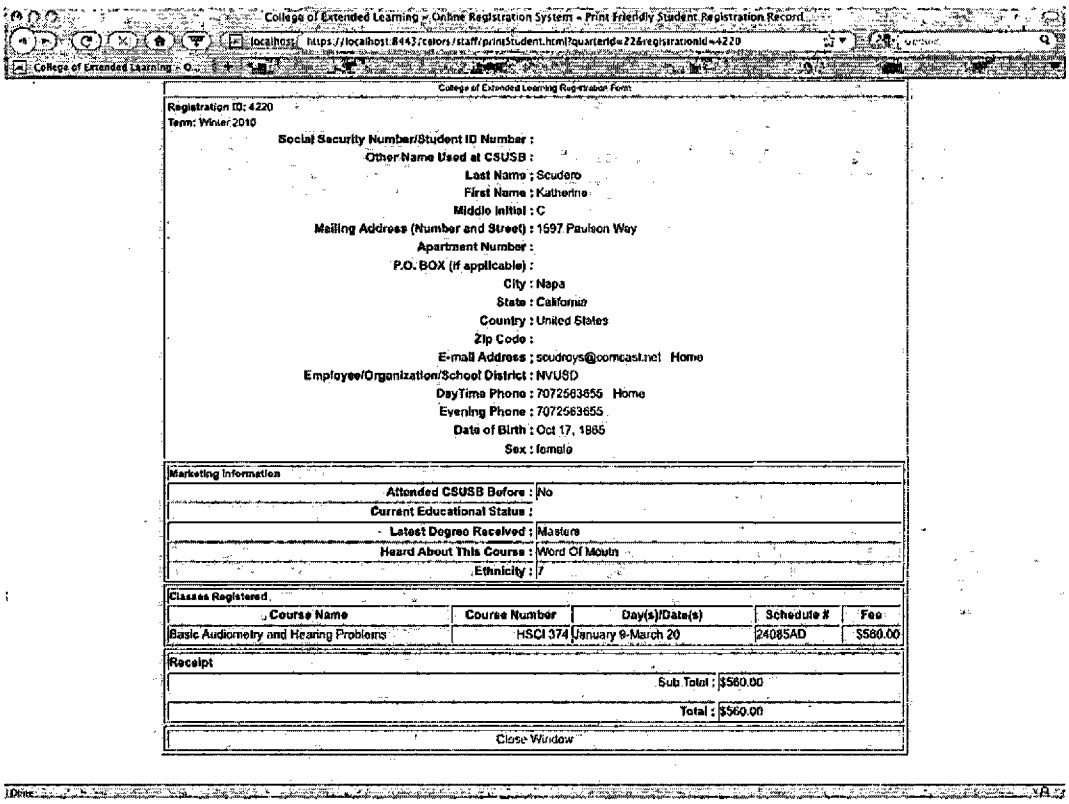


Fig. 4.42: Print Registration Record Page

Change Password Page

This page is for changing staff's own password. If the staff member forgot his/her password, he/she must contact the administrator to reset the password. This page can only be accessed once a staff member logs in successfully (Figure 4.43).

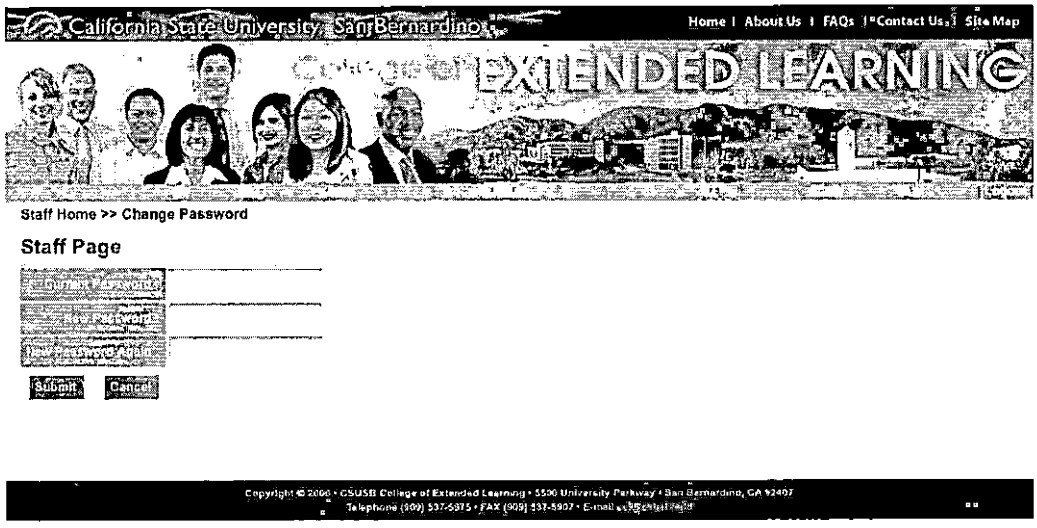


Fig. 4.43: Change Password Page - Staff

Staff Remote Login Page

This page (Figure 4.44) is used for staff to quickly edit or print a specific registration record. The link is provided by the CELORS registration notification E-mail. A valid username, password, and "process students" permission are required to access a student's record.

Fig. 4.44: Remote Login Page

4.1.3 Student Pages

Index Page

This page (Figure 4.45) is designed for accessing the online registration system by typing in "http://celors.csusb.edu/cel" directly instead of clicking the preconfigured link through the CEL Web site. All viewable quarters are shown on the "select quarter" drop-down menu.

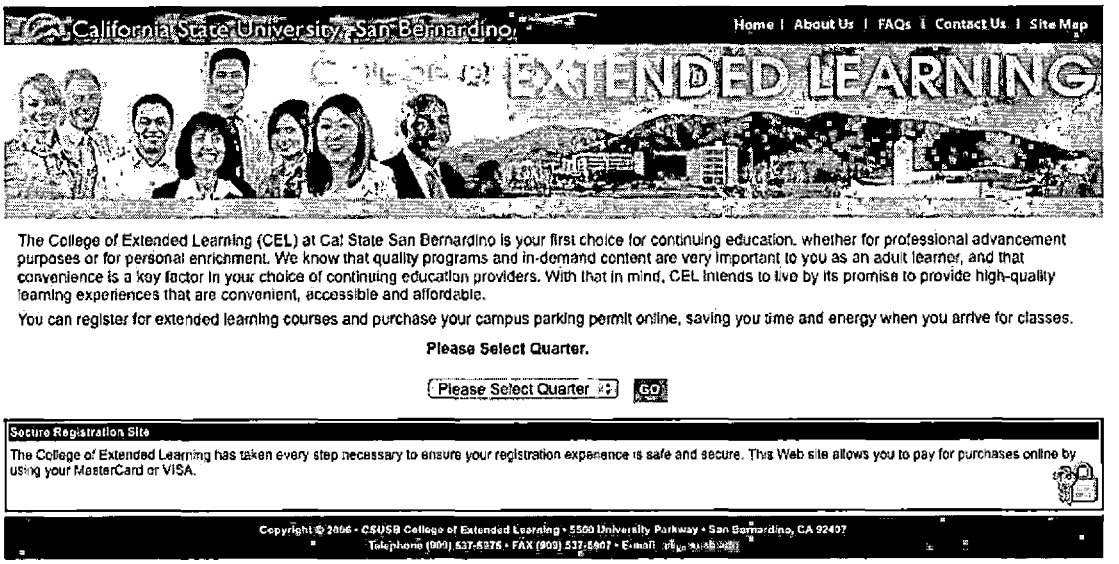


Fig. 4.45: Index Page Without Specified Quarter

View Bulletin Page

This is the page potential students are redirected to from the CEL catalog page or the previous index page. (fig 4.46).

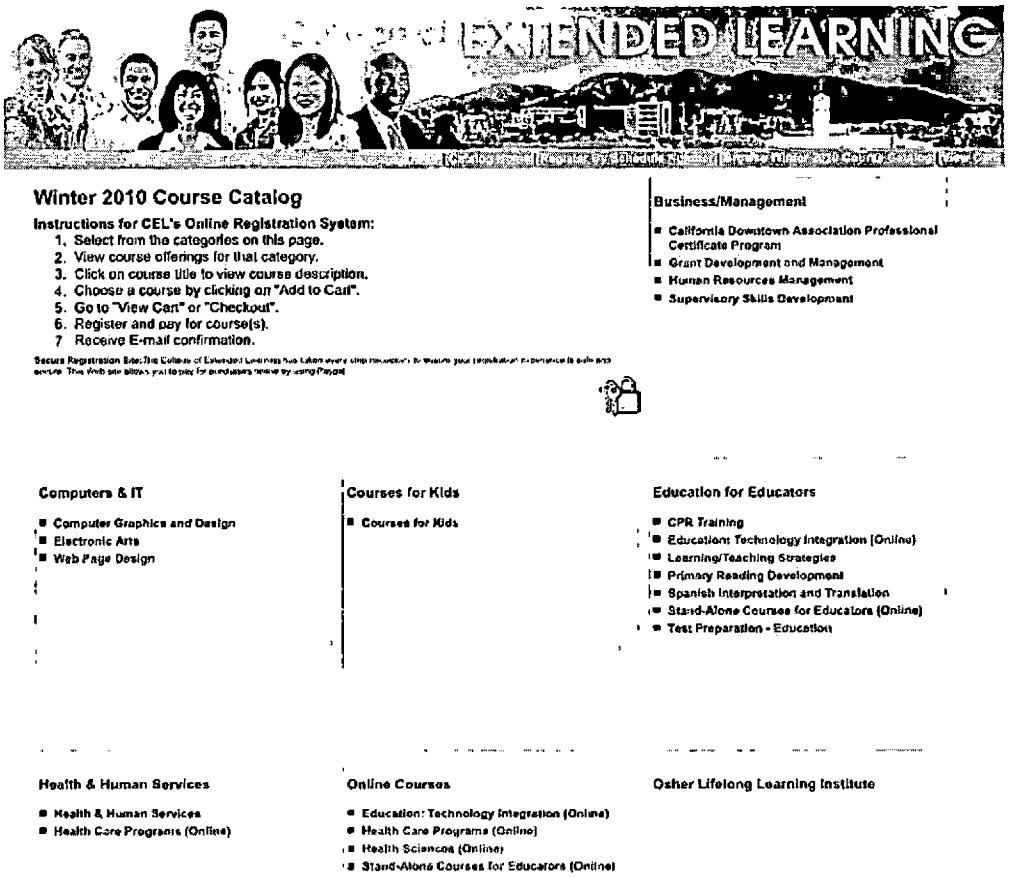


Fig. 4.46: View Bulletin Page

View Course List Page

This page shows the courses listed under a specific category. (Figure 4.47).

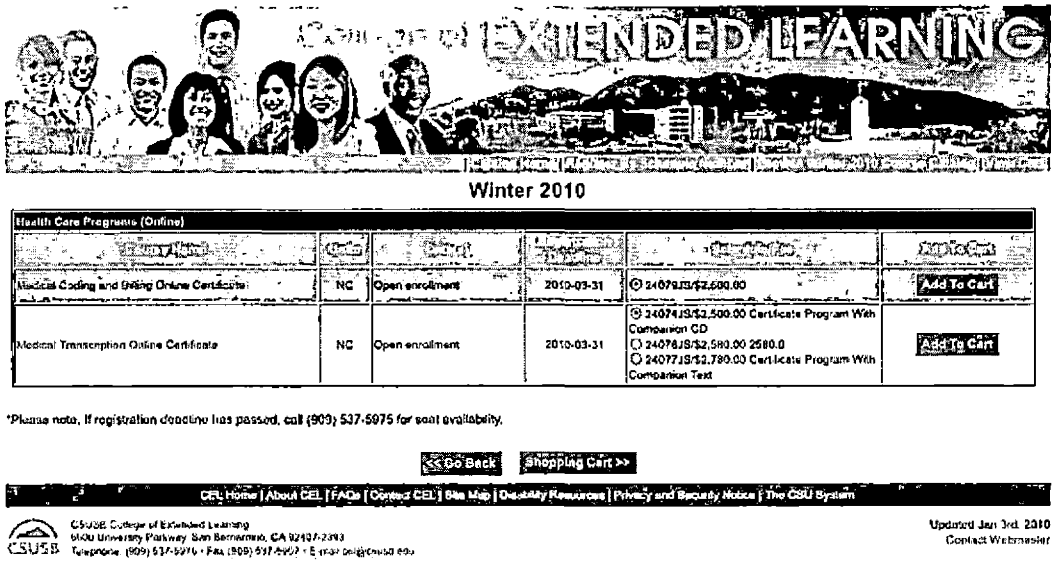


Fig. 4.47: View Course List Page

View Course Page

This page (Figure 4.48) shows the detail information of a selected course such as the course schedule, the course description, the fees, the instructor, etc.

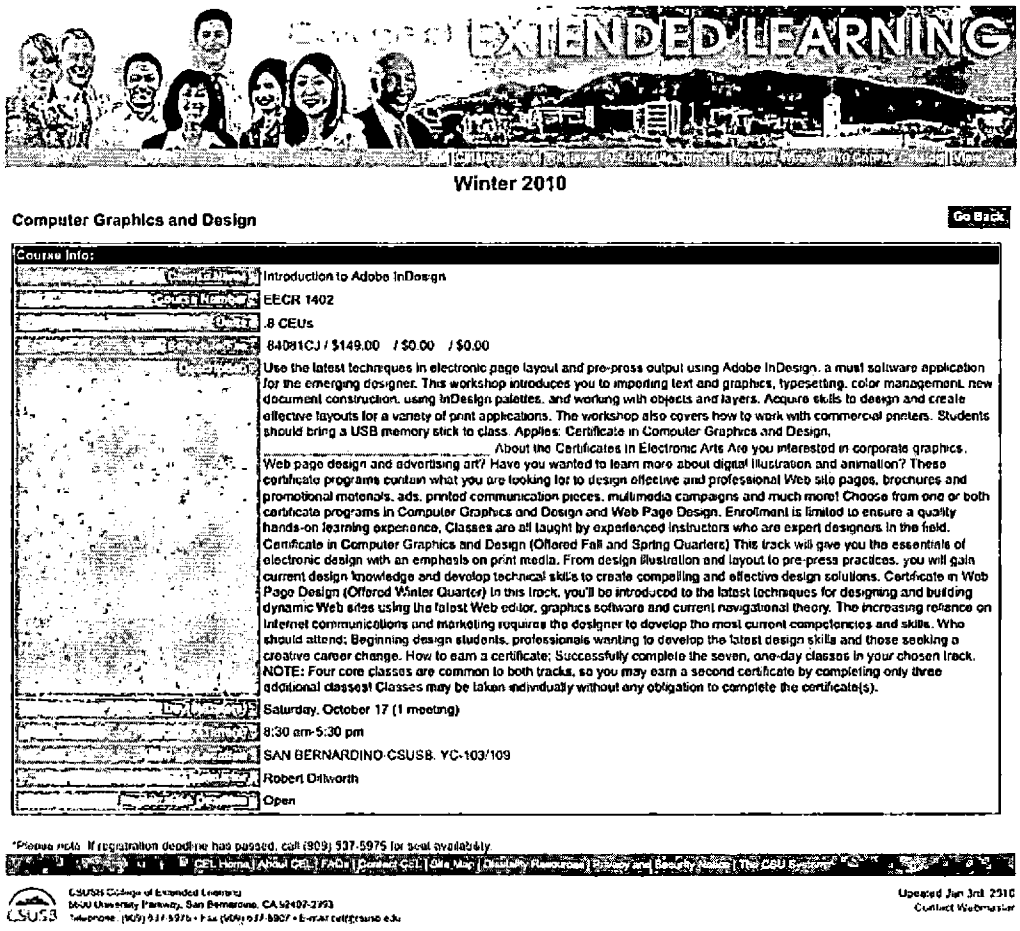


Fig. 4.48: View Course Page

Register By Schedule Number Page

This page allows a student to quickly register by directly entering a schedule number (Figure 4.49).

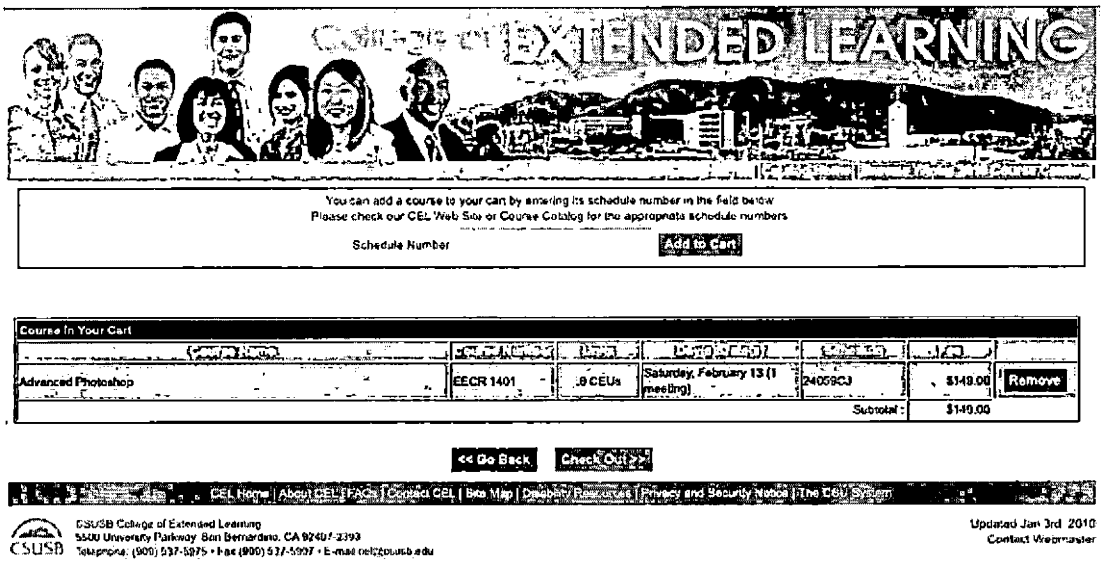


Fig. 4.49: Register By Schedule Number Page

View Cart Page

The student can review the courses selected before checking out (Figure 4.50).

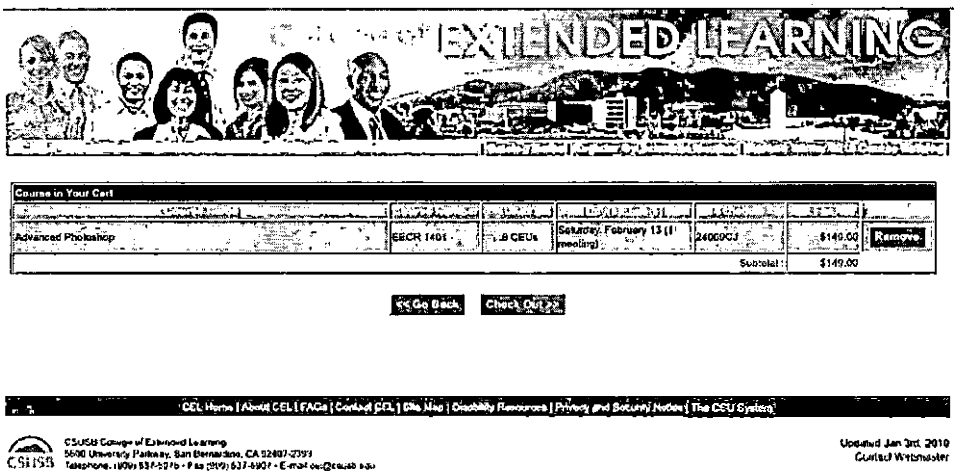


Fig. 4.50: View Cart Page

Registration Information Page

This is the first page of the registration checkout process (Figure 4.51). This page collects the personal contact information. In order to ensure that important course-related notification are delivered, entering the E-mail address two times is required.

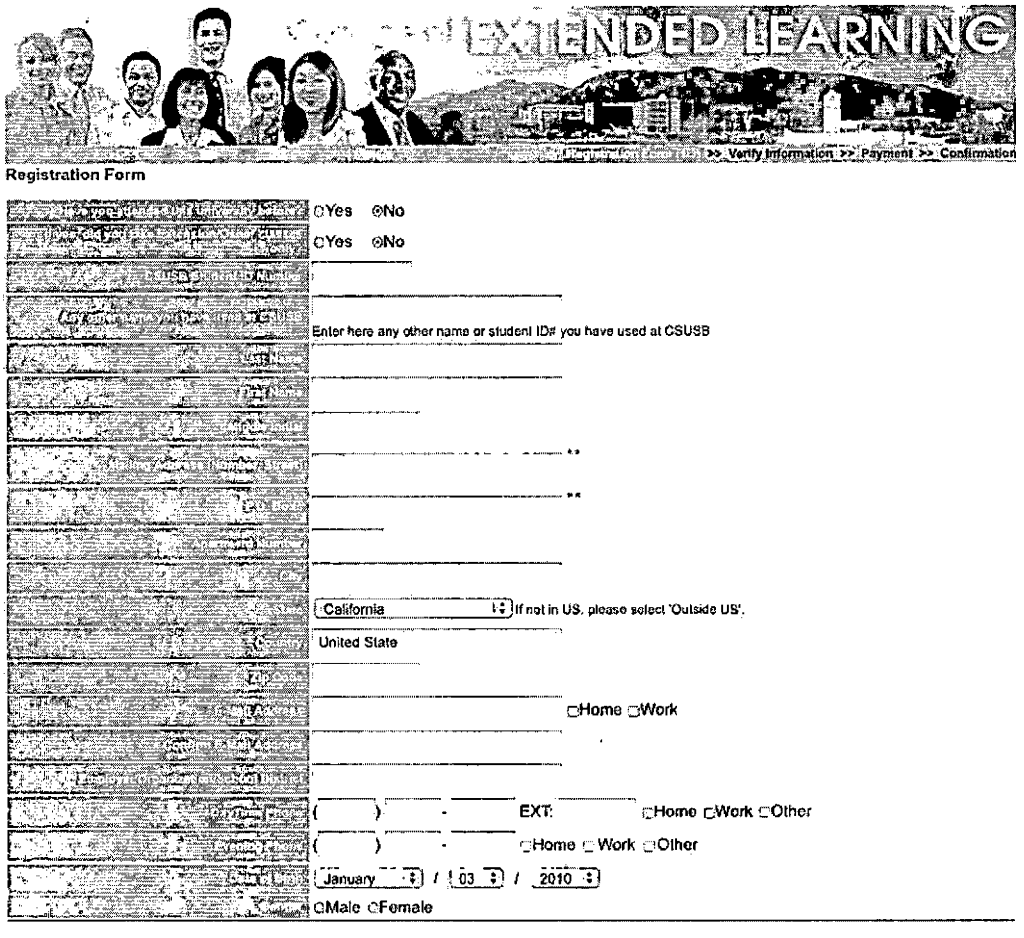


Fig. 4.51: Registration Information Page

Registration Discount/Parking Pass Page

This is the second page of the registration checkout process (Figure 4.52). The student enters a coupon or discount code, if applicable, and purchases daily or quarterly parking passes on this page.

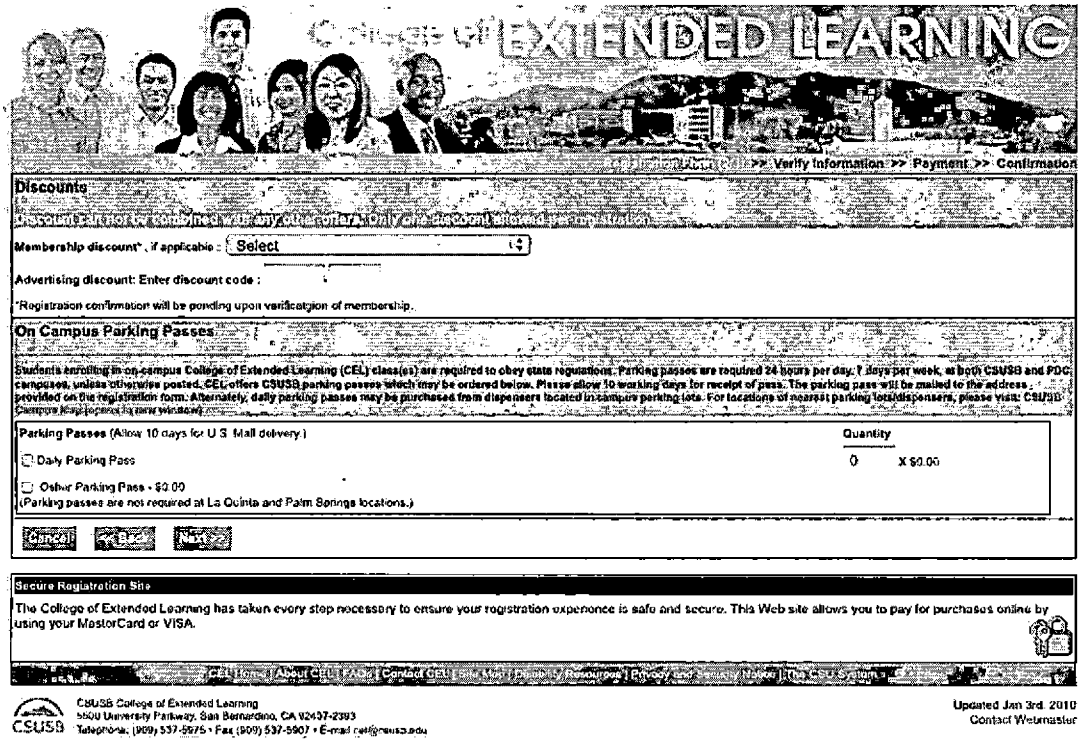


Fig. 4.52: Registration Discount/Parking Pass Page

Registration Marketing Page

This is the last registration page (Figure 4.53). This page collects marketing information for future analysis.

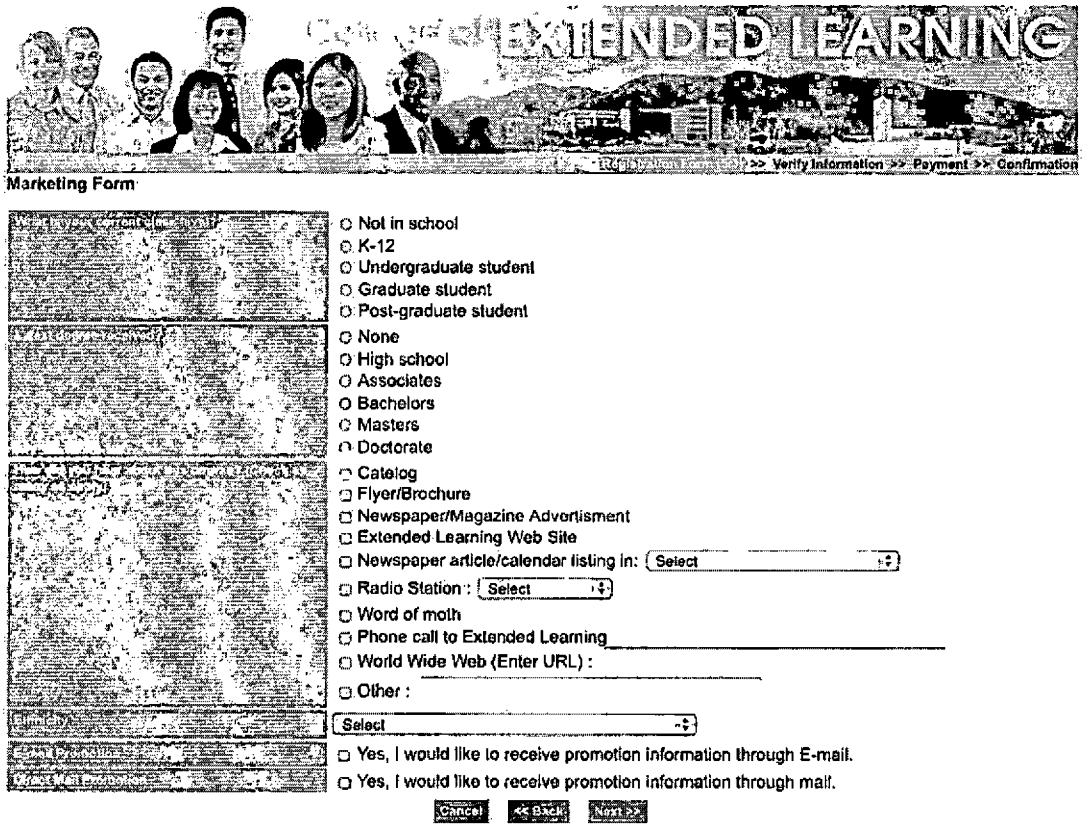


Fig. 4.53: Registration Marketing Page

Registration Review Page

This is the review page before proceeding toward the payment page (Figure 4.54). This page lists the selected courses, fees, discounts, parking fees, and the Osher membership fee, if applicable.

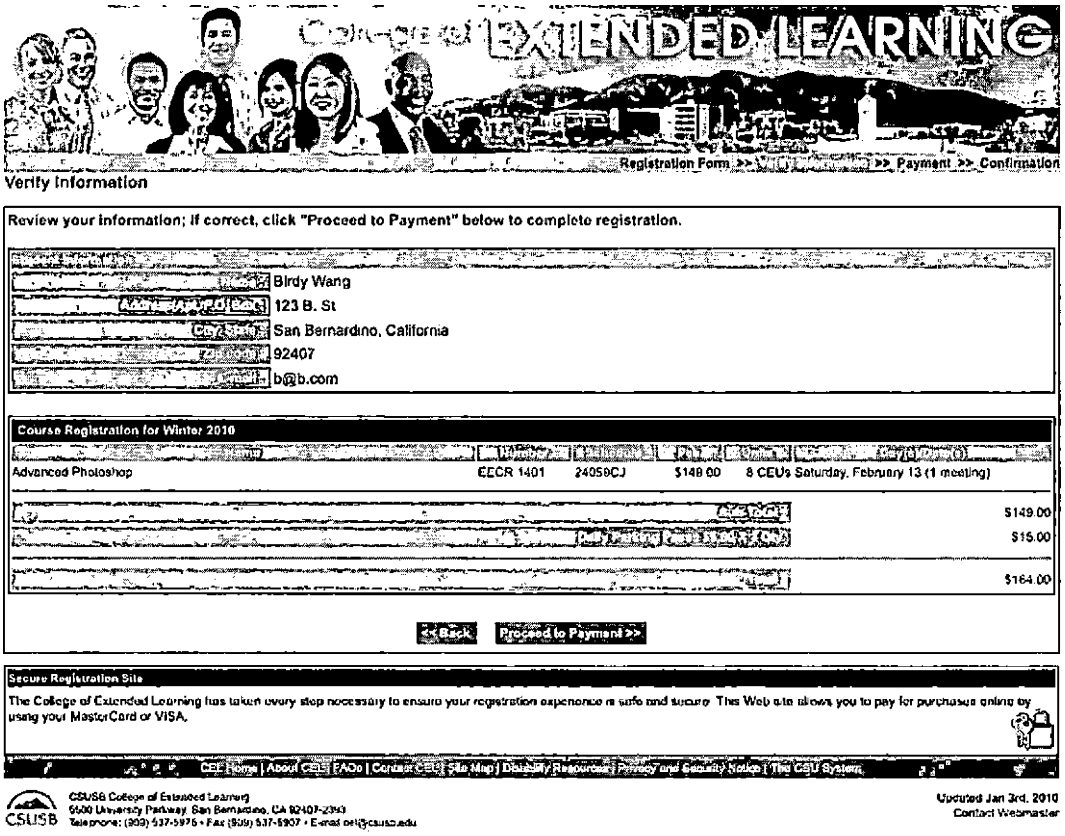


Fig. 4.54: Registration Review Page

Registration Payment Page

This page (Figure 4.55) acquires the student's payment information, then processes the payment through the Paypal payment system. The credit card holder doesn't have to be the same person as the registered student. All major credit cards are accepted such as American Express, Visa, masterCard, and Discover.

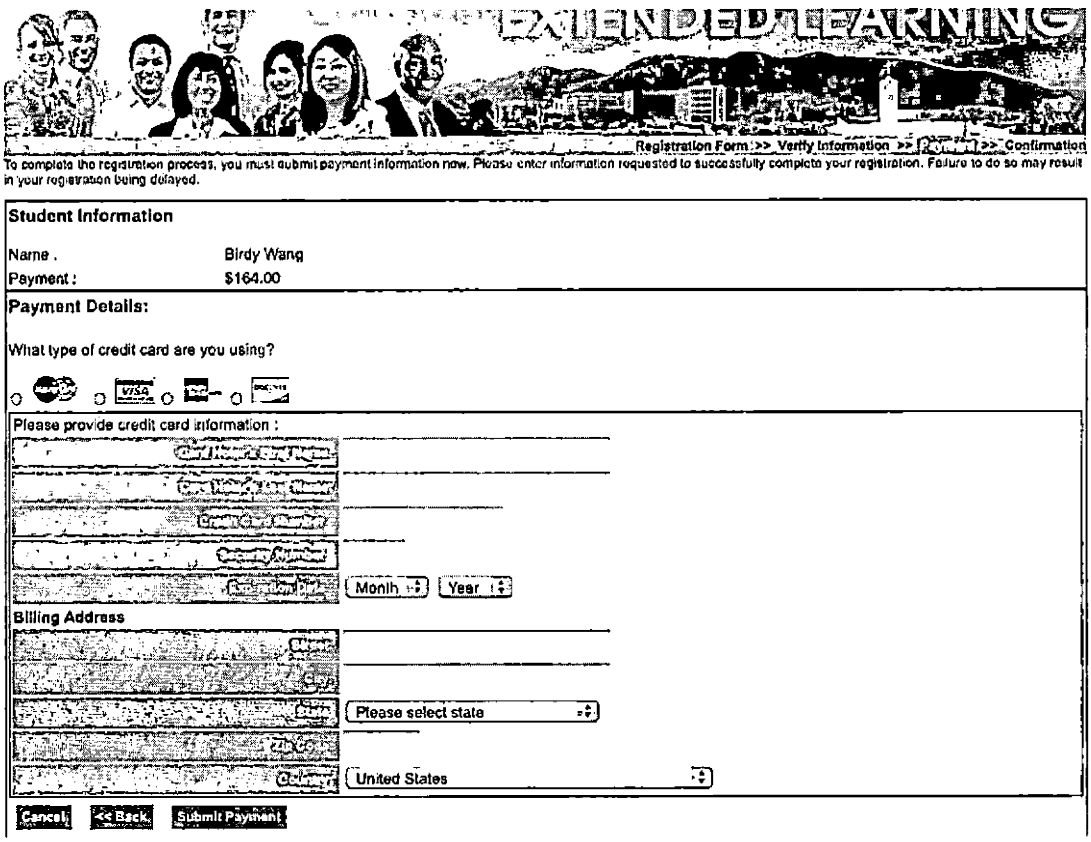


Fig. 4.55: Registration Payment Page

Registration Confirmation Page

This is the last page of the registration process (Figure 4.56). This page shows the Paypal transaction ID and other information. It is recommended that the student prints out this page for future reference.

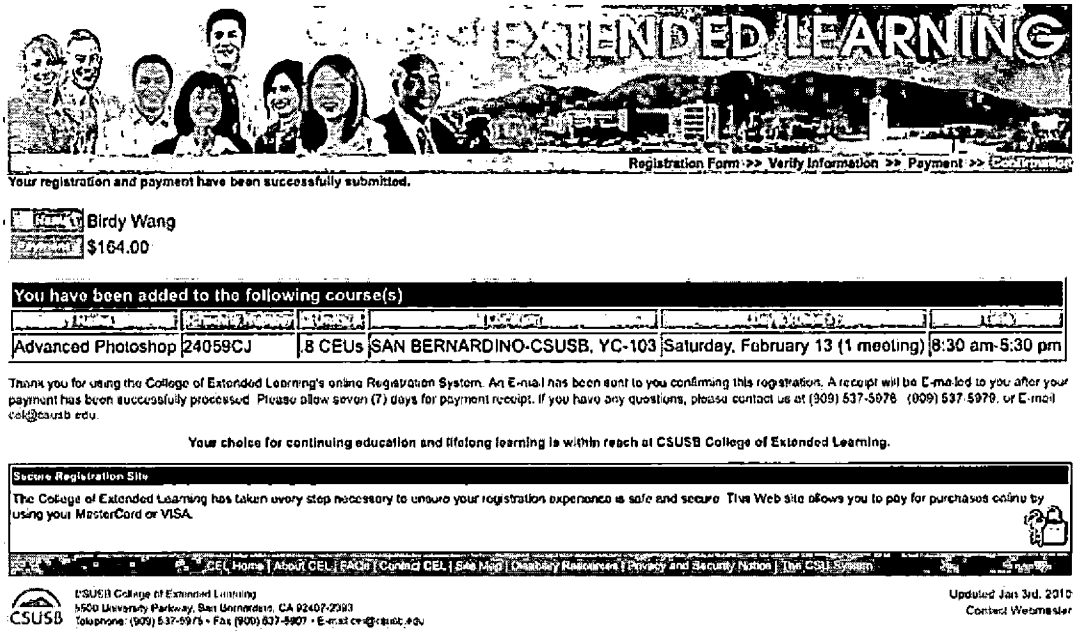


Fig. 4.56: Registration Confirmation Page

5. SOFTWARE VALIDATION

This chapter is meant to document the validation of the CELORS. The purpose of the software validation is to evaluate the attributes or capabilities of the CELORS and determine if it meets its required results. The following sections depict the results in different testing levels. The Paypal Sandbox environment was used to test payment processing.

5.1 Unit Testing

Unit testing is the first formal test activity performed in the software life cycle and it occurs during the implementation phase after each software unit is finished. Unit testing verifies the functionality of a specific section of code. Both structural (white box) testing and functional (black box) testing techniques are used in this section. The detailed results of the unit testing are shown in Table 5.1.

Tab. 5.1: Unit Testing Results

Page	Test	Result
Home	Verify that HTTPS is used Verify that it redirects to selected bulletin page	PASS
Change Password	Verify all buttons and links work properly Verify that error message displays correctly when error occurs Verify that new password has been saved correctly	PASS
Login	Verify that HTTPS is used Verify all buttons work properly Verify that error message displays correctly when error occurs Verify that page is redirecting to proper page when correct authentication is entered	PASS
List/View Page	Verify that records are displayed Verify that all links are working	PASS
Create Page	Verify all buttons and links work properly Verify that error message displays correctly when error occurs Verify that record is correctly saved	PASS
Edit Page	Verify all buttons and links work properly Verify that error message displays correctly when error occurs Verify that record is correctly saved	PASS
Delete Page	Verify all buttons and links work properly Verify that error message displays correctly when error occurs Verify that record is correctly removed	PASS
Paypal direct payment	Verify that error code is send back from Paypal when invalid data is given Verify that required Paypal information is configured	PASS

5.2 System Testing

The CELORS is divided into three subsystems: the administrator, staff, and student subsystems. The system testing is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. Table 5.2 shows the system testing results by mixing GUI software testing, error handling testing, and accessibility testing techniques.

Tab. 5.2: System Testing Results

System	Test	Result
Administrator	Verify that only authenticated personnel can access this system Verify that administrator can create, edit and delete staff accounts Verify administrator can change his/her own password	PASS
Staff	Verify that only authenticated personnel can access this system Verify that staff can only access authorized subsystem Verify that course manager can manage courses, categories, etc Verify that registration manager can manage student registration records Verify that payment record can be seen on Paypal Payment Web site with correct amount Verify that designated staff receive notification by E-mail	PASS
Student	Verify that student can register and process payment Verify that student can register through quick registration Verify that registration fee is calculated and charged correctly Verify that program discount or membership discount is deducted correctly Verify that student receives confirmation and notification E-mails	PASS
Accessibility	Compliant with CSUSB Web Accessibility regulations	PASS

5.3 System Integration Testing

This testing mainly verifies that the CELORS is integrated into the Paypal payment system with Paypal API and other third party systems such as MySQL and Apache Tomcat Web server. Table 5.3 shows the testing results.

Tab. 5.3: System Integration Testing

System	Test	Result
MySQL	Installation & Start up	PASS
Tomcat	Installation & Start up	PASS
Paypal API	Set up Paypal Payment Pro account Request for Paypal API username and password Set up API environment Process payment with testing accounts	PASS

6. MAINTENANCE MANUAL

System maintenance plays another important role in the CELORS. Without a well maintained and stable Web and database server, the College of Extended Learning can not provide a reliable and 24/7 online registration system for students. This following sections contain instructions for setting up and managing the production Web application server and all necessary applications for running the CELORS.

6.1 Operation System Installation

As mentioned in Chapter One, the CELORS is a 24/7 Web application. Therefore, setting up a stable and secure server was our target. Since CentOS is a fully functioning, stable and free enterprise distro, it was chosen to be the operating system running on CEL's production server.

6.1.1 Downloading and Installing CentOS

The easiest installation method is to download the CentOS ISO and burn it to a CD/DVD. The latest ISO image can be obtained from the CentOS official Web site [15].

insert the first CD or DVD and boot up. Ensure your BIOS is set to boot from the CD/DVD before any other bootable devices such as hard drives, USB flash drives, or SCSI RAID card.

After a short delay, a screen says "boot:" prompt should appear. To trigger the installation, type:

```
linux text
```

The Welcome screen will appear. Follow the instructions to install in English with a U.S. English keyboard and select "custom" installation.

6.1.2 Disk Partition Setup

Select Disk Druid as the method of setting up the disk.

Delete any existing partitions and create the partition as shown in Table 6.1. Select the GRUB boot loader.

Tab. 6.1: Disk Partition Specification

	Mount Point	Filesystem Type	Size
1	/boot	ext2	100 MBs
2		swap	2 times of the RAM size
3	/	ext3	Fill all available space

6.1.3 Network Configuration

Uncheck the "Configure using DHCP" box. Type in the IP address, netmask, gateway, and DNS settings as shown in Table 6.2.

In the Hostname Configuration screen, type in "celors.csusb.edu".

If the production server has more than one network card, only set up eth0 as mentioned above. For the others, check "Configure using DHCP" and make sure that "start on boot" is unchecked.

Tab. 6.2: Network Configuration

IP:	192.168.111.2
Netmask:	255.255.255.0
Gateway:	192.168.111.1
Primary DNS:	139.182.2.1
Secondary DNS:	139.182.2.6

6.1.4 Others

In the Security Enhanced Linux screen, disable Security Enhanced Linux.

In the Time Zone Selection screen, set the time zone to "America - Los Angeles".

Set a strong root password.

Finally, deselect everything in the package selection and click "Ok" to complete the installation.

Once completed, remove the CD/DVD and reboot.

6.1.5 Post Installation

After rebooting, login as "root". There are a few more steps that need to be completed before starting to configure the CELORS:

System Time Setup

Set the system time from the government time server and write the new time into the hardware clock as below:

```
rdate -s time.nist.gov
hwclock -w
```

Create file /root/netfilter.sh with the following contents:

```
# Flush all rules
iptables -F
# Set default policies of all chains. all packets are accepted.
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P INPUT ACCEPT
# Accept all packets from localhost
iptables -A INPUT -i lo -j ACCEPT
# Accept udp packets for DNS
iptables -A INPUT -p udp --sport 53 -j ACCEPT
# Drop all other upd packets
iptables -A INPUT -p udp -j DROP
# Accept all tcp packets for public services
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
# Reject all other tcp initialization packets
iptables -A INPUT -p tcp --syn -j REJECT
# Accept all other packets
```

Make the "netfilter.sh" executable.

```
$ chmod 700 /root/netfilter.sh
```

Run "netfilter.sh", save the state of iptables, and start the iptables service.

```
$ /root/netfilter.sh
$ iptables-save > /etc/sysconfig/iptables
$ service iptables start
```

Add the following line to "/etc/rc.d/rc.local" to run the netfilter script at bootup:

```
$ /root/netfilter.sh
```

Update Installed Packages

Execute the following commands to update the kernel and packages installed from the CD and reboot after updated:

```
$ yum -y update
$ shutdown -r now
```

Create User Account

Create a regular user with "sudo" privilege by adding user to "wheel" group (No space between "," and "wheel").

```
$ useradd <username> -G users,wheel
$ passwd <username>
```

Run "visudo" command and add the following line to allow members of the wheel group to become root without entering password:

```
%wheel ALL=(ALL) ALL
```

For extra security, adding the following line in "/etc/ssh/sshd_config" to disable root remote login:

```
PermitRootLogin no
```

Restart sshd service.

```
$ service sshd restart
```

Setup Daily Cron Task

Create "/root/daily.sh" file and add the following contents:

```
#!/bin/bash
echo Running daily.sh for CELORS server
echo
/usr/bin/rdate -s time.nist.gov
/sbin/hwclock -w
/usr/bin/locate -u
/usr/bin/yum -y update
/sbin/shutdown -r 10
```

Make the file executable. Create "/root/cronfile" file with following content:

```
27 2 * * *      /root/daily.sh
```

Register cron task with command:

```
$ crontab /root/cronfile
```

Verify that the "crond" daemon is set to run at boot with the following command:

```
$ chkconfig --list crond
```

If it shows all off, then execute the command to turn on crond:

```
$ chkconfig --add crond
$ service crond start
```

6.1.6 Java Installation

In this section, the Java Development Kit (JDK) is going to be installed.

Download and Installation

The latest Java Development Kit (JDK) version can be found from the Java official Web site [12]. Select the Linux RPM as a self-extracting file.

Make the file executable and execute it.

```
$ chmod +x jdk-<version>-linux-i586-rpm.bin
$ ./jdk-<version>-linux-i586-rpm.bin
```

Press space bar to view the copyright terms. At the end, type **yes** to agree to the terms and finish installation.

Configuration

Set up the Java environment variables. Create **/etc/profile/java.sh** file with following contents:

```
export JAVA_HOME=/user/java/latest
export PATH=$PATH:$JAVA_HOME/bin
```

Make the file executable.

```
$ chmod +x /etc/profile.d/java.sh
```

6.1.7 Tomcat Installation

Apache Tomcat (or Jakarta Tomcat or simply Tomcat) is a Web container developed at the Apache Software Foundation. Tomcat implements the servlet and the Javaserver Page(JSP) specifications, preceding an environment for Java code to run in cooperation with a Web server. Tomcat requires Java Runtime Environment(JRE) to run. Therefore, make sure you have installed and configured JDK by following the instructions mentioned in Chapter 6.1.6.

Download and Installation

Download the most recent stable Tomcat in "tar.gz" format from the Tomcat Official Web site (see Figure 6.1). Make sure the Tomcat version you download supports the Java version you installed earlier.

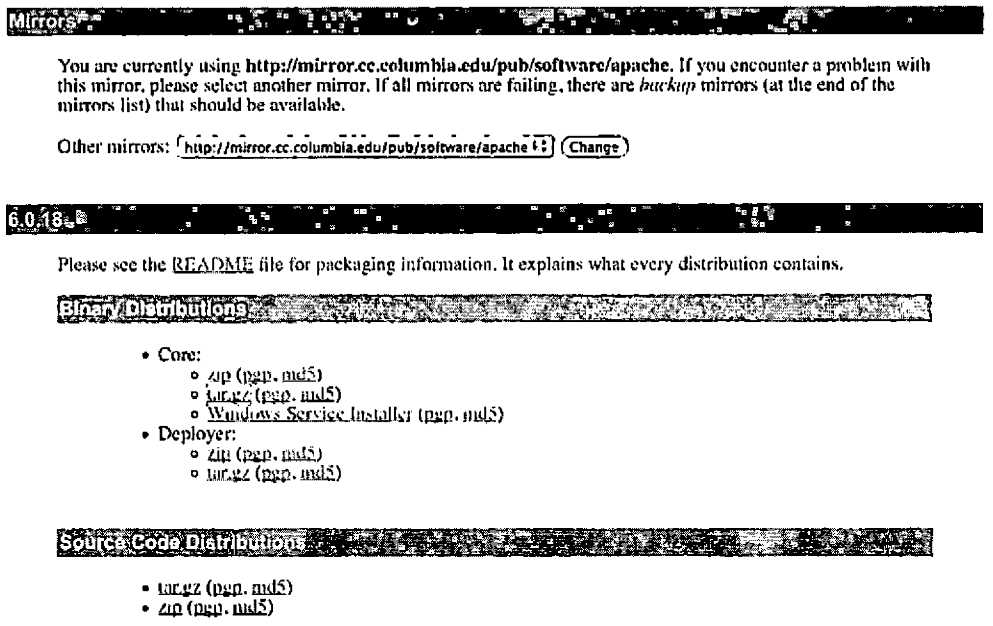


Fig. 6.1: Download The Latest Tomcat Package

After download, run the following commands to install Tomcat under "/opt" directory:

```
$ cd /opt
$ tar -zxvf apache-tomcat-VERSION.tar.gz
$ ln -s apache-tomcat-VERSION tomcat
```

Configuration

Create "/etc/init.d/tomcat" with contents as shown below:

```
#! /bin/sh
#
# tomcat Starts tomcat
#
# chkconfig: 2345 98 02
# description: tomcat is a J2EE web application container.
```

```
#
. /etc/init.d/functions
. /etc/profile.d/java.sh
export TOMCAT_HOME=/opt/tomcat
[ -f ${TOMCAT_HOME}/bin/startup.sh ] || exit 0
[ -f ${TOMCAT_HOME}/bin/shutdown.sh ] || exit 0
set -e
case "$1" in
      start)
    echo -n "Starting Tomcat..."
    $TOMCAT_HOME/bin/startup.sh >> /var/log/tomcat 2>&1
    echo "Started."
    ;;
  stop)
    echo -n "Stopping Tomcat..."
    $TOMCAT_HOME/bin/shutdown.sh >> /var/log/tomcat 2>&1
    sleep 1
    rm -f $TOMCAT_HOME/logs/*
    echo ''Stopped.''
    ;;
  restart|force-reload)
    echo -n "Restarting Tomcat..."
    $TOMCAT_HOME/bin/shutdown.sh >> /var/log/tomcat 2>&1
    sleep 1
    $TOMCAT\_HOME/bin/startup.sh >> /var/log/tomcat 2>>&1
    echo "Restarted."
    ;;
  *)
    N=/etc/init.d/tomcat
```

```
    echo "Usage: $N {start|stop|restart}" >&2
    exit 1
    ;;
esac
exit 0
```

Make the script executable.

```
$ chmod +x /etc/init.d/tomcat
```

Turn on tomcat service.

```
$ chkconfig --add tomcat
```

Post Configuration

Modify "/opt/tomcat/conf/tomcat-users.xml" with the following contents. Replace "USERNAME" and "PASSWORD" with your own setting.

```
<?xml version='1.0'?>
<tomcat-users>
    <role rolename="manager"/>
    <role rolename="admin"/>
    <user username="USERNAME" password="PASSWORD" roles="admin,manager"/>
</tomcat-users>
```

Remove any unnecessary preinstalled applications under "webapps" and "work" directories.

```
$ rm -rf /opt/tomcat/webapps/*
$ rm -rf /opt/tomcat/work
```

HTTPS Configuration

As mentioned earlier 2.5.10, we use the CSUSB-signed certificate to secure the CELORS Web site. Here are the steps to request the CSUSB-signed certificate through the In-

formation Security Office.

First, create a certificate signing request (CSR) to request a certificate.

Generate a keystore with only a self-signed certificate:

```
$ keytool -genkey \
          -alias server \
          -keyalg RSA \
          -keystore server.keystore \
          -storepass changeit
```

A series of questions will appear similar to the below. Fill out appropriate unit name, hostname, and E-mail.

```
What is your first and last name?
    [Unknown]: myserver.csusb.edu
What is the name of your organizational unit?
    [Unknown]: College of Examples
What is the name of your organization?
    [Unknown]: California State University San Bernardino
What is the name of your City or Locality?
    [Unknown]: San Bernardino
What is the name of your State or Province?
    [Unknown]: California
What is the two-letter country code for this unit?
    [Unknown]: US
Is CN=myserver.csusb.edu, OU=College of Examples, O=California State
University San Bernardino, L=San Bernardino, ST=California, C=US correct?
    [no]: yes
Enter key password for <server>
(RETURN if same as keystore password):
```

Create the CSR:

```
keytool -certreq \
          -alias server \
          -storepass changeit \
          -keystore server.keystore > cert_req.pem
```

Post the contents of "cert_req.pem" to the Certificate Signing Request page.

Once you receive the signed certificate and its certificate chain from the ISO administrator by E-mail, use the following steps to install the signed certificate into the Tomcat Server.

Copy the following files to "conf" directory:

- Signed celors-csusb-edu.crt (DER formatted certificate)
- IntermediaryCA_cert.crt
- sslCA.crt
- csusbCA.crt

Import the trusted certificate into the keystore used to create the CSR earlier:

```
$ keytool -import -storepass changeit \
  -keystore server.keystore \
  -trustcacerts -alias sslCA \
  -file CSU-San-Bernardino-SSL-CA.crt
$ keytool -import -storepass changeit \
  -keystore server.keystore \
  -trustcacerts -alias IntermediaCA \
  -file CSU-San-Bernardino-Intermediary-CA.crt
$ keytool -import -storepass changeit \
  -keystore server.keystore \
  -trustcacerts -alias csusbCA \
  -file CSU-San-Bernardino-Root-CA.crt
```

Replace the self-signed certificate with the CA-signed certificate:

```
$ cp celors-csusb-edu.crt iep.crt
$ keytool -import -storepass changeit \
  -keystore server.keystore \
  -alias server \
  -file iep.crt
```

Find the connector that has port attribute 8080 and change "redirectPort" attribute to "443" as follows:

```
<Connector  port="80"
        maxThreads="150"
        minSpareThreads="25"
        maxSpareThreads="75"
        enableLookups="false"
        redirectPort="443"
        acceptCount="100"
        connectionTimeout="20000"
        disableUploadTimeout="true"
/>
```

Uncomment the connector element in "/opt/tomcat/conf/server.xml" that has a port attribute set to 443 and modify it as shown below:

```
<Connector  port="443"
        maxHttpHeaderSize="8192"
           maxThreads="150"
           minSpareThreads="25"
           maxSpareThreads="75"
           enableLookups="false"
           disableUploadTimeout="true"
```

```
            acceptCount="100"
            debug="0"
            scheme="https"
            secure="true"
            keystore="conf/server.keystore"
            keystorePass="changeit"
            clientAuth="false"
            sslProtocol="TLS"
/>
```

6.1.8 *MySQL Installation and Configuration*

This section will demonstrate how to install and configure MySQL for the CELORS.

Pre-installation

MySQL depends on "perl-DBI" so perl-DBI must be installed before the MySQL installation. The perl-DBI can be installed with the "yum" command.

```
$ yum -y install perl-DBI
```

Install MySQL

Go to the MySQL official Web site [1] and download the Linux x86 generic RPM release of the MySQL client and server from the section "Red Hat Enterprise Linux RPM" downloads.

Install both client and server rpms as below:

```
$ rpm -ivh MySQL-*
```

Start the MySQL server:

```
$ service mysql start
```

Run the following command:

```
$ /usr/bin/mysql_secure_installation
```

You will be given the option of removing the test database and anonymous user created by default.

Do the following to eliminate Tomcat and MySQL init script conflict:

```
$ service mysql stop
$ chkconfig --del mysql
```

Modify "/etc/init.d/mysql" by removing the following lines:

```
# Comments to support LSB init script conventions
### BEGIN INIT INFO
# Provides: mysql
# Required-start: $local_fs $network $remote_fs
# Should-Start: ypbind nscd ldap ntpd xntpd
# Required-stop: $local_fs $network $remote_fs
# Default-start: 2 3 4 5
# Default-stop: 0 1 6
# Short-Description: start and stop MySQL
# Description: MySQL is a very fast and
# reliable SQL database engine.
### END INIT INFO
```

Save the file and type the following commands:

```
$ chkconfig --add mysql
$ service mysql start
```

6.1.9 JDBC Connector Installation

MySQL Connector/J is a native Java driver that converts JDBC calls into the network protocol used by the MySQL database. It lets developers working with the Java programming language easily build programs and applets that interact with MySQL.

Download the latest MySQL Connector/J from the MySQL official Web site [1]. Extract the "tar.gz" file into home directory:

```
$ tar -zxvf mysql-connector-java-<version>.tar.gz
```

Copy the jar file into "/opt/tomcat/lib" directory:

```
$ cd <MySQL_Connector_DIRECTORY>
$ cp mysql-connector-java-<version>-bin.jar <TOMCAT_HOME>/lib/
```

6.2 CELORS Installation

The installation of the CELORS includes two parts, the Web application and the database.

6.2.1 Database Installation

Check if the database "ors" exists or not. If it exists, skip the following steps:

```
$ mysql -u root -p<mysql_root_password>
$ mysql> create database ors
$ mysql> quit
```

As mention earlier, the CELORS restructures the current database and removes the existing orphan records. Those orphan records will cause the CELORS system to crash. Therefore, alter the old database into the new format and remove the orphan records.

Edit "alterDB.sql" with the following:

```
Alter Table course_category
  add column id bigint auto_increment first,
  add primary key (id);
Alter Table course_discount
```

```
  add column id bigint auto_increment first,
  add primary key (id);
Alter Table course_multiple_cohort
  add column id bigint auto_increment first,
  add primary key (id);
Alter Table course_multiple_discount
  add column id bigint auto_increment first,
  add primary key (id);
Alter Table course_registration
  add column id bigint auto_increment first,
  add primary key (id);
Alter Table header_category
  add column id bigint auto_increment first,
  add primary key (id);
Alter Table registration_type
  drop primary key,
  add column id bigint auto_increment first,
  add primary key (id);
Alter Table registration
  add column enrolledDate date default null,
  change column payment status int;
Alter Table quarter
  add column parking_fee double default 0,
  add column osher_parking_pass double default 0,
  add column osher_membership_fee double default 0,
  add column max_osher_courses_allowed int default 0;
Alter Table marketing
  add column mail_promotion boolean default false,
  add column email_promotion boolean default false;
```

```
Alter Table staff
  add column manage_course_permission boolean default false,
  add column process_student_permission boolean default false,
  add column manage_report_permission boolean default false;
update staff set manage_course_permission = '1'
  where type = 'all' or type = 'ManageCourses';
update staff set process_student_permission = '1'
  where type = 'all' or type = 'ProcessStudents';
Alter Table staff drop column type;
```

Edit "convertDB.sql":

```
-- Category
update category set view = 0 where view = 1;
update category set view = 1 where ISNULL(view);

update header set view = 2 where view = 1;
update header set view = 1 where view = 0 or ISNULL(view);
update header set view = 0 where view = 2;

--  view : 1 --> 0, NULL --> 1, 0 --> 1
update course set view = 2 where view = 1;
update course set view = 1 where view = 0 or ISNULL(view);
update course set view = 0 where view = 2;
```

Edit "dropDB.sql":

```
drop table parkingPass;
drop table parkingFee;
drop table creditCard;
drop table cohortUrl;
```

Have the up-to-date current database .sql file named as cel.sql ready for conversion.

Execute the following commands in sequence:

```
$ cat cel.sql | sed -e 's/MyISAM/InnoDB/g' > celnew.sql
$ mysql -u root -p ors < celnew.sql
$ mysql -u root -p ors < alterDB.sql
$ mysql -u root -p ors < delete-orphan.sql
$ mysql -u root -p ors < convertDB.sql
$ mysql -u root -p ors < dropDB.sql
```

It will prompt for a password while executing the "mysql" command. Enter the root password set up earlier for the MySQL database.

Once done, the database can be restored:

```
$ mysql -u root -p<MYSQL-ROOT-PASSWORD> ors < newcel.sql
```

6.2.2 Software Installation

Run the "build.xml" file to generate the war file from the development environment. Stop Tomcat if it is running. Copy the war file to the CELORS server under the "webapps" directory.

Restart Tomcat:

```
$ service tomcat restart
```

Check that no error message is showing in the "cel.log" file under the "logs" directory.

6.3 System Backup and Restore

This section describes the commands to store and restore the CELORS database.

6.3.1 Database Backup

```
$ mysqldump -u root -p ors > <NAME_OF_DATABASE>.sql
```

6.3.2 *Database Restore*

Drop the existing "ors" database:

```
$ mysql -u root -p

$ mysql> drop database ors;

$ mysql> quit
```

Restore the backup MySQL database file:

```
$ mysql -u root -p ors < <BACKUP_SQL_FILE>
```

7. CONCLUSION AND FUTURE DIRECTION

7.1 Conclusion

The College of Extended Learning Online Registration System facilitates the processing of registrations for the College of Extended Learning (CEL). Based upon the feedback received from CEL staff, and new regulation requirements from the CSU administration, this version of the system represents an improvement over the previous version in many aspects.

First, the course manager has more control of the CELORS. For instance, when the Osher membership fee or the maximum number of courses allowed to take per Osher quarter changes, the course manager has full control to change the number as desired and it is affected immediately. Currently, the course manager must contact CEL's contract programmer to modify source codes and which usually takes several hours to reveal. The clone function makes populating courses descriptions into a new quarter catalog a more expedient process.

Second, this version of the CELORS makes the payment process more accurate, efficient, and secure. In older verisons, CEL staff extract encrypted payment information from the system and stored into a portable device, usually a floppy disk or flash drive. They bring it to an isolated computer to decrypt and print registrations out. They then enter credit card information and fees one-by-one into the credit card machine. A typo could cause the payment to not go through or charge the wrong person. With automating the payment process through Paypal, CEL staff still has control of the payments but less chance for human errors. The registrations won't go

through the CELORS if students make any mistakes during the payment process.

Finally, by using the newer technologies of Spring Framework and Hibernate provided, the system runs more efficiently and is easier to maintain and implement.

7.2 Future Direction

The development of the project progressed much more slowly than anticipated. The project itself has a great deal of life ahead of it though, which can be viewed as a positive aspect, especially if future programmers take interest enough to complete the remaining tasks.

A short list of work that could be done in the future are:

- Implementing marketing analysis reports functionalities
- Adding the Paypal refund functionality
- Adding a pay-through-Paypal-page function (redirect registrating students to the CEL-customized Paypal page)

REFERENCES

[1] MySQL 5.5 reference maunal. http://dev.mysql.com/doc/refman/5.5/en/.

[2] L. Atkinson. *Core MySQL.* Prentice Hall PTR, 2002.

[3] C. Bauer and G King. *Hibernate in Action.* Manning Publishcations Co., 2004.

[4] K. Coar and R. Bowen. *Apache Cookbook.* O'Reilly, 2004.

[5] The PCI Security Standards Council. PCI security standards official web site. https://www.pcisecuritystandards.org.

[6] J. Elliott. *Hibernate: A Developer's Notebook.* O'Reilly, 2004.

[7] The Apache Software Foundation. Apache tomcat documentation. http://tomcat.apache.org/tomcat-6.0-doc/index.html.

[8] R. J. Yarger G. Reese and T. King. *Managing and Using MySQL.* O'Reilly, 2002.

[9] M. Hall. *More Servlets and JavaServer Pages.* Prentice Hall PTR, 2001.

[10] W. Iverson. *Hibernate: A J2EE Developer's Guide.* Addison-Wesley, 2005.

[11] R. Johnson. J2ee development framework.

[12] Sun Microsystems. Java platform, standard edition 6 api spcification. http://java.sun.com/javase/6/docs/api/.

[13] K. Mukhar, C. Zelenak with J. L. Weaver, and J. Crume. *Beginning Java EE 5: From Novice to Professional.* Apress, 2005.

[14] P. Peak and N. Heudecker. *Hibernate Quickly.* Manning Publications Co., 2005.

[15] The Community ENTerprise Operating System. Centos Web site. http://www.centos.org.

[16] C. Walls and R. Breidenbach. *Spring in Action.* Manning Publications Co., 2003.

[17] L. Welling and L. Thomason. *MySQL Tutorial.* Sams Publishing, 2004.

[18] D. Williams. *Pro PayPal E-Commerce.* Apress, 2007.