

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI[®]

**A WEB BASED STUDENT COURSE REGISTER SYSTEM USING
JSP TECHNOLOGY**

GANG CHENG

**A MAJOR REPORT
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE**

**PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE
CONCORDIA UNIVERSITY
MONTREAL, QUEBEC, CANADA**

APRIL 2003

© GANG CHENG, 2003



**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**395 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**395, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-77708-1

Canada

ABSTRACT

A WEB BASES STUDENT COURSE REGISTER SYSTEM USING JSP TECHNOLOGY

Gang Cheng

Web-based application has been developed rapidly since mid 1990s. With the use of JSP technology and the output presented by HTML, Java and J2EE technology provide the enterprise solution for building a modular system. In this report, I describe an online application web site --- Student Course Register System. The goal is to separate the application logic from presentation. The project is designed with Model-View-Controller architecture, and implemented with JSP and Java Bean technology.

Table of Contents

1 INTRODUCTION.....	1
2 BACKGROUND.....	3
2.1 Introduction to Java Servlet Technology	3
2.2 Introduction to JSP.....	5
3 REQUIREMENTS AND SPECIFICATIONS.....	6
3.1 Purpose	6
3.2 Product Perspective	6
3.3 User Characteristics.....	6
3.3.1 Student.....	7
3.3.2 Registry administrative staffs	7
3.3.3 Software Developer.....	7
3.3.4 Maintenance staffer	7
3.4 Functional Requirements Specification.....	7
3.4.1 Student requirement functions	10
3.4.2 Maintenance staff requirement functions.....	12
3.4.3 Registration staff requirement functions.....	12
3.5 System Requirements	15
3.5.1 Platform.....	15
3.5.2 Java Environment.....	15
3.5.3 Java Web Server	15
3.5.4 Internet Connection	16
3.6 User Interface Requirements.....	16
3.7 Performance Requirements.....	17
4 DESIGN	18
4.1 Design Rationale.....	18
4.2 System Architecture	20
4.2.1 Architecture Diagram	20
4.2.2 Deployment Diagram	22
4.3 Object Model and Class Diagram.....	23
4.3.1 Modules in Major Components.....	24
4.4 System Topology	26

4.5.1 Database Assumptions.....	27
4.5.2 Entity-Relation Diagram of CRS Database.....	28
4.5.3 Database Schema	29
5 TESTING AND RESULTS	31
5.1 Environmental Testing.....	31
5.2 Functional Testing.....	32
5.2.1 Test Cases for the Maintenance Module.....	32
5.2.2 Test Cases for the Registration Module.....	35
5.2.3 Test Cases for the online registry module.....	41
6 CONCLUSION AND FUTURE WORK	46
6.1 Conclusion.....	46
6.2 Future work.....	47
6.2.1 Add functionalities.....	47
6.2.2 Add course time and classroom management subsystem	47
6.2.3 Generate a template web application system	47
7 REFERENCES.....	48
8 ABBREVIATIONS	49
APPENDICES:.....	50
Appendix A: Sample JSP source files	50
A-1.Display available courses	50
A-2 select a course and put into register form	52
A-3 register the courses.....	55
A-4 record grades for a course.....	56
Appendix B: Sample JavaBean.....	59
B-1 dbacc bean	59
B-2 avacourse bean.....	60
B-3 regform bean.....	63
B-3 dropcourse bean	65
Appendix C: The installation instructions	68
C-1 Installing JDK.....	68
C-2 Installing Web server—Resin-2.1.6	68
C-3 Installing MySQL	69
C-4 Installing JDBC for MySQL.....	70
C-5 Installing the Web site files	70

List of Figures

Figure 2.1: Servlet Container.....	4
Figure 3.1: Use case diagram.....	9
Figure 4.1: System conceptual model.....	19
Figure 4.2: System Architecture Diagram of CRS.....	20
Figure 4.3: The sub-components in JSP package.....	22
Figure 4.4: The deployment diagram of CRS system.....	23
Figure 4.5: Class diagrams of Modules in Major Components.....	25
Figure 4.6: System Topology of CRS.....	26
Figure 4.7: ER diagram of CRS database.....	28

1 Introduction

Since the mid-1990s, the growth of the Internet as a communications vehicle has been developed rapidly. Up to now, the Internet has evolved in potentially most parts of our life. As the base of technology development, the university is also the pioneer of the new technology application. The online student course register system can simplify the procedure of the student course registry. This system is convenient to students and the university management. Online application has changed all the traditional rules of the university management. This web site contains not only the static content, but also the dynamic information, which include data query, data exchange, etc.

After many years development, now we have many methods that can be used in implementing the dynamic Internet application, for example CGI, Perl, ASP (Active Server Page), PHP, and JSP (Java Server Page). [2]When developing a web application, selecting the right implementation technology is very important. A right choice will make the system can be used not only today, but also in the future. But if we select an unsuitable implementation technology, the system may go into trouble in the future. The reason is that we must meet not only the current needs but also some possible needs in the future when we developing a project. Now days, more and more web applications are using a multi-ply, J2EE architecture. In this model, HTML is used for the presentation layer. Most software developers are trying to separate the content of the web site from the presentation logic. They embed as little as possible logic programs into the HTML file. This makes it possible to adopt new technologies in sending and receiving information. By separating the content of the web site from the presentation logic, the programmer can concentrate on the implementation logic and the web designer can concentrate on the interface graphic design.

This report describes a university student course registry web site (CRS) that I built using JSP and Java Bean technologies. In this document, chapter 2 provides the background of the technologies used in the project, chapter 3 outlines the requirement specifications,

chapter 4 is the software design document, chapter 5 is the testing plan and result, chapter 6 gives the conclusion and the future work, chapter 7 lists the references used in this project, chapter 8 gives the abbreviations used in this document, and chapter 9 appends the sample implementation codes and the installation instruction.

2 Background

2.1 Introduction to Java Servlet Technology

Before, Microsoft's Active Server Pages (ASP) is almost the only choice for every web application developer. But along with the appearance of Sun Microsystems's Java Server Pages (JSP), more and more developers are adopting JSP as their development technology. In fact, JSP and ASP are similar technologies: both add to HTML special tags, a scripting language, and the ability to call out to external software components. JSP is based on Java servlet technology. At run time, every JSP is a servlet class.[3]

The servlet container is a part of a web server or application server that provides the network services over which requests and responses are sent, decodes HTTP based requests, and formats HTTP based responses. A servlet container also contains and manages servlets through their lifecycle.

A servlet container (Figure 1) can be built into a host web server, or installed as an add-on component to a Web Server via that server's native extension. All servlet containers must support HTTP as a protocol for requests and responses. [1][4]

J2SE 1.2 is the minimum version of the underlying Java platform with which servlet containers must be built.

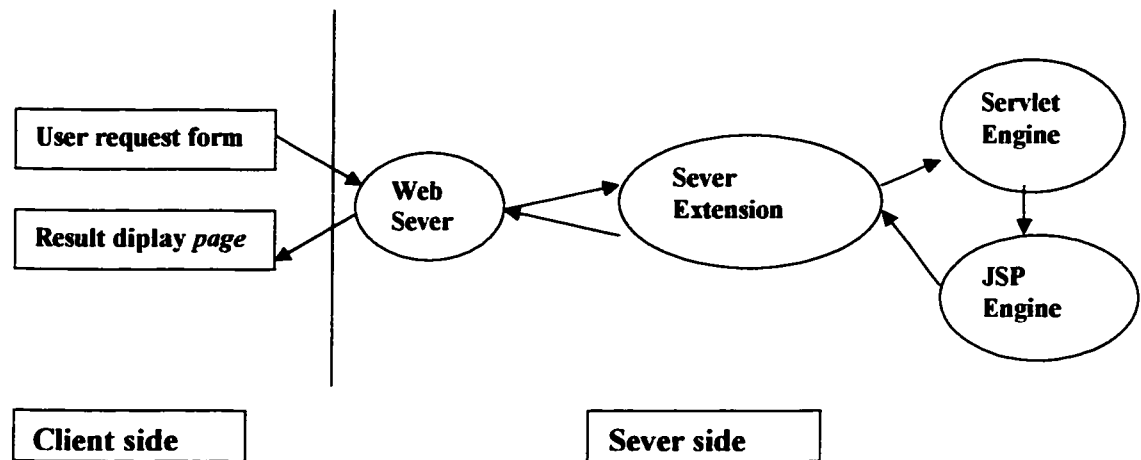


Figure 2.1: Servlet Container

The following is a typical sequence of events:[5]

- A client (e.g., a web browser) accesses a web server and makes an HTTP request.
- The request is received by the web server and handed off to the servlet container. The servlet container can be running in the same process as the host web server, in a different process on the same host, or on a different host from the web server for which it processes requests.
- The servlet container determines which servlet to be invoked based on the configuration of its servlets, and calls it with objects representing the request and response.
- The servlet uses the request object to find out who the remote user is, what HTTP POST parameters may have been sent as part of this request, and other relevant data. The servlet performs whatever logic it was programmed with, and generates data to send back to the client. It sends this data back to the client via the response object.
- Once the servlet has finished processing the request, the servlet container ensures that the response is properly flushed, and returns control back to the host web server.

2.2 Introduction to JSP

What exactly *is* a JavaServer Page? [1] In its basic form, a JSP page is simply an HTML web page that contains additional bits of code that execute application logic to generate dynamic content. JSP comes out of the gate as a component-centric platform. It's based on the a model in which JavaBeans™ components contains the business and data logic for an application, and it provides tags and a scripting platform for exposing the content generated or returned by the beans in HTML pages. This application logic involves JavaBeans™ and JDBC™ objects, which can be easily accessed from a JSP page. For example, a JSP page may contain HTML code that displays static text and graphics, as well as a method call to a JDBC object that accesses a database; when the page is displayed in a user's browser, it will contain both the static HTML content and dynamic information retrieved from the database.[5]

The separation of user interface and program logic in a JSP page allows for a very convenient delegation of tasks between web content authors and developers. It also allows developers to create flexible code that can easily be updated and reused. Because JSP pages are automatically compiled as needed, web authors can make changes to presentation code without recompiling application logic. This makes JSP a more flexible method of generating dynamic web content than Java servlets, whose functionality JavaServer Pages extend. [5][6]

JavaServer Page technology is an extension of the Java Servlet™ API. Servlets are platform-independent, 100% pure Java server-side modules that fit seamlessly into a web server framework and can be used to extend the capabilities of a web server with minimal overhead, maintenance, and support. [7] Unlike other scripting languages, servlets involve no platform-specific consideration or modifications; they are Java application components that are downloaded, on demand, to the part of the system that needs them. Together, JSP technology and servlets provide an attractive alternative to other types of dynamic web scripting/programming that offers platform independence, enhanced

performance, separation of logic from display, ease of administration, extensibility into the enterprise and most importantly, ease of use.[4]

3 Requirements and Specifications

3.1 Purpose

The graduate student course register system is going to allow graduate students to register course via Internet. This system also allows the course registry staffs of the department in university to do administrative matters that concern with the student course registry. This project implements an online course registry web site for a department of an university. The web site can be accessed anywhere through the Internet, and allows the users to select and register course or assign new course.

3.2 Product Perspective

The finished product will be installed in a computer, which runs a Java web server (support JDK1.3) and connected to the Internet. The operating system could be any system if it supports JDK1.3, Windows, Unix/Linux. Data is stored in a small database, and a database management system is required.

When accessing the system, users would need an Internet connection through a local Internet Service Provider. Users would use a web browser, such as Internet Explorer 5.0, and Netscape 4.7. In addition, user will need an email service to retrieve the online confirm of their course registry and/or drop.

3.3 User Characteristics

There will be four categories of users: students, registry administrative staffers, software developer and maintenance staff.

3.3.1 Student

A student, currently registered in this department and has student ID, is a person who select and register courses through the system. Most of them are assumed to be familiar with the Internet applications, and have experience of using Internet Explorer or Netscape.

3.3.2 Registry administrative staffs

A registry administrative staffer is a person who supplies the courses listed in the system. The registry administrative staffer would need to provide the detailed information of the courses, and update them on time. The registry administrative staffer would also need to deal with the course registry and dropping deadline dates management. The registry administrative staffer would also need to deal with the student course dropping application. An email service is needed for the registry administrative staff to retrieve the online registry, and contact the students.

3.3.3 Software Developer

A software Developer is a person who developed the system and may participated in installing the system. They would have the knowledge of the system and configurations. The system they built should meet the requirements of the university, and may add more functionality later on. They need to fix any problems reported by the maintenance staff.

3.3.4 Maintenance staffer

A member of the maintenance staffer is a person whose responsibilities include stores, updates or deletes the user information, and reports any error occurred in the system. Maintenance staff would have the knowledge of the system, and access to the student and course data files regularly.

3.4 Functional Requirements Specification

In this section, a more detailed description of each of the system functions will be given. A forms-based approach will be used. At the beginning, we shall describe a standard

template to be used as a guide for requirement specification. The template may have all or some of the following fields:

Function's name

Actor: function's user,

Purpose: function's objective,

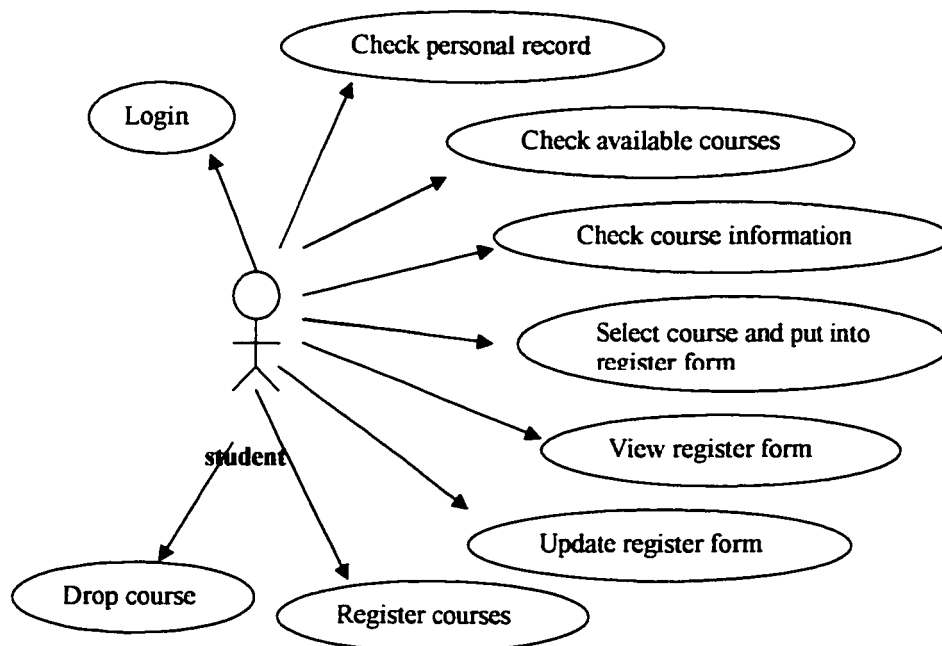
Description: self-explanatory.

Requisite(s): pre-condition(s) that must be satisfied before the function is invoked.

Remarks: additional commentaries about the function.

As mentioned above, more than one actor may access the same function(s). The services provided by such functions may vary from an actor to another. Hence, more detailed description is called for as seen below.

The main functional requirements are divided into three groups which are student requirements and registry administrative staff requirements, and maintenance staff requirements. they are shown in the use case diagram (Figure 3.1).



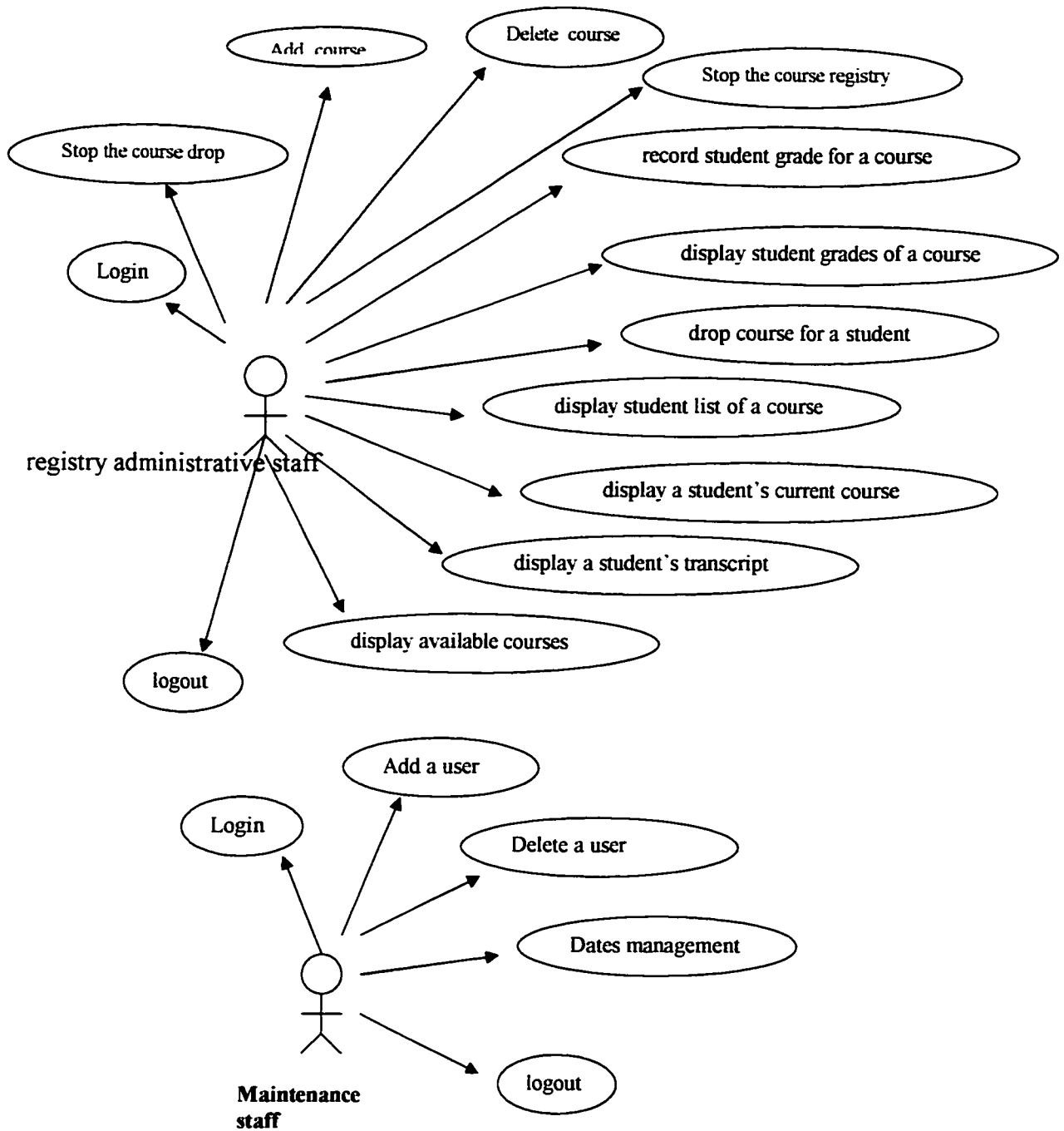


Figure 3.1: Use case diagram

The following is the detailed description of each function:

3.4.1 Student requirement functions

3.4.1.1 Login

Actor: All currently registered graduate students of the department.

Purpose: Login the system to register course

Description: The user login the student course register system with personal user name and password. The system will automatically recognize if the user is a full time student or part time student and assign the maximum number of courses that is allowed the user to register within a term.

3.4.1.2 Check personal record

Actor: All currently registered graduate students of the department.

Purpose: Display the user's personal record.

Description: The actor can display personal record. The record will show all courses that actor has finished or registered with the grade and the term in which the actor take the course.

3.4.1.3 Check available courses

Actor: All currently registered graduate students of the department.

Purpose: Allow the user to view the available course list.

Description: All currently available courses with detailed information concerning with the courses register are displayed. The information includes course name, course section, course term, course/lab day and time, classroom, professor name, tutor name and the course prerequisite. For any current course, if there is no available position anymore, the course will not be displayed.

3.4.1.4 Check course information

Actor: All currently registered graduate students of the department.

Purpose: Allow the user to check the detailed information of a course.

Description: The user can view the detailed information about the course and the information of the professor. The links to the course homepage and the professor homepage are provided.

3.4.1.5 Select a course and put into register form

Actor: All currently registered graduate students of the department.

Purpose: Allow the user to select course and put into register form.

Description: The user can select a course from the current available courses list and put it into the course register form.

3.4.1.6 View register form

Actor: All currently registered graduate students of the department.

Purpose: Allow the user to view the content of the register form.

Description: All courses that the user has put into the register form will be listed in the register form with their name, section and term.

3.4.1.7 Update register form

Actor: All currently registered graduate students of the department.

Purpose: Allow the user to edit the course register form before the final registry.

Description: The user can modify the course register form by removing the course from the register form. The register form will be updated accordingly.

3.4.1.8 Register courses

Actor: All currently registered graduate students of the department.

Purpose: Allow the user to register the selected courses.

Description: By submit the register form, the user will finally register all courses in the register form.

3.4.1.9 Apply to drop course

Actor: All currently registered graduate students of the department.

Purpose: Allow the user to apply to drop the selected courses.

Description: By submit the drop application form, the user will send the course drop application to the course registry staffs. The course drop application must be sent before the drop course deadline.

3.4.2 Maintenance staff requirement functions

3.4.2.1 Add a user

Actor: Administrative staff who is in charge of the student course register system

Purpose: Allow the user to create a new student information and set access right for each student.

Description: The actor is allowed to create user personal information. The actor can change them also. When the student is created, the default access right is to student and the student status (full time or part time) is also created.

3.4.2.2 Delete a user

Actor: Administrative staff that is in charge of the student course register system

Purpose: Allow the user to delete a student information from the student list, then the student can not use the system anymore.

Description: By providing the student name and student id, the student information will be deleted from the student list .

3.4.2.3 Dates management

Actor: Administrative staff that is in charge of the student course register system

Purpose: Allow the user to update the course registry deadline date and update the course dropping deadline date for every semester.

Description: By providing the course registry due date, the course drop deadline date and selected the semester, the corresponding term's dates will be updated.

3.4.3 Registration staff requirement functions

3.4.3.1 Display a student's personal record

Actor: Administrative staff that is in charge of the student course register affair

Purpose: Allow the user to view a student 's personal record.

Description: By providing the student name and student id, all courses that the student has finished will be displayed with their name, term and the student's grade.

3.4.3.2 Display available courses list

Actor: Administrative staff that is in charge of the student course register affair

Purpose: Allow the user to view the currently available courses list.

Description: The currently available courses list is display with the course information (course name, section, term, course day, course time, professor name, classroom, lab time, lab day, tutor name, the capacity of the class, current student number and the course prerequisite).

3.4.3.3 Display student list for a course

Actor: Administrative staff that is in charge of the student course register affair

Purpose: Allow the user to view the student list of a course.

Description: By providing the course name, section and term information, the user can view the student list(name and student id) of the course

3.4.3.4 Display a student's current course

Actor: Administrative staff that is in charge of the student course register affair

Purpose: Allow the user to view a student 's current registered courses.

Description: By providing the student name and student id, all courses that the student has registered will be displayed with their name and term information.

3.4.3.5 Cancel course for a student

Actor: Administrative staff that is in charge of the student course register affair

Purpose: Allow the user to delete a course for a student .

Description: By providing the student name and student id, all courses this student currently registered will be display. The Actor could select a course from the list and click delete to delete it from the register list for the student.

3.4.3.6 Record grade for a course

Actor: Administrative staff that is in charge of the student course register affair

Purpose: Allow the user to record the student grades for a course.

Description: By providing the course name, section and term information, the student list of this course will be listed with student name and student id. Along with every student, there is a grade input box for the user to record the student's grade. By submit, the course 's student grade will be saved.

3.4.3.7 Display all student grades for a course

Actor: Administrative staff that is in charge of the student course register affair

Purpose: Allow the user to view all student grades of a course.

Description: By providing the course name, the student grades will be listed with student name ,student id and student grade.

3.4.3.8 Add a new course

Actor: Administrative staff that is in charge of the student course register affair

Purpose: Allow the user to create a new course information and set it to be available to students.

Description: The actor is allowed to create a new course information. The information includes the course name, section, term, course day, course time, professor name, classroom, lab time, lab day, tutor name, the capacity of the class and the course prerequisite.

3.4.3.9 Delete a course

Actor: Administrative staff that is in charge of the student course register affair

Purpose: Allow the user to delete a course from the available course list.

Description: The user deletes a course from the available course list by provide the course name, section and term information.

3.5 System Requirements

The Internet application typically runs on a web server. In this report, JSP and Java will be the implementation language. To support these requirements, the system will be built on a platform, which supports Java. Since Java language is platform independent, the system could be in either Windows 98/2000/NT/XP or Unix/Linux platform with a Java web server.

In this Student Course Register System, hundreds of data will be stored. The data access involves reading and writing. Therefore, it will need a database management system to manage the data.

Based on the above reason, the following is the system requirements:

3.5.1 Platform

Windows 98/2000/NT/XP or Unix/Linux will be the platform for this project. The minimum requirements for PC are: CPU 486 or higher, 32 MB memory or higher, and 5 GB hard disk or higher.

3.5.2 Java Environment

Java 2 Standard Edition (J2SE™) v1.3 or above contains the essential Java SDK, tools, runtime environment, and APIs for developers writing, deploying, and running applications in the Java programming language. It can be downloaded free of charge from SunJava.[7]

3.5.3 Java Web Server

Resin-2.1.6 Server is used in this project. Resin is a cutting-edge XML Application Server. It has the fastest Servlets 2.3 engine, supports JSP 1.2. It can be installed in either Windows or Unix/Linux system. Resin is open source, the latest server version can be downloaded free from Resin download site.[8] In this system, most use cases involve interact with the backend database. Most web pages are dynamic pages, and JSP is used

to deal with these dynamic pages. Resin is a fast servlet and JSP engine supporting load balancing for increased reliability. Processing of JSP 1.2 files is the simplest use of Resin. Unlike Apache server is good in dealing with static pages, resin server is good in dynamic pages. So I select Resin as the server for this system.

The system security is also very an important problem when select the web server. Resin server can be configured as a security server. It has three separate functions for its security configuration: authentication, authorization and encryption. Authentication can be used to detect the user who is trying to use the resource. It has two kinds of methods to realize the authentication:

- . login-config: How the client sends the user and password.
- . authenticator: How the server checks the user and password to authenticate a user.

For this system, it is a special web site for special users. We will put the server inside the university's intranet. So physically, the server should be secure because it is behind the firewall. In the future work of this system, I will make the server more secure from physical, network and administrative aspects.

3.5.4 Internet Connection

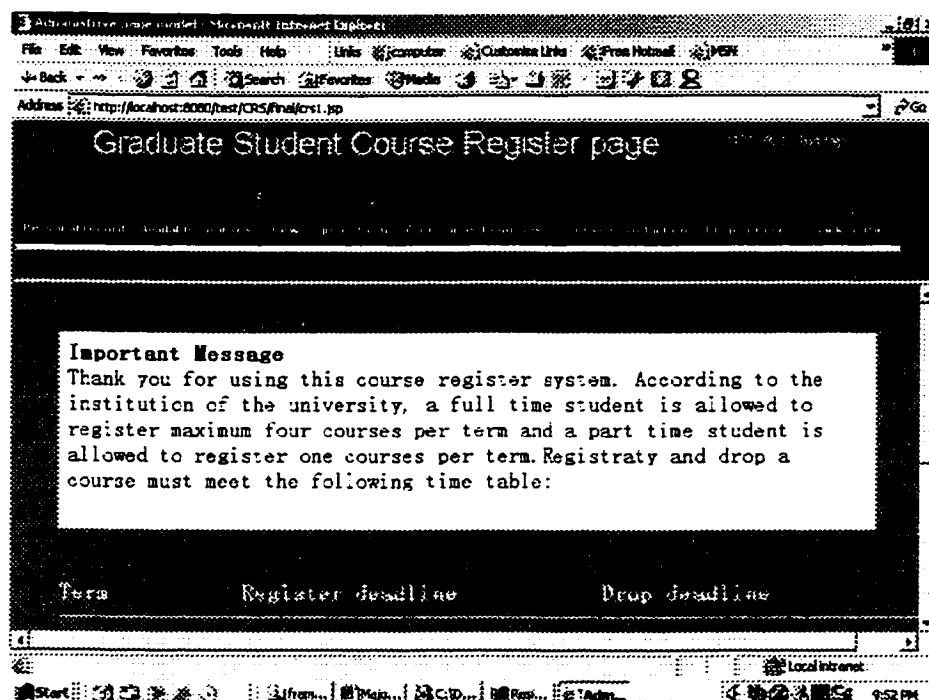
A steady Internet connection is required for running the web server. DSL high-speed connection or cable connection is recommended, though dial-up connection with at least 56k bps modem can also be considered.

3.6 User Interface Requirements

A user friendly GUI (Graphic User Interface) is required for frequent visitors to browse through the web browser. The web browser should be Internet Explorer 5.0 or higher, and Netscape 4.6 or higher.

The user interface will be transparent to the user, and extremely easy to learn. No special skills are required to use the system, just the ability to read and understand the displayed

textual information. General options are provided on every page so that a user will always have the opportunity to restart everything from the beginning.



3.7 Performance Requirements

The system will provide the concurrent access at any time. MySQL's maximum connection default is 100 users. Depending on the Internet connection and the Resin server specification, I choice that the system can be accessed concurrently by 80 users.

The system response time to a normal request must not exceed ten seconds. In some cases remote users may experience longer response-times depending on network-traffic condition over the Internet. For email service, it will take no more than 15 seconds. It also depends on the Internet connection speed.

4 Design

4.1 Design Rationale

My goal in the CRS(Course Registry System) design is to achieve good maintainability for easy modifying and expending which depend on the designed architecture of the CRS system. The CRS is the application that leverages Web server, Web clients (such as Web browsers) and standard Internet protocols. It also typically leverages existing application and data from external non_Web services. The Application Framework for CRS is designed to be an integrated and consistent collection of APIs, protocols, services, and conventions that provides an open, standards based, scalable and complete foundation for developing and deploying web application solutions. At the heart of Framework is a single, unifying Java-based programming model for building CRS. The JavaServer PageTM (JSP) is a key component serving as the preferred request handler and response mechanism. This ensures us to easily develop high quality, maintainable CRS Web applications.

Figure 4.1 illustrates a three-tier model of CRS Web applications. The client tier is composed of multiple clients, which request services from the middle tier. The middle-tier consists of two sub-tiers, the Web Server and Application Server. The Web server contains JSP pages as event handler, and the Application server contains JavaBean as dynamic component of logical rules. They access data from the database tier, apply logical rules to data, and return the results to the client tier. The JSP page and Bean in middle tier server plays a vital role in three-tier application.

4.2 System Architecture

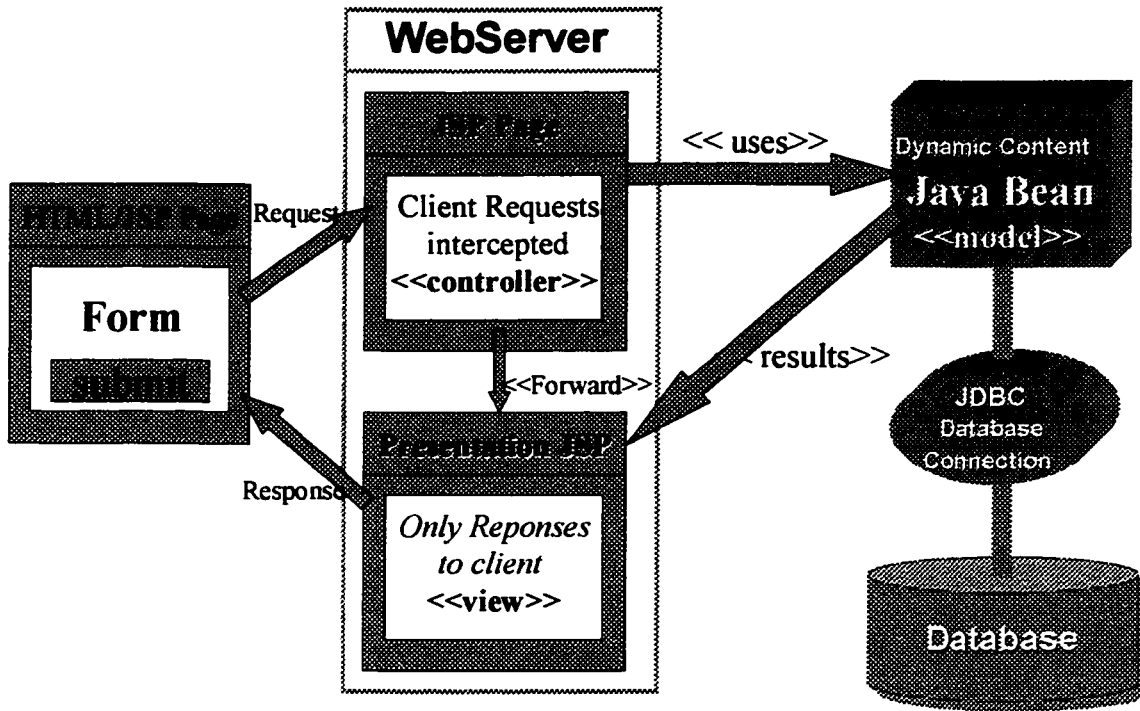


Figure 4.2: System Architecture Diagram of CRS

4.2.1 Architecture Diagram

The CRS system has to respond to asynchronous events from the users and database. Users are allowed to view or query and even update information of CRS thus interacting with the system. The Interactive architecture best addresses these requirements. On the other hand, the system contains no predefined sequence of action and only responds according to user inputs (either data input or control input), so I identify that CRS system is event-driven interactive system. The CRS should also have the characteristics of the user interactivity and friendliness, the database orientation.

The architecture, shown in Figure 4.2, is a hybrid approach for serving dynamic content, by using JSP. It takes advantage of the predominant strengths of both technologies, using

JSP to generate the presentation layer and to perform process-intensive tasks. We chose Model View Controller as the most appropriate pattern for system design. MVC separates the Model and View, thus offering a way to increase the system modularity. Here, the JSP page acts as the *controller* and is in charge of the request depending on the user's actions. The JSP page forwards the request to the JavaBean as *model*. Note particularly that there is no processing logic within the JSP page itself; it is simply responsible for retrieving any objects or beans that may have been previously created, and extracting the dynamic content from that servlet for insertion within static templates. This approach typically results in the cleanest separation of presentation from content, leading to clear delineation of the roles and responsibilities of the developers and page designers on our programming team.

As shown in the system architecture diagram in Figure 4.2, the CRS system consists of the following subsystems:

- **User Interface:** The UI allows the general users to interact with the CRS system through the web using an internet browser, and also some authorized user to use the system directly through their personal computer with different privileges by inputs (mouse clicks, keyboard strokes etc.).
- **Web Server:** It is major design part of CRS system, to handle the user requests from user inputs. The JSP page in Web server uses logical rules in JavaBean dynamic component, to get the results. **JSP Package (the Controller)** consists of sub-components for SYSTEM module and for users, as shown as in Figure 4.3 **Presentation JSP Pages (the View)** generates the presentation pages of the response from JavaBean, and send to client to display in user interface browser.
- **JavaBean:** It is dynamic component served as **the Model**, it receives the information from Web server and connects the database with the JDBC. It contains formatting beans, command beans and application logic.

- **JDBC:** the database connection.
- **Database:** To store the whole information of the CRS system. To receive the SQL queries and to give the query results out.

The software to be used in the different subsystems in our design can be listed as follows:

- MySQL as relational database.
- Resin-2.1.6 on Windows NT 4.0 /98/2000 or on Linux/Unix as Web server engine.
- JDBC as the connectivity between the Web server and Database server.
- Microsoft Internet Explorer or Netscape Browser.

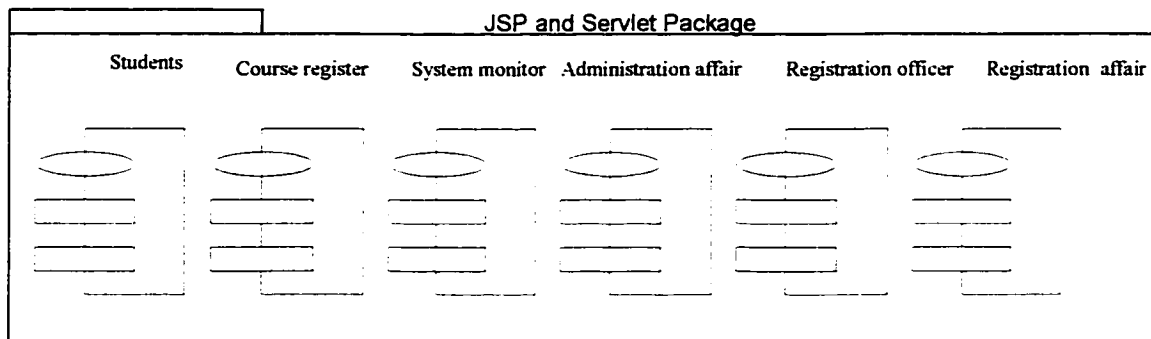


Figure 4.3: The sub-components in JSP package

4.2. 2 Deployment Diagram

The deployment structure for current CRS system is relatively simple. The client Web Browser is on client PC. The Web Server engine is Resin on Windows NT/98/2000 or Unix/Linux. The connection between Web Browser and Web Server is standard internet protocols, TCP/IP. The system architecture of CRS ensures that it is platform independent.

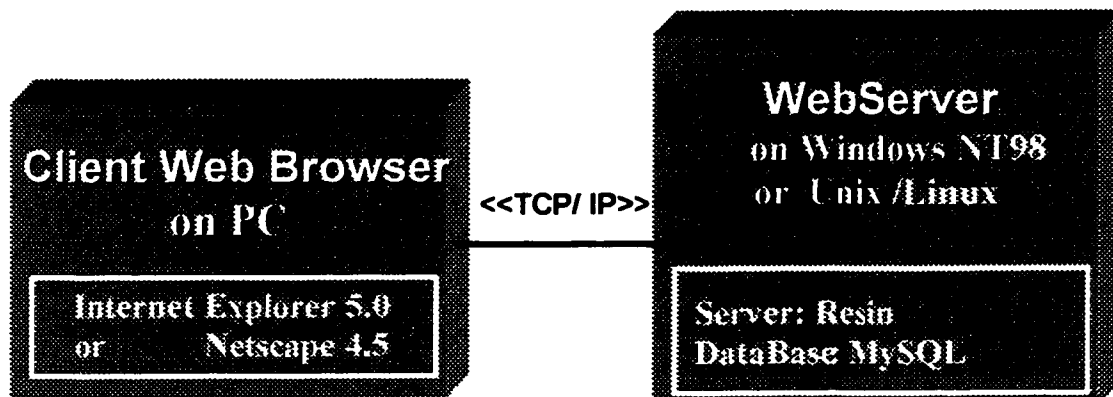


Figure 4.4: The deployment diagram of CRS system

4.3 Object Model and Class Diagram

The object model shows the static data structure of the real-world system and organizes into workable pieces. The object model describes real-world object class and their relationships to each other.

Class diagram shows classes and the relationships between classes. The class diagrams of each component of the system are divided into two sections. Due to the simplicity of the functionality, we classify the user interface (UI), WebServer, BeanTemplate, JDBC and Database into the section of the major modules, the class diagram has been shown in Figure 4.5. All modules in JSP package are classified in another section.

4.3.1 Modules in Major Components

4.3.1.1 User Interface Module

The user interface subsystem consists of only one user interface module. The UI subsystem is simple to display the interfaces to general or advanced users. It contains the static text, images, tables, forms, buttons, check boxes, etc. The users can get information, click on a command button to select a service, do a form processing, and display a table.

4.3.1.2 WebServer Module

The WebServer module has two attributes: scriptCompiler and scriptInterpreter, and four functions: compileScript(), runScript(), setClientRequest() and receiveJSPResponse().

4.3.1.3 BeanTemplet Module

As the logical component, the BeanTemplet module consists of several Bean templates, which are CourseRegisterBean, AdministrationAffairBean and Bean classes for database functions.

4.3.1.4 JDBC Module

The JDBC consists of two attributes: driverManager and driver, and three member functions: getConnection(), creatStatement() and executeQuery().

4.3.1.5 Database (DB) Module

The DB module consists two attributes: Entities and Relationship, and three member functions: readSQL(), runSQL() and returnRsults().

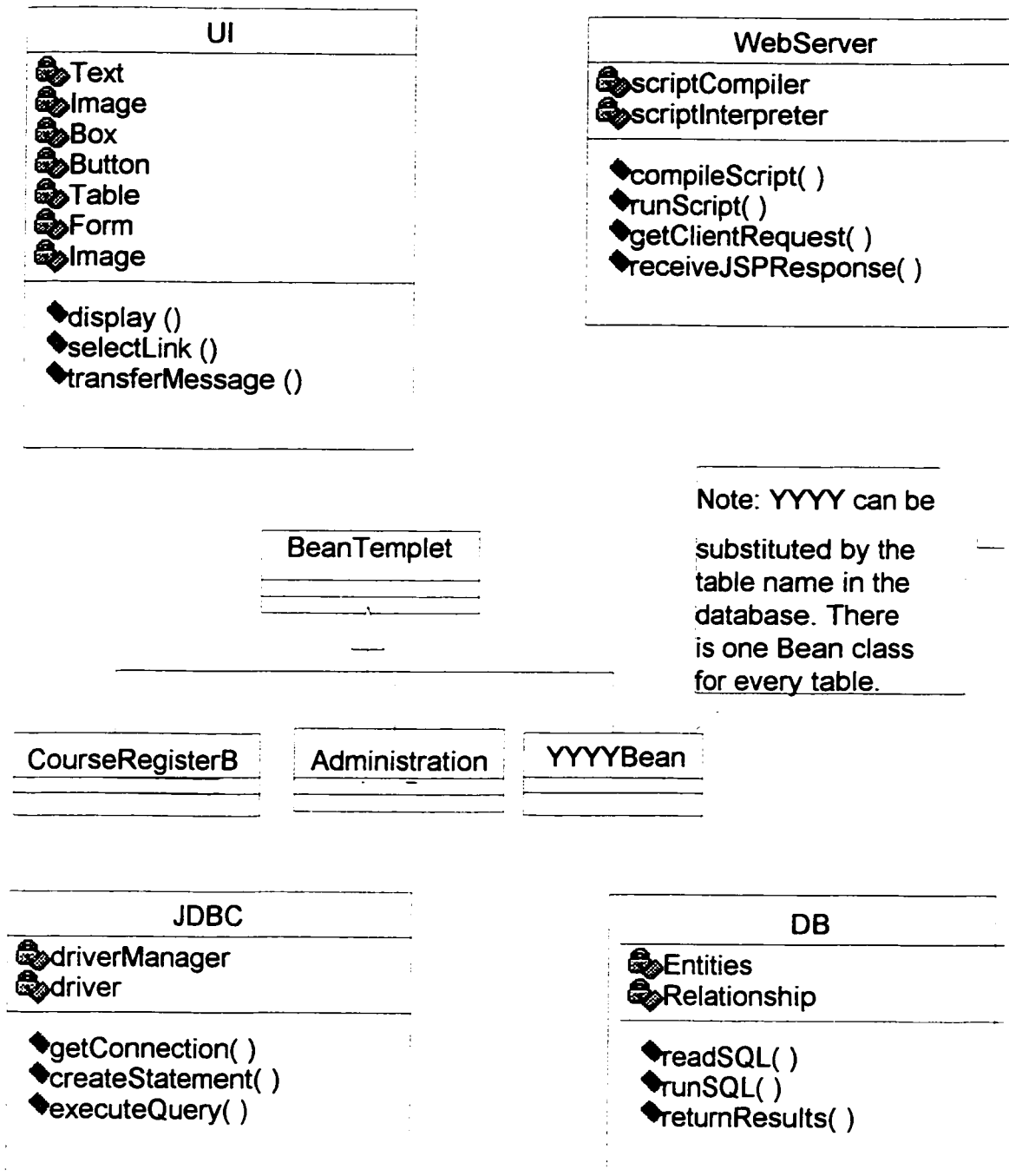


Figure 4.5: Class diagrams of Modules in Major Components

4.4 System Topology

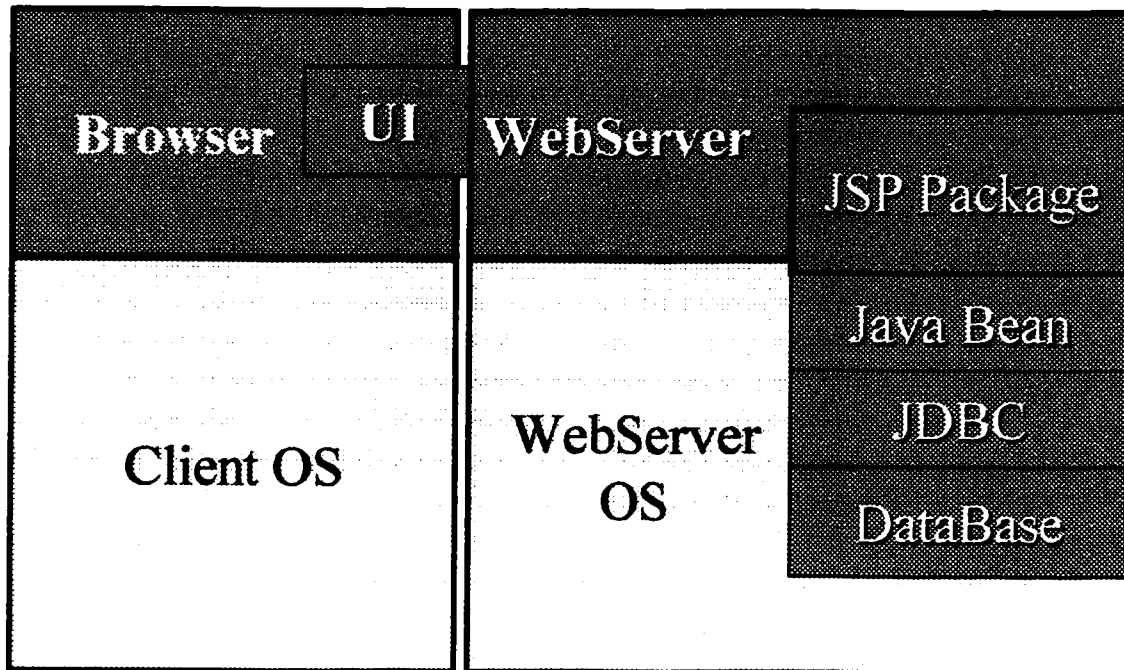


Figure 4.6: System Topology of CRS

The system topology of CRS is shown in Figure 4.6 is composed of two subsystems: Web Server and Web Browser. UI is a sub-component of Web Browser. The relationship between UI and Web Server is the peer-to-peer.

The JSP package is a sub-component of Web server. JDBC connects the JavaBean and DB (database). The relationships between JSP package and JavaBean, JavaBean and JDBC, JDBC and DB are of Client/Supplier type.

The bottom layer contains the user operating system and the server operating system.

4.5 Data Model

4.5.1 Database Assumptions

Based on the requirement analysis and specification for the CRS System, some assumptions and constraints are made to develop a high-level description of the data to be stored in the database, which are listed as follows:

- A ***user*** should be identified by a unique user name, user ID and have a password for enter the system. Each user has a level as unique privilege identification for access the system; The user information must be kept in the database;
- There are three levels users: students, registry administration staff, and system administrator.
- The system administrator has privilege to access administration subsystem and create user, set user level, delete user, create course, delete course.
- The registry administration staff has privilege to access course registry management subsystem and drop course for student, record grads for a course, display name list for a course, display student record, etc.
- The student has privilege to access individual course register subsystem, view his/her individual student record and register individual courses and apply to drop a course.
- The system administrator can create/delete a ***user*** and an ***avacourse***.
- The student user can create a ***reglist*** and a ***dropcourse***. The registry administration staff can create a ***sturecord***.
- The system should store some important ***dates*** for the deadline of course registry and drop course.

4.5.2 Entity-Relation Diagram of CRS Database

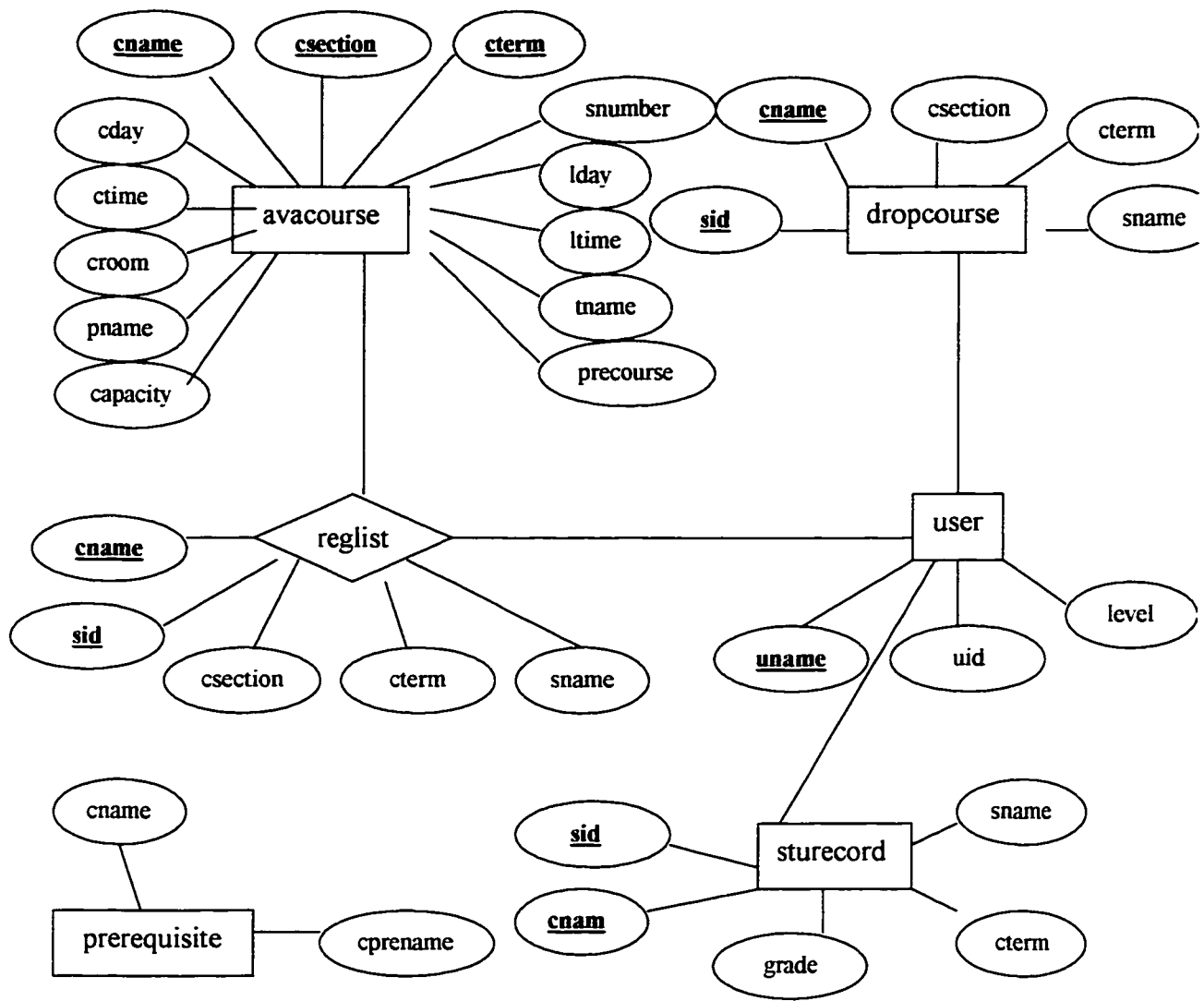


Figure 4.7: ER diagram of CRS database

The conceptual design of the CRS database is carried out using the ER model, which allows us to describe the data at a high level of abstraction. Based on the above assumptions and constraints, five entities and relationships were chosen to build the ER diagram as shown in Figure 4.7: ER diagram of CRS database

Different shapes and lines are used in this ER diagram and they are illustrated as follows. The rectangles in the diagram represent entities, the black line rectangles represent weak entities, the diamonds represent relationships among them. The attributes for an entity or relationship are given in the ellipses. The primary key in an entity is in **bold and underlined**, and a partial key in a weak entity is in **bold and italic**. The relationship set can be one-to-one (1:1), one-to-many (1:n), and many-to-many (m:n).

4.5.3 Database Schema

The ER diagram is translated into 6 relations. For each relation (schema), its constitution will be indicated. The primary key is written in **bold**. A short description will be given to the attributes marked by *.

4.5.3.1 user

Entity: user

user (**uname**:text, **pwd**:text, uid*:number, level*:number)

Level indicates full time student(4), part time student (1), coordinator(0), administrator(3); uid indicates the student id number for student.

4.5.3.2 avacourse

Entity: avacourse

avacourse(**cname**:text, **csection**:text, **cterm**:text, cday:text, ctime:text, croom:text, pname: text, capacity*: text, snumber*: text, lday: text, ltime: text, tname: text, precourse*: number)

capacity indicates the maximum number of students allowed in this section;
snumber indicates the number of students that currently register this course;
precourse indicates no prerequisite course needed(0), need prerequisite course(1).

4.5.3.3 sturecord

Entity: sturecord

sturecord(**sid**:text, **cname**:text, grade:text, term:text, sname:text)

4.5.3.4 reglist

Relationship: reglist

reglist(cname:text, sid:text, csection:text, cterm:text, sname:text)

4.5.3.5 prerequisite

Entity: prerequisite

prerequisite(cname:text, cprenamename:text)

4.5.3.6 dropcourse

Relationship: dropcourse

dropcourse t(cname:text, sid:text, csection:text, cterm:text, sname:text)

5 Testing and Results

Testing is defined as "The process of exercising or evaluating a system by manual or automatic means to verify that it satisfies specified requirements or to identify differences between expected and actual results" (IEEE, 1983).

The testing plan for CRS is based on the requirement specification. The goal is to validate and verification of this product, to ensure the system's reliability, availability and maintainability.

The testing strategy I will use is the bottom-up testing, where testing starts with the fundamental component and works upwards. I chose bottom-up testing as my testing strategy because each individual object can be tested using its own test driver (user-interface, in our system), which is very easy to implement. Then, all objects will be integrated and the object collection will be tested. A stress testing will be conducted to try to estimate the max number of people that can access, online, the database simultaneously.

In defect testing, I will use black-box testing. The tests will be conducted in different phases of the development. One reason that I chose black-box testing as the testing strategy is because I could start the tests as early as the program specifications are set. Therefore, the sooner I start the testing, the least defective our developed system will be. Another important reason for choosing black-box testing is because it was found to be more effective in discovering faults than white-box testing (Basili and Selby).

5.1 Environmental Testing

The environment testing is used to determine the server side and client side hardware and software requirements for this system. The results are listed below:

Server side:

- CPU: Pentium III 800.
- Memory: 256 MB.
- Hard Disk Storage: 40GB.

- Operating System: Windows 2000 from Microsoft.
- Server: Resin-2.1.6 with JSP/Servlet engine, from Caucho technology.
- Java Environment: Java 2 Standard Edition from Sun.
- Internet Connection: DSL from Bell Sympatico High Speed Edition.

Client side:

- Browser: Internet Explorer 5.0 and Netscape 4.7 at Windows 2000.
- Internet Connection: DSL from Bell Sympatico High Speed Edition.
- URL:

Server-Client connection:

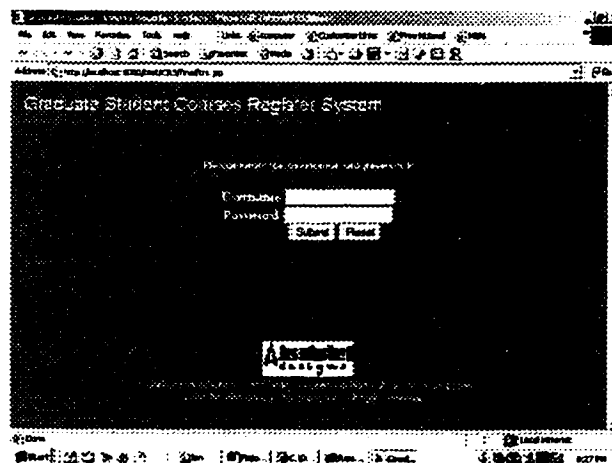
- Server and client are in the same computer.
- Server and client are connected through TCP/IP protocol.

5.2 Functional Testing

5.2.1 Test Cases for the Maintenance Module

In this section the test cases of the functionality of the maintenance module is presented.

5.2.1.1



Test function: user login (User's authority)

Rationale: The system has to check the user's authority, first, to allow each user use the functionality of the system. Users have different privilege levels.

Notes: There are three major types of users:

1. Current students;
2. Registrar's officers;
3. System monitor.

Case: On the system's entry page, there is a form with two input text boxes for inputting user name and password. All these are necessary information. If any necessary information of the student is not filled, the page will not proceed to the next page, the related field name will be in red color which indicates that the specific information should be filled or corrected. The message "Please fill in correct information" will be provided. After filling the form, click the submit button, if the user is a student, main page for students will be displayed ; if the user is a registrar's officer, main page for registrar's officers will be displayed ; if the user is a system monitor, main page for system monitor will be displayed.

5.2.1.2

Test function: add a user

Rationale: To test the correct operation of adding a student.

Case: Upon clicking on the "add a student" button, a form with input text boxes for student name, student id and student level will appear on the page. The student name, student id and student level are necessary information. If any necessary information of the student is not filled, the page will not proceed to the next page, the related field name will be in red color which indicates that the specific information should be filled or corrected. The message "Please fill in correct information" will be provided. After filling the form, click the submit button, this student will be added into the user table and the message " the student has been added" will be provided in a new page with all of this student's information displayed.

5.2.1.3

Test function: delete a user

Rationale: To test the correct operation of deleting a student.

Case: Upon clicking on the “delete a student” button, a form with input text boxes for student name and student id will appear on the page. All of these are necessary information. If any necessary information of the student is not filled, the page will not proceed to the next page, the related field name will be in red color which indicates that the specific information should be filled or corrected. The message “Please fill in correct information” will be provided. After filling the form, click the submit button, this student will be deleted from the “user” table and the message “ the student has been deleted” will be provided in a new page. This student will not be able to use the system again.

5.2.1.4

No Bill View Feedback Task List Link Computer Database Info Feedback Chat

12:00 PM 12/12/2000

Administrative main page

Add a new available course:

Course name:		Course section:	
Course term:		Classroom:	
Professor name:		Course capacity:	
Course day:		Course time:	
Lab day:		Lab time:	
Course name:		Pre-enrollment:	1 - yes; 0 - no

Click on the course name to see the details of the course.

Course Pre-enrollment:	1.	2.
3.	4.	5.

Local network

Test function: add a new course

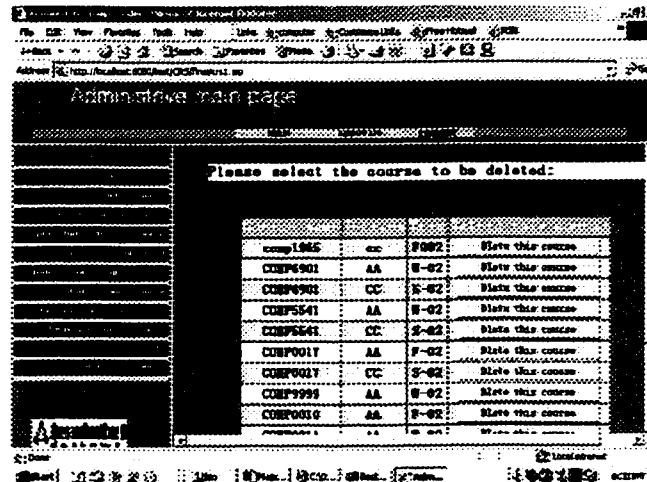
Rationale: To test the correct operation of adding a course.

Case: Upon clicking on the “add a new course” button, a form with input text boxes for course name, course section and course term course day, course time etc will appear on the page. The course name, course section, course term, course day, course time and course capacity are necessary information. If any necessary information of the course is not filled, the page will not proceed to the next page, the related field name will be in red color which indicates that the specific information should be filled or corrected. The message “Please fill in correct information” will be provided. After filling the form, click the submit button, this course will be added into the available courses table and the

message “ the course has been added” will be provided in a new page with all of this course’s information displayed.

5.2.1.5

Test function: delete a course



Rationale: To test the correct operation of deleting a course.

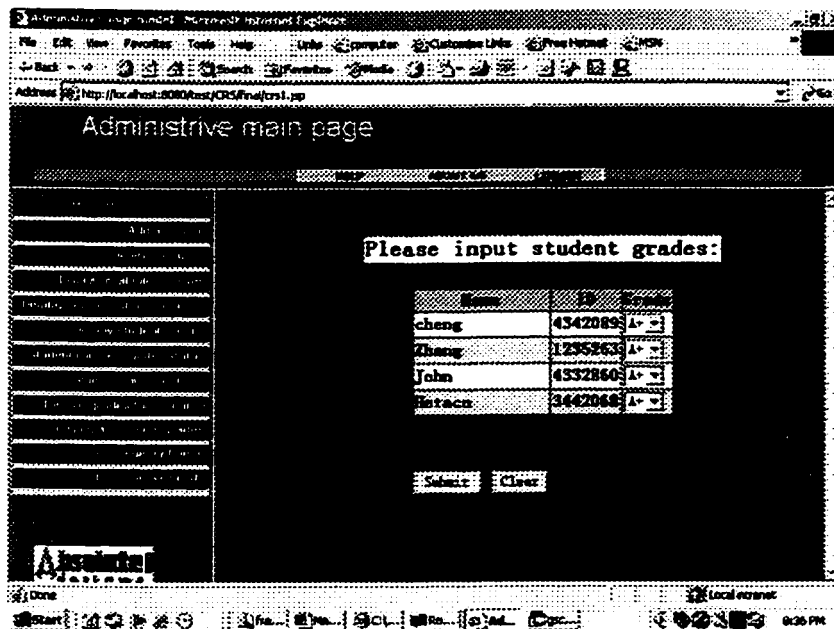
Case: Upon clicking on the “delete a course” button, a form with input text boxes for course name, course section and course term will appear on the page. After inputting all these necessary information. If any necessary information of the course is not filled, the page will not proceed to the next page, the related field name will be in red color which indicates that the specific information should be filled or corrected. The message “Please fill in correct information” will be provided. After filling the form, click the submit button, this course will be deleted from the available courses table and the message “ the course has been deleted” will be provided in a new page.

5.2.2 Test Cases for the Registration Module

In this section the test cases of the functionality of the registration module is presented.

5.2.2.1

Test function: record student grade for a course



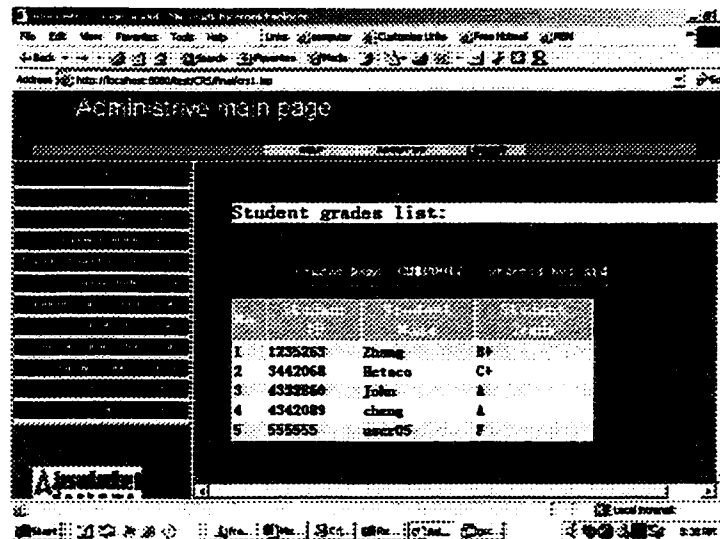
Rationale: To test the correct operation of recording student grades for a course.

Notes: All student grades will be recorded and the course will be deleted from the avacourse table. In this function there is a form with a set of mandatory fields to be field by the user then to be confirmed (submit).

Case: When click the “record grades for a course” button, a form with input text boxes for course name, course section and course term will appear on the page. After inputting all these necessary information, click the submit button, the student list of this course will display in a new HTML page with the course name, course section and course term information. Every student name and student id will display with a select box for inputting the grad for this student. The available grade values are A+, A, A-, B+, B, B-, C+, C, C-, D and F. If any necessary information of the course is not input, the message “Please provide the course information” will be provided. If the course grades has been recorded already, the message “The grades of this course has been recorded already” will be provided. After selecting grade for every student, click submit button, the student grades will be recorded into student record. This can be check by “display student grades for a course” function.

5.2.2.2

Test function: display student grades of a course

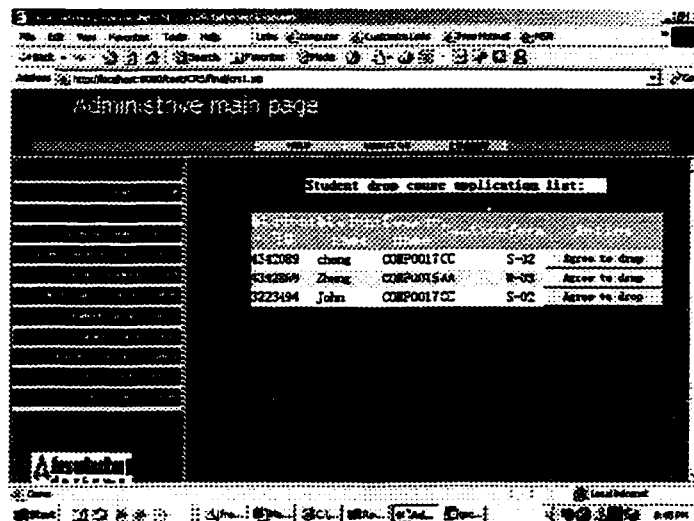


Rationale: To test the correct operation of displaying student grades for a course.

Case: Upon clicking on the “Display grades for a course” button, a form with input text boxes for course name, course section and course term will appear on the page. After inputting all these necessary information, click the submit button, the student grades of this course will display in a new HTML page with the course name, course section and course term information. If any necessary information of the course is not filled, the page will not proceed to the next page, the related field name will be in red color which indicates that the specific information should be filled or corrected. The message “Please fill in correct information” will be provided.

5.2.2.3

Test function: drop course for a student



Rationale: To test the correct operation of dropping course for a student.

Notes: We assume the student is valid student and the registration officer get the course dropping application from the student.

Case: Upon clicking on the “drop course for a student” button, the student’s application list with student name, student id, the course name, course section and course term will be listed on the page. Attached with every student name, there is a button. By clicking the submit button, the selected student’s application will be accepted. A new page will list all accepted student applications with the message “The applications have been accepted”. The student name will be deleted from the corresponding course register list and the course’s register student number will be deducted. This can be shown by “check the available course” function. At the same time, the accepted application will not appear again in the student’s application list. This can be shown by re-clicking on the “drop course for a student” button.

5.2.2.4

Test function: display student list of a course

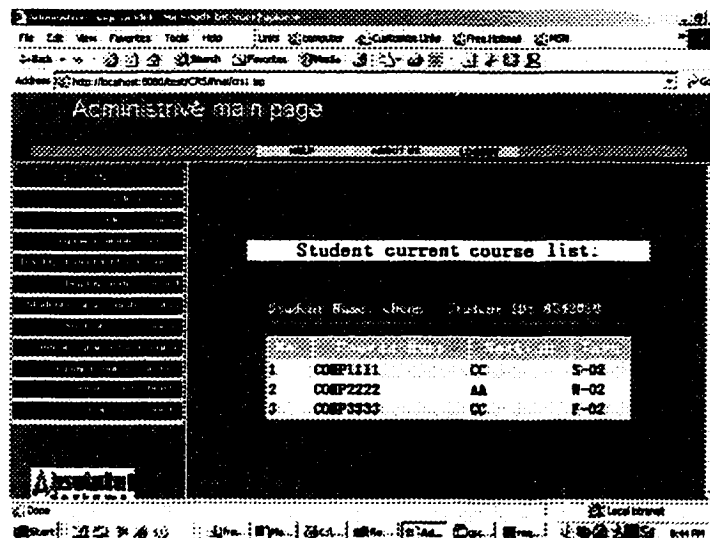


Rationale: To test the correct operation of displaying student list of a course.

Case: Upon clicking on the “display student list of a course” button, a form with input text boxes for course name, course section and course term will be displayed. All these information is required to be complete. If any information is not filled, the page will not proceed to the next page, the related field name will be in red which indicates that the specific information should be filled or corrected. If the input information is not a current course, a message will be provided on a new page, that is “Can not find this course”. Otherwise, the student list of this course will displayed on a new page.

5.2.2.5

Test function: display a student’s current course list



Rationale: To test the correct operation of displaying a student's current course list.

Case: Upon clicking on the “display a student's current course list” button, a form with input text boxes for student name and student id will be displayed. All these information is required to be complete. If any information is not filled, the page will not proceed to the next page, the related field name will be in red which indicates that the specific information should be filled or corrected. If the input information is not a valid student, a message will provided on a new page, that is “Can not find this student”. Otherwise, the student course list will displayed on a new page.

5.2.2.6

Test function: display a student's transcript

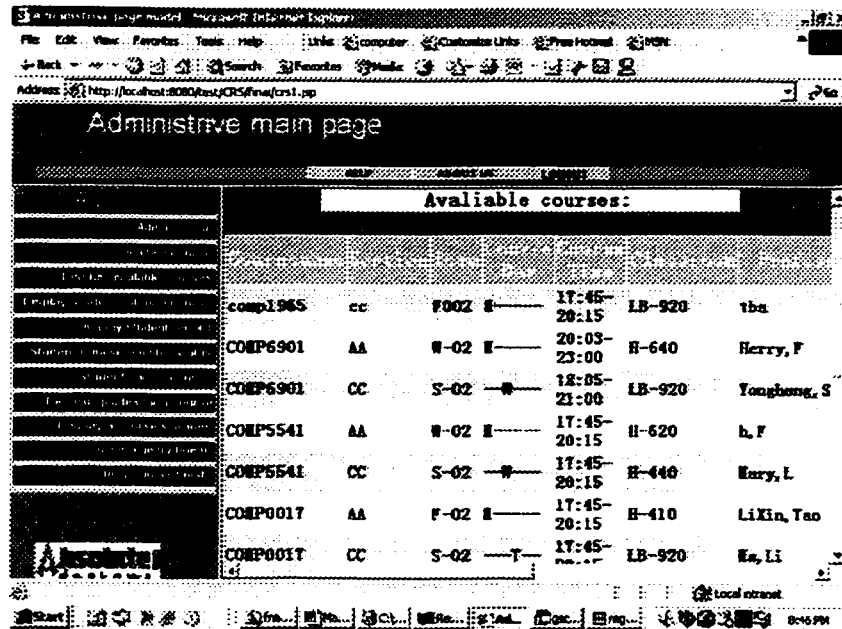
The image is a screenshot of a web browser window. The browser's address bar shows a URL starting with 'http://localhost:8080/'. The page title is 'Administrative main page'. On the left side, there is a vertical navigation menu with several links, including 'Home', 'About Us', 'Contact Us', 'Privacy Policy', 'Terms of Service', 'FAQ', 'Help', 'Admin', 'User', 'Course', 'Section', 'Student', 'Teacher', 'Enrollment', 'Grade', 'Transcript', 'Report', 'Setting', and 'Logout'. The main content area of the page is titled 'Display student record:'. Below this title, there are two input fields: 'Student name:' and 'Student id:'. To the right of these fields are two buttons: 'Submit' and 'Clear'. The browser's status bar at the bottom shows 'Local Internet' and the time '8:45 PM'.

Rationale: To test the correct operation of displaying a student's transcript.

Case: Upon clicking on the “display a student's transcript” button, a form with input text boxes for student name and student id will be displayed. All these information is required to be complete. If any information is not filled, the page will not proceed to the next page, the related field name will be in red which indicates that the specific information should be filled or corrected. If the input information is not a valid student, a message will provided on a new page, that is “Can not find this student”. Otherwise, the student's transcript will displayed on a new page with the student's name, student id, all taken course name, course term and grade.

5.2.2.7

Test function: display available courses



The screenshot shows a web browser window titled 'Administrative main page'. The address bar displays 'http://localhost:8080/test/CRS/fin/crs1.jsp'. The page content includes a table titled 'Available courses:' with the following data:

Course ID	Section	Term	Day	Time	Room	Instructor
comp1965	cc	F002	W	17:45-20:15	LB-920	tba
COMP6901	AA	W-02	W	20:03-23:00	H-640	Herry, F
COMP6901	CC	S-02	W	18:05-21:00	LB-920	Yanghong, S
COMP5541	AA	W-02	W	17:45-20:15	H-620	h, F
COMP5541	CC	S-02	W	17:45-20:15	H-440	Herry, L
COMP0017	AA	F-02	W	17:45-20:15	H-410	LiXin, Tao
COMP0017	CC	S-02	T	17:45-20:15	LB-920	Ma, Li

Rationale: To test the correct operation of displaying all available courses.

Notes: In this model, all current courses are defined as available courses, no matter if there are still any available positions in this course.

Case: Upon clicking on the “Display available courses” button, the available courses list will display in a new HTML page with all information about the course.

5.2.3 Test Cases for the online registry module

In this section the test cases of the functionality of the student online register module is presented.

5.2.3.1

Test function: check personal student record

Rationale: To test the correct operation of checking a student’s record.

Case: Upon clicking the “check personal student record” button, all courses that have been taken by this student will be listed with course name, taken term and grade. For a new student, a message like “there is no student record for you” will be provided.

5.2.3.2

Test function: check available courses

Rationale: To test the correct operation of displaying all available courses for student to select.

Notes: In this model, all current courses are defined as available courses, but only the course that is not full can be selected by the student to register.

Case: Upon clicking on the “Check available courses” button, the available courses list will display in a new HTML page with all information about the course. For every course that has available positions for the student, a “put into register form” button is attached with it. If the course is full, then there is not a button attached with it.

5.2.3.3

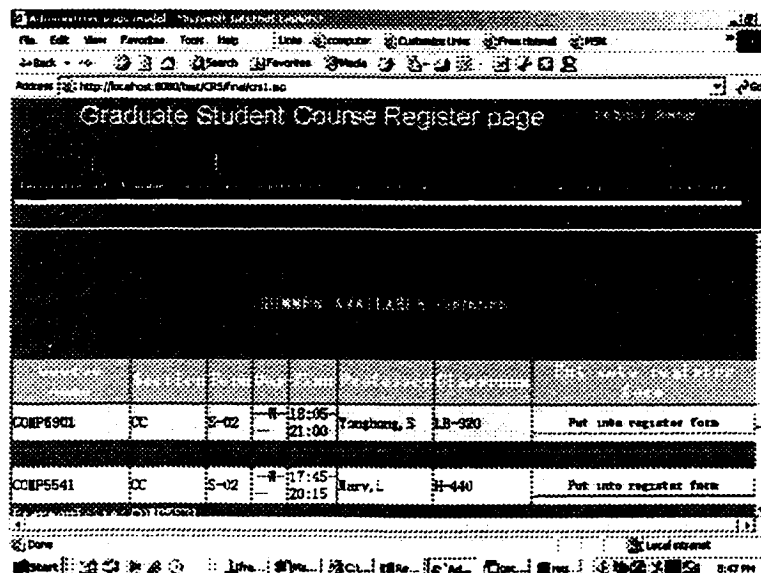
Test function: check course information

Rationale: To test the correct operation of displaying all information of a course.

Case: Upon clicking on the “Check course information” button, all course name will be list on the page. Clicking on the course name, a new small window will pop out to display information(course introduction, course prerequisite and course credit) about this course and a link to the main page of this course.

5.2.3.4

Test function: select a course and put into register form



Rationale: To test the correct operation of selecting a course from the available course list and put it into register form.

Notes: we assume that before entering this module, student's ID and password has been verified and accepted. By checking the student's "level" and his/her current register status, the capacity of the student register form is assigned. For full time student, 4 courses are allowed to register in a term, for part time student, 1 course a term is allowed. If the student does not meet a course's prerequisite, this course can not be put into his/her register form by this student. A student can not take a course more than once.

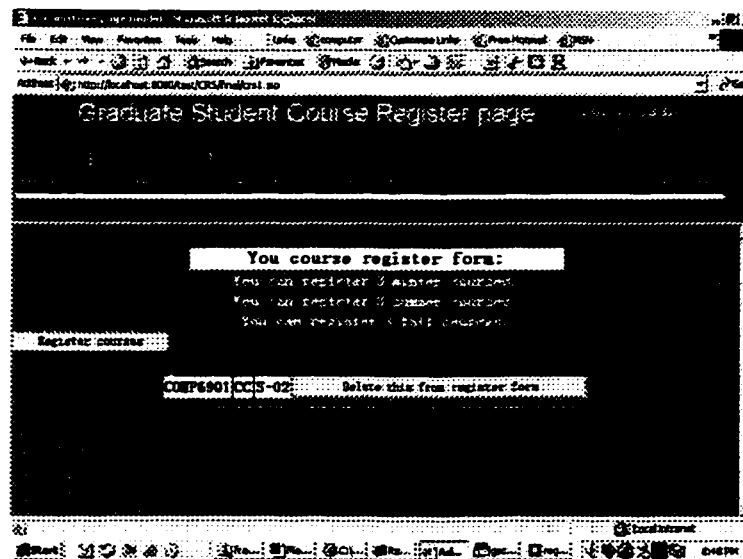
Case: Login by a full time student who has not registered any courses, the student can put maximum 4 courses into his/her register form. If the student didn't take a course's prerequisite courses or the grade is not good enough(below 2.7), when he/she click the "put into register form" button attached with this course, the message "you are not allowed to register this course because you do not meet the prerequisite condition" will appear in a pop up window and the selected course will not appear in the register form.. Otherwise, the student's register form will display in a new HTML page and the selected course name will appear in the register form.

If the student's register form is full, when he/she click the "put into register form" button attached with this course, the message "Your register form is full" will appear in a pop up window and the selected course will not appear in the register form.

When the student select a course more that once or select a course he/she has taken before, when he/she click the "put into register form" button attached with this course, the message "You have taken this course before" will appear in a pop up window and the selected course will not appear in the register form.

5.2.3.5

Test function: check/update register form



Rationale: To test the correct operation of checking register form.

Case: By clicking the “check/update register form” button, the student’s register form will be displayed in a new page with the “submit” button at the bottom of the page. All selected courses(name, section and term) are display in the form. Attached with every course, there is a “remove” button. Upon click on a “remove” button, a new page will display with the updated register form. The removed course will not appear in the new register form.

5.2.3.6

Test function: register selected courses

Rationale: To test the correct operation of register selected courses.

Case: Upon clicking on the “submit” button on the register form, a new page with the registered course name list and the message “The following courses have been registered” will display.

5.2.3.7

Test function: check personal current courses

Rationale: To test the correct operation of checking a student’s current courses.

Case: Upon clicking the “check personal current courses” button, all courses that have been registered by the student will be listed on the page.

5.2.3.8

Test function: apply for dropping course

Rationale: To test the correct operation of application for dropping a course

Case: Upon clicking the “Drop courses” button, all courses that have been registered by the student will be listed on the page. With every course name that is not been applied for dropping, there is an “Apply for drop” button. If the drop course application has been sent with a course, no “Apply for drop” button appears with this course. If the registry staff has accepted the drop course application, the course will not appear on the page. By clicking the “Apply for drop” button, the application for drop course will be sent for this course.

6 Conclusion and Future Work

6.1 Conclusion

Separating presentation layer from the logic layer completely is a good design approach. No more HTML code inside Java code. Java programmers work with pure Java code, so debugging is much easier. Graphic designers work at HTML with more flexibility. Even the company has to pay more for graphic designers and maybe it needs more graphic programmers; the products are for sure more dynamic. Moreover, the product can be expanded to other devices, such as PDA, WAP, etc.

The work flow for common used JSP page (Java code mix with HTML presentation) is, first, graphic designer created a HTML page with static information (for example, “display available course name”); second, Java programmer replaced the static information with dynamic information (read available courses information from database). Therefore, programmers design business logic, graphic designers design a presentation (HTML), and they will be merged together. Most of dynamic web sites use this way, because it’s easy and straightforward.

The difficulties of this project are in the design period. A good design can make the following work easier. I spent many time in designing the database. In the application, I used not only the Java bean and also the JavaScript technology to deal with the logic. Sometimes, it is not easy to find a straightforward way to achieve the design goal. I had to try to use different way and finally selected the best one.

From this project, I have learned the technology of using HTML and JSPs, and designed the architecture of separating the business logic from presentations in web application.

6.2 Future work

6.2.1 Add functionalities

By using the present project, the different kinds of users may find that some useful functions are not provided by this project, so in the future many functions will need to be added to this system.

6.2.2 Add course time and classroom management subsystem

As a course registry management system, this system needs to expend its functionalities in managing course time and classroom when add a new course. By adding course time and classroom management subsystem, this project will become a

6.2.3 Generate a template web application system

For any software developer, software reuse is an important issue. Software reuse can reduce the cost of a project. In this project, JSP technology is used. Java language is the perfect language to write reusable software, this makes the software of this project be possible as a template of web register application system. In fact, this template also can be used in e-business web site by online shopping company. When creating a new online application web site, simply generate a new database, select a suitable application logic subsystem and a view subsystem and put everything to a web server.

7 References

1. Larnie Pekowsky, *JavaServer Pages*
2000, Publishing by Addison-Wesley
2. Mark H. Butler, *Current Technologies For Device Independence*
<http://www.hpl.hp.co.uk/people/marbut/currTechDevInd.htm>
3. Sun's Java Servlet Technology
<http://java.sun.com/products/servlet/>
4. James Goodwill, *Developing Java Servlets*,
1999, Sams Publishing
5. Sun's Java Server Pages technology
<http://java.sun.com/products/jsp/>
6. Sun Java J2SE 1.3
<http://java.sun.com/j2se/1.3/>
7. Sun Java J2EE
<http://java.sun.com/j2ee>
8. Resin Server download site
<http://www.caucho.com/download/index.xtp>

8 Abbreviations

CRS	Course Register System
ASP	Active Server Page
API	Application Interface
CGI	Common Gateway Interface
DOM	Document Object Model
DSL	Digital Subscriber Line
DSSSL	Document Style Semantics and Specification Language
EJB	Enterprise Java Bean
GUI	Graphic User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
J2SE	Java 2 Standard Edition
J2EE	Java 2 Enterprise Edition
JAXP	Java API for XML Processing
JDBC	Java Database Connectivity
JDK	Java Development Kit
JMS	Java Message Service
JNDI	Java Naming and Directory Interface
JSP	Java Server Page
MVC	Model-View-Controller
PC	Personal Computer
PDA	Personal Digital Assistants
PDF	Portable Document Format
PHP	PHP: Hypertext Preprocessor
SAX	Simple API for XML
SGML	Standard Generalized Markup Language
SQL	Standard Query Language
SVG	Scalable Vector Graphics

Appendices:

Appendix A: Sample JSP source files

A-1. Display available courses

```
<%@ page language="java" import="java.sql.*" %>
<%@ page import="gscrsbean.prerequisite" %>
<%@ page import="gscrsbean.avacourse" %>
<jsp:useBean id="avacoursebean" scope="page" class="gscrsbean.avacourse" />
<jsp:useBean id="prerequisitebean" scope="page" class="gscrsbean.prerequisite" />
<jsp:useBean id="workM" scope="page" class="gscrsbean.dbacc" />
<jsp:useBean id="userSection" scope="session" class="gscrsbean.tmsection" />
<HTML>
<HEAD><TITLE>Available courses page</TITLE></HEAD>
<BODY text="white" link=#0000ff bgColor=#000000 background=images\background.gif>
<BR>< CENTER >
<TABLE border="0" cellpadding="1" cellspacing="0" width="100%" align="center">
<TR><TD class=small bgColor=#ffffd5><B class=sans>
<FONT color=yellow>AVAILABLE COURSES</FONT></B>
<BR><BR>
</TD></TR></CENTER>
<BR>
<TABLE align=center>
<TR bgColor=#aaaaaa><B><font face=arial size=-2><TH>Course      Name</TH><TH>Section</TH>
<TH>Term</TH><TH>Day</TH><TH>Time</TH><TH>Professor</TH><TH>Classroom</TH><TH>Put
ut into register form</TH></TR>

<%
    ResultSet RS = avacoursebean.executeQuery("SELECT
avacourse.cname,avacourse.csection,avacourse.ctrm,avacourse.cday,avacourse.ctime,avacourse.croom,ava
course.pname,avacourse.capacity,avacourse.snumber,avacourse.lday,avacourse.ltime,avacourse.tname,avac
ourse.precourse FROM avacourse WHERE avacourse.snumber<avacourse.capacity");
    String t1;
    String t2;
    String t3;
    String t4;
    String t5;
    String t6;
    String t7;
    String t8;
    String t9;
    String t10;
    String t11;
    String t12;
    String t13;
    String t14;
    String precou;
```

```

        int i=0;
while (RS.next()) {
%>
<FORM action="mainr2-1.jsp" method="POST">
<%
    i++;
    t1=RS.getString("cname");
    t2=RS.getString("csection");
    t3=RS.getString("cterm");
    t4=RS.getString("cday");
    t5=RS.getString("ctime");
    t6=RS.getString("croom");
    t7=RS.getString("pname");
    t8=RS.getString("capacity");
    t9=RS.getString("snumber");
    t10=RS.getString("lday");
    t11=RS.getString("ltime");
    t12=RS.getString("tname");
    t13=RS.getString("precourse");
    int pre=Integer.parseInt(t13);
    if(pre==1)
        t14="yes";
    else t14="no";

    if(i%2==1){
out.print("<TR bgColor=#cccccc><TD><input TYPE=textbox size=8 NAME=cname VALUE=\"" + t1 +
"></TD><TD><input TYPE=textbox size=3 NAME='csection' VALUE=\"" + t2+ "></TD> <TD><input
TYPE=textbox size=6 NAME='cterm' VALUE=\"" + t3+ "></TD><TD><input TYPE=textbox size=6
NAME='cday' VALUE=\"" + t4+ "></TD><TD><input TYPE=textbox size=10 NAME='ctime' VALUE=\""
+ t5+ "></TD><TD><input TYPE=textbox size=10 NAME='pname' VALUE=\"" + t7+ "><TD><input
TYPE=textbox size=6 NAME='croom' VALUE=\"" + t6+ "></TD><TD><b><font size=2
color=darkblue><INPUT type=submit value='Put into register form'></TD></TR>");
    }
    else if(i%2==0){
out.print("<TR bgColor=#eeeecc><TD><input TYPE=textbox size=8 NAME=cname VALUE=\"" + t1 +
"></TD><TD><input TYPE=textbox size=3 NAME='csection' VALUE=\"" + t2+ "></TD> <TD><input
TYPE=textbox size=6 NAME='cterm' VALUE=\"" + t3+ "></TD><TD><input TYPE=textbox size=6
NAME='cday' VALUE=\"" + t4+ "></TD><TD><input TYPE=textbox size=10 NAME='ctime' VALUE=\""
+ t5+ "></TD><TD><input TYPE=textbox size=10 NAME='pname' VALUE=\"" +
t7+ "></TD><TD><input TYPE=textbox size=6 NAME='croom' VALUE=\"" +
t6+ "></TD><TD><b><font size=2 color=darkblue><INPUT type=submit value='Put into register
form'></TD></TR>");
    }
    //=====check prerequisite=====
    if(pre==1){
ResultSet RS1 = prerequisitebean.executeQuery("SELECT prerequisite.pcname FROM prerequisite
WHERE prerequisite.cname='"+t1+"'");
out.print("<tr><td><b><font size=2 color=white>Prerequisite:</td>");
while (RS1.next()) {
    precou=RS1.getString("pcname");
    out.print("<td><input TYPE=textbox size=8 NAME=precou VALUE=\"" + precou+ "></td>");
}
}
%>
</tr>
</FORM>
<%

```

```

    }
    RS1.close();
    %>

<TR>
  <TD colSpan=4><IMG height=1 alt=pixel src=" images\grey-pixel.gif" width="100%" align=top
vspace=6>
  </TD>
  <TD colSpan=4><IMG height=1 alt=pixel src=" images\grey-pixel.gif" width="100%" align=top
vspace=6>
  </TD>
</TR>
</FORM>
<%
  }

  RS.close();
  %>
</TD></TR>
</TABLE>
</CENTER>
</BODY></HTML>

```

A-2 select a course and put into register form

```

<%@ page language="java" import="java.sql.*" %>
<%@ page import="gscrsbean.avacourse" %>
<%@ page import="gscrsbean.reglist" %>
<%@ page import="gscrsbean.sturecord" %>
<%@ page import="gscrsbean.regform" %>
<jsp:useBean id="avacoursebean" scope="page" class="gscrsbean.avacourse" />
<jsp:useBean id="reglistbean" scope="page" class="gscrsbean.reglist" />
<jsp:useBean id="sturecordbean" scope="page" class="gscrsbean.sturecord" />
<jsp:useBean id="workM" scope="page" class="gscrsbean.dbacc" />
<jsp:useBean id="userSection" scope="session" class="gscrsbean.tmsection" />
<jsp:useBean id="regformbean" scope="session" class="gscrsbean.regform" />
<HTML>
<HEAD>
<TITLE>Content</TITLE>
</HEAD>
<BODY text=yellow link=#0000ff bgColor=#FFFFFF background=images\background.gif>

<br>
<table align="center">
  <br> <tr><td bgcolor=#eeeecc align="center">
    <b><font size=3 color="darkblue">You have put the following courses into your register form:</b>
  </td></tr>
</table>
<br>
<FORM method="POST" action="mainrs2.jsp">
<br>

```

```

<table align="center">
<%
String sname=userSection.getUserName();
String sid=userSection.getUserPassword();
String cname=request.getParameter("cname");
String csection=request.getParameter("csection");
String cterm=request.getParameter("cterm");
String [] precous=request.getParameterValues("precou");
double GPA=0; //average GPA for a course
double count=0; //total GPA
int mark=0; //is there a prerequisite has not been taken? 0:no. 1:yes.
double pregpa=0; //gpa of a prerequisite course
String precou;
String DEC="";
if(regformbean.getStatus()==0)
    out.print("<tr><td width=500 align=center><b><font size=2 color= white>You are not allowed to
register more course!</td></tr>");
    else {
//=====check repeat status=====
        if((reglistbean.searchcourse(cname,sid)==0)||((sturecordbean.searchcourse(cname,sid)!=0))
ResultSet RS = sturecordbean.executeQuery("SELECT sturecord.grade FROM sturecord WHERE
sturecord.cname='"+ cname +" AND sturecord.sid='"+ sid +"'");
ResultSet RS2 = reglistbean.executeQuery("SELECT * FROM reglist WHERE reglist.cname='"+ cname
+" AND reglist.sid='"+ sid +"'");
        String t1;
        int c=0;
        while (RS.next()) {
            c++;
            t1=RS.getString("grade");
        }
        RS.close();
        while (RS2.next()) {
            c++;
        }
        RS2.close();
        if(c!=0)
            out.print("<tr><td align=center width=500><b><font size=2 color= white>You have registered this
course before!</td></tr>");
        else {
//-----check prerequisite-----
            if (precous == null)
                { //no pre
                    int result=regformbean.addCourse(cname, csection, cterm);
                    if(result==1)
                        out.print("<tr><td width=500 align=center><b><font size=2 color= white>Your register form is
currently full!</td></tr>");
                    if(result==2)
                        out.print("<tr><td width=500 align=center><b><font size=2 color= white>This course has already in
your register form!!</td></tr>");
                    else{
                        int st=regformbean.getStatus();
                        for(int i=0;i<st;i++)
                        {
                            cname=regformbean.getCourseName(i);
                            csection=regformbean.getCourseSection(i);
                            cterm=regformbean.getCourseTerm(i);

```

```

        if(cname!="")
            out.print("<TR bgColor=#aaaaaa><B><font face=arial size=-2><TH>Course
name</TH><TD>"+cname+"</TD>
<TH>csection</TH><TD>"+csection+"</TD><TH>Term</TH><TD>"+cterm+"</TD></TR>");
    }
    }
    }
    else{
        int cnum=precous.length; //number of prerequisite courses
        String pregrade="";
        for (int i = 0; i <cnum; i++)
        {
            precou=precous[i];
            ResultSet RS1 = sturecordbean.executeQuery("SELECT sturecord.grade FROM sturecord
WHERE sturecord.cname='"+ precou +" AND sturecord.sid='"+ sid +"");
            String t2;
            int j=0;
            while (RS1.next())
            {
                j++;
                pregrade=RS1.getString("grade");
            }
            if(j==0)
                mark=1;
            RS1.close();
            if(pregrade.equals("A+")) pregpa=4.3;
            else if(pregrade.equals("A")) pregpa=4.0;
            else if(pregrade.equals("A-")) pregpa=3.7;
            else if(pregrade.equals("B+")) pregpa=3.3;
            else if(pregrade.equals("B")) pregpa=3.0;
            else if(pregrade.equals("B-")) pregpa=2.7;
            else if(pregrade.equals("C+")) pregpa=2.3;
            else if(pregrade.equals("C")) pregpa=2.0;
            else if(pregrade.equals("C-")) pregpa=1.8;
            else if(pregrade.equals("D")) pregpa=1.5;
            else pregpa=0.0;
            count+=pregpa;
        }
        if(mark==1)
            out.print("<tr><td width=500 align=center><b><font size=2 color= white>You have not taken all the
prerquisity course!</td></tr>");
        else{
            GPA=count/cnum;
            if(GPA<2.7)
                out.print("<tr><td width=500 align=center><b><font size=2 color= white>You prerquisity ourse GPA
doesn't meet the requirement!</td></tr>");
            else{
                int result=regformbean.addCourse(cname, csection, cterm);
                if(result==1)
                    out.print("<tr><td width=500 align=center><b><font size=2 color= white>Your register form is
currently full!</td></tr>");
                if(result==2)
                    out.print("<tr><td width=500 align=center><b><font size=2 color= white>This course has already in
your register form!!</td></tr>");
                else{
                    int st=regformbean.getStatus();

```

```

        for(int i=0;i<st;i++)
        {
            cname=regformbean.getCourseName(i);
            csection=regformbean.getCourseSection(i);
            cterm=regformbean.getCourseTerm(i);
            if(cname!="")

out.print("<TR bgColor=#aaaaaa><B><font face=arial size=-2><TH>Course
name</TH><TD>"+cname+"</TD>
<TH>Section</TH><TD>"+csection+"</TD><TH>Term</TH><TD>"+cterm+"</TD></TR>");
        } } } } }
    %>
</td></tr>
</table>
<table align=center>
<br> <tr><td><INPUT type="submit" value="Continue to select course"></td></tr>
</table>
</FORM>
<FORM method="POST" action="mainsr3.jsp">
<table align=center>
<br><tr><td><INPUT type="submit" value="Send or edit register form"></td></tr>
</table>
</FORM>
</BODY></HTML>

```

A-3 register the courses

```

<%@ page language="java" import="java.sql.*" %>
<%@ page import="gscrsbean.reglist" %>
<jsp:useBean id="reglistbean" scope="page" class="gscrsbean.reglist" />
<jsp:useBean id="workM" scope="page" class="gscrsbean.dbacc" />
<jsp:useBean id="userSection" scope="session" class="gscrsbean.tmsection" />
<jsp:useBean id="regformbean" scope="session" class="gscrsbean.regform" />
<HTML>
<HEAD>
<TITLE>Content</TITLE>
</HEAD>
<BODY text="#FFFFFF" link=#0000ff bgColor=#FFFFFF background=images\background.gif>
<br>
<table border="0" cellpadding="0" cellspacing="0" width="400" align="center">
<tr><td width=100% bgcolor=#003399 align=nowrap >
    <b> - You have register the following courses:</b>
</tr>
</table>
<%
String sname=userSection.getUserName();
String sid=userSection.getUserPassword();
int level=userSection.getUserLevel();
String cname;
String csection;
String cterm;

```

```

int s=regformbean.isEmpty();
if(s==0)
    out.print("<tr><td width=400>Your register form is currently empty!</td></tr>");
else{
    int st=regformbean.getStatus();
    for(int i=0;i<st;i++)
    {
        cname=regformbean.getCourseName(i);
        csection=regformbean.getCourseSection(i);
        cterm=regformbean.getCourseTerm(i);
        if(cname!=""){
            reglistbean.createreglist(cname,sname,sid,csection,cterm,0);
            int re=regformbean.deleteCourse(cname,csection,cterm);
        }
        out.print("<tr><td width=100>" + cname + "<td width=100>" + csection + "<td width=100>" + cterm);
        regformbean.updateStatus();
    }
}
}%>
</TABLE>
</BODY></HTML>

```

A-4 record grades for a course

```

<%@ page language="java" import="java.sql.*" %>
<%@ page import="gscrsbean.reglist" %>
<%@ page import="gscrsbean.temp" %>
<jsp:useBean id="reglistbean" scope="page" class="gscrsbean.reglist" />
<jsp:useBean id="workM" scope="page" class="gscrsbean.dbacc" />
<jsp:useBean id="tempbean" scope="session" class="gscrsbean.temp" />

<HTML>
<HEAD><TITLE>Content mainad10-1</TITLE>
<META http-equiv=Content-Type content="text/html; charset=gb2312">
</HEAD>
<BODY text="#FFFFFF" link="#0000ff" bgColor=#FFFFFF background=images/background.gif>
<P>&nbsp;</P>
<UL>
<!--Begin of different code -->
<!--Begin of different code -->
<table border="0" cellpadding="0" cellspacing="0" width="100%" align="center" valign="bottom">
<FORM action="mainad10-1-1.jsp" method="POST">
<table border="0" cellpadding="0" cellspacing="0" width="100%" align="center">
    <tr><td bgcolor=#eeeecc align="center">
        <b><font size=4 color="darkblue">Please input student grades:</b>

```



```
</td></tr>
</table>
<br>
<table border="0" cellpadding="0" cellspacing="0" width="100%" align="center">
<%
String DES="";
String cname=request.getParameter("cname");
String csection=request.getParameter("csection");
String cterm=request.getParameter("cterm");
int i=0;

//=====check whether all necessary parameters are available=====
if(cname.equals(DES))
    out.print("<br><center>Please provide course name!</center><br><br><br>");
else if(csection.equals(DES))
    out.print("<br><center>Please indicate the course section!</center><br><br><br>");
else if(cterm.equals(DES)))
    out.print("<center><br><br><br>Please indicate the course term!</center><br><br><br>");
else {
    tempbean.setSTR1(cname);
    tempbean.setSTR2(csection);
    tempbean.setSTR3(cterm);
    ResultSet RS = reglistbean.executeQuery("SELECT reglist.sid,reglist.sname FROM reglist WHERE
    reglist.cname='"+ cname +"' AND reglist.csection='"+ csection +"' AND reglist.cterm='"+ cterm +"'");
    String t1;
    String t2;
out.print("<tr><td><b><font size=2 color=yellow>Course Name: <font size=2 color=white>" + cname
+"<font size=2 color=yellow>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~Section: <font size=2 color=white>" + csection +"<font
size=2 color=yellow>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~Term: <font size=2 color=white>" + cterm +" </td></tr>");
%>
<TR>
    <TD colSpan=4><IMG height=1 alt=pixel src=" images/grey-pixel.gif" width="100%" align=top
vspace=6>
    </TD>
</TR>
<%
    while (RS.next()) {
        i++;
        t1=RS.getString("sname");
        t2=RS.getString("sid");
out.print("<tr><td>"+i+"<input TYPE=textbox NAME=sname VALUE="+ t1 + ">"+<input
TYPE=textbox NAME=sid VALUE="+ t2+"></td></tr>");
%><tr><td><select name="grade">
    <option selected>A+
    <option>A <option>A- <option>B+ <option>B <option>B- <option>C+ <option>C
    <option>C- <option>D+ <option>D <option>D- <option>F
    </select></td></td>
<%
    }

    RS.close();
    if(i==0)
        out.print("<tr><td><b><font size=2 color=red>This course is not a current course! </td></tr>");
}

```

```

if(i!=0){
%>
</td></tr>
<tr><td align="center"><INPUT type="submit" value="Submit">
<INPUT type="reset" value="Clear"></td></tr>
<%
}
%>
</TABLE>
</FORM>
</TABLE>
</BODY></HTML>

```

Appendix B: Sample JavaBean

B-1 dbacc bean

```
package gscrsbean;
import java.sql.*;
public class dbacc {
    String sDBDriver = "org.gjt.mm.mysql.Driver";
    String sConnStr = "jdbc:mysql://localhost/gscrs";
    Connection conn = null;
    ResultSet rs = null;
    public dbacc() {
        try {
            Class.forName(sDBDriver);
        }
        catch(java.lang.ClassNotFoundException e) {
            System.err.println("dbacc(): " + e.getMessage());
        }
    }
    public ResultSet executeQuery(String sql)    {
        rs = null;
        try {
            conn = DriverManager.getConnection(sConnStr);
            Statement stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
        }
        catch(SQLException ex) {
            System.err.println("dbacc.executeQuery: " + ex.getMessage());
        }
        System.err.println("Run Query:");
        if(rs==null)
            System.err.println("Run Query null:");
        else
            System.err.println("Run Query ok:");
        return rs;
    }
    public void executeUpdate(String update)
    {
        try {
            conn = DriverManager.getConnection(sConnStr);
            Statement stmt = conn.createStatement();
            stmt.executeUpdate(update);
        }
        catch(SQLException ex) {
            System.err.println("dbacc.executeUpdate: " + ex.getMessage());
        }
    }
}
```

B-2 avacourse bean

```
package gscrsbean;
import java.sql.*;
public class avacourse {
    String cname;
    String csection;
    String cterm;
    String croom;
    String pname;
    int capacity;
    int snumber;
    String ctime;
    String cday;
    String ltime;
    String lday;
    String tname;
    int prerequisite;
    String pre1;
    String pre2;
    String pre3;
    String pre4;
    String pre5;
    String sDBDriver = "org.gjt.mm.mysql.Driver";
    String sConnStr = "jdbc:mysql://localhost/gscrs";
    Statement stmt = null;
    ResultSet rs = null;

    public avacourse() {
        try {
            Class.forName(sDBDriver);
        }
        catch(java.lang.ClassNotFoundException e) {
            System.err.println("mylogin(): " + e.getMessage());
        }
    }

    public int createavacourse(String cname,String csection,String cterm,String cday,String ctime,String
    croom,String pname, int capacity,int snumber, String lday,String ltime, String tname,int prerequisite) {
        int count=0;

        try {
            if ( rs != null ) rs.close();
            if ( stmt != null ) stmt.close();
            Connection conn = DriverManager.getConnection(sConnStr);
            stmt = conn.createStatement();
            StringBuffer query=new StringBuffer(300);
            query.append("INSERT INTO avacourse VALUES ( " );
            query.append(cname);
            query.append(" " );
            query.append(csection);
            query.append(" " );
            query.append(cterm);
```

```

        query.append(" ", "");
        query.append(cday);
        query.append(" ", "");
        query.append(ctime);
        query.append(" ", "");
        query.append(croom);
        query.append(" ", "");
        query.append(pname);
        query.append(" ", "");
        query.append(capacity);
        query.append(" ", "");
        query.append(snumber);
        query.append(" ", "");
        query.append(lday);
        query.append(" ", "");
        query.append(ltime);
        query.append(" ", "");
        query.append(tname);
        query.append(" ", "");
        query.append(prerequisite);

        query.append(")");
        System.err.println("Query sentences is: " + query.toString() + "\n");
        stmt.executeUpdate(query.toString());
    } catch (SQLException ex) {
        System.err.println("addCourse.executeQuery: " + ex.getMessage());
        return 1;
    }
}
return 0;
}

```

```

public int defprecourse(String cname,String pre1,String pre2,String pre3,String pre4,String pre5) {
    String DES="";
    String[] A=new String[5];
    A[0]=pre1;
    A[1]=pre2;
    A[2]=pre3;
    A[3]=pre4;
    A[4]=pre5;
    int C=0; //counter used for counting the number of prerequisite courses
    for(int i=0;i<5;i++)
    {
        if(!A[i].equals(DES))
            C++;
    }

    for(int j=0;j<C;j++)
    {

```

```

int count=0;

```

```

        try {
            if ( rs != null ) rs.close();
            if ( stmt != null ) stmt.close();

```

```

        Connection conn = DriverManager.getConnection(sConnStr);
        stmt = conn.createStatement();
        StringBuffer query=new StringBuffer(300);
        query.append("INSERT INTO prerequisite VALUES ( " );
        query.append(cname);
        query.append(" , ");
        query.append(A[j]);
        query.append(" )");
        System.err.println("Query sentences is: " + query.toString() + "\n");
        stmt.executeUpdate(query.toString());
    }catch(SQLException ex) {
        System.err.println("addPreCourse.executeQuery: " + ex.getMessage());
        return 1;
    }
}
return 0;
}

```

```

public int deleteavacourse(String cname, String csection,String cterm) {
    int count=0;

    try {
        if ( rs != null ) rs.close();
        if ( stmt != null ) stmt.close();
        Connection conn = DriverManager.getConnection(sConnStr);
        stmt = conn.createStatement();
        StringBuffer query=new StringBuffer(100);
        query.append("DELETE * FROM avacourse ");
        query.append("WHERE avacourse.cname=");
        query.append(cname);
        query.append(" ");
        query.append("AND ");
        query.append(" avacourse.csection=");
        query.append(csection);
        query.append(" ");
        query.append("AND ");
        query.append(" avacourse.cterm=");
        query.append(cterm);
        query.append(" ");

        System.err.println("Query sentences is: " + query.toString() + "\n");
        stmt.executeUpdate(query.toString());
    }catch(SQLException ex) {
        System.err.println("deleteavacourse.executeQuery: " + ex.getMessage());
        return 1;
    }
}
return 0;
}

```

```

public ResultSet executeQuery(String sql) {
    try {
        if ( rs != null ) rs.close();
    }
}

```

```

        if ( stmt != null ) stmt.close();
        Connection conn = DriverManager.getConnection(sConnStr);
        stmt = conn.createStatement();

        rs = stmt.executeQuery(sql);
    }
    catch(SQLException ex) {
        System.err.println("dbacc.executeQuery: " + ex.getMessage());
    }
    System.err.println("Run Query:");
    if(rs==null)
        System.err.println("Run Query null:");
    else
        System.err.println("Run Query ok:");
    return rs;
}
}

```

B-3 regform bean

```

package gscrsbean;
import java.sql.*;
public class regform {

    String studentname;
    String studentid;
    String [] coursenamename;
    String [] coursesection;
    String [] courseterm;
    int status;
    public regform(){

    public void createRegform(String sname,String sid,int s) {

        studentname=sname;
        studentid=sid;
        status=s;
        coursenamename= new String[s];
        coursesection= new String[s];
        courseterm= new String[s];
        for(int i=0;i<s;i++)
        {
            coursenamename[i]="";
            coursesection[i]="";
            courseterm[i]="";
        }

        return;
    }

    public int getStatus(){ //how many course the student can register

```

```

        return status;
    }

    public void updateStatus(){ //change status when register a course
        status--;
    }

    public int checkStatus(){//if the student have select full course
        for(int i=0;i<status;i++)
        {
            if(coursename[i]=="")
                return i;
        }
        return -1;
    }

    public int checkRepeat(String cname){//if the course has been in the form
        for(int i=0;i<status;i++)
        {
            System.err.println("cname is"+cname);
            System.err.println("coursename"+i+" is"+coursename[i]);
            if(coursename[i].equals(cname))
                return 0;
        }
        System.err.println("Not found repeat!");
        return 1;
    }

    public int isEmpty(){//is there any course in the form
        for(int i=0;i<status;i++)
        {
            if(coursename[i]!="")
                return 1;
        }
        return 0;
    }

    public int addCourse(String cname,String csection,String cterm) {
        //put a course into register form
        int t=checkStatus();
        if(t==-1)
        {
            System.err.println("register form full!");
            return 1;
        }
        else {
            if(checkRepeat(cname)==0)
            {
                System.err.println("repeated course!");
                return 2;
            }
        }
    }

```



```

        else{
            coursename[t]=cname;
            coursesection[t]=csection;
            courseterm[t]=cterm;
            return 0;
        }
    }
}

public int deleteCourse(String cname,String csection,String cterm){
    System.err.println("begin todelete course:"+cname);
    for(int i=0;i<status;i++){
        System.err.println("course[i] name: " +coursename[i]);
        if(coursename[i].equals(cname)&&coursesection[i].equals(csection)&&courseterm[i].equals(cterm))
        { System.err.println("delete course!");
            coursename[i]="";
            coursesection[i]="";
            courseterm[i]="";
            return 0;
        }
    }
    System.err.println("Did not delete course!");
    return 1;
}

public String getCourseName(int s) {
    return coursename[s];
}

public String getCourseSection(int s) {
    return coursesection[s];
}

public String getCourseTerm(int s) {
    return courseterm[s];
}

}

```

B-3 dropcourse bean

```

package gscrsbean;
import java.sql.*;
public class dropcourse {
    String cname;

```

```

String sname;
String sid;
String csection;
String cterm;
String sDBDriver = "org.gjt.mm.mysql.Driver";
String sConnStr = "jdbc:mysql://localhost/gscrs";
Statement stmt = null;
ResultSet rs = null;

public dropcourse() {
    try {
        Class.forName(sDBDriver);
    }
    catch(java.lang.ClassNotFoundException e) {
        System.err.println("mylogin(): " + e.getMessage());
    }
}

public int createdropcourse(String sname,String sid, String cname, String csection,String cterm) {
    int count=0;

    try {
        if ( rs != null ) rs.close();
        if ( stmt != null ) stmt.close();
        Connection conn = DriverManager.getConnection(sConnStr);
        stmt = conn.createStatement();
        StringBuffer query=new StringBuffer(300);
        query.append("INSERT INTO dropcourse VALUES ( " );
        query.append(sname);
        query.append(" , ");
        query.append(sid);
        query.append(" , ");
        query.append(cname);
        query.append(" , ");
        query.append(csection);
        query.append(" , ");
        query.append(cterm);
        query.append(" )");
        System.err.println("Query sentences is: " + query.toString() + "\n");
        stmt.executeUpdate(query.toString());
    }catch(SQLException ex) {
        System.err.println("createdropcourse.executeQuery: " + ex.getMessage());
        return 1;
    }
    return 0;
}

public int deletedropcourse(String sid, String cname, String csection,String cterm ) {
    int count=0;

    try {
        if ( rs != null ) rs.close();
        if ( stmt != null ) stmt.close();
        Connection conn = DriverManager.getConnection(sConnStr);

```

```

        stmt = conn.createStatement();
        StringBuffer query=new StringBuffer(100);
        query.append("DELETE * FROM dropcourse ");
        query.append("WHERE dropcourse.cname=");
        query.append(cname);
        query.append(" ");
        query.append("AND ");
        query.append(" dropcourse.sid=");
        query.append(sid);
        query.append(" ");
        query.append("AND ");
        query.append(" dropcourse.csection=");
        query.append(csection);
        query.append(" ");
        query.append("AND ");
        query.append(" dropcourse.ctrm=");
        query.append(ctrm);
        query.append(" ");
        System.err.println("Query sentences is: " + query.toString() + "\n");
        stmt.executeUpdate(query.toString());
    }catch(SQLException ex) {
        System.err.println("deletedropcourse.executeQuery: " + ex.getMessage());
        return 1;
    }
    return 0;
}

```

```

public ResultSet executeQuery(String sql) {
    try {
        if ( rs != null ) rs.close();
        if ( stmt != null ) stmt.close();
        Connection conn = DriverManager.getConnection(sConnStr);
        stmt = conn.createStatement();

        rs = stmt.executeQuery(sql);
    }
    catch(SQLException ex) {
        System.err.println("dbacc.executeQuery: " + ex.getMessage());
    }
    System.err.println("Run Query:");
    if(rs==null)
        System.err.println("Run Query null:");
    else
        System.err.println("Run Query ok:");
    return rs;
}
}

```

Appendix C: The installation instructions

C-1 Installing JDK

In order to run this web site, you need to install JDK as Java environment.

C-1.1 Windows

For Windows environment, you can download the Java 2 SDK at Sun's Web site:

<http://java.sun.com/j2se/1.3/download-windows.html>. When the download is complete, double-click the file, the installer starts running. To complete the installation, you need to configure two environment variables: JAVA_HOME and PATH. You can do this through the control panel.

C-1.2 Linux

For Linux environment, you can point your browser to

<http://java.sun.com/j2se/1.3/download-linux.html>. Download and save the file in the /tmp/ directory. You also can find the installation document at the same Web site. Install the JDK according to the instruction of the installation document.

C-2 Installing Web server—Resin-2.1.6

C-2.1 Windows

Download the resin-2.1.6.zip at the Web site: <http://www.resin.com/download-widows>, unzip resin-2.1.6 and execute resin-2-1-6/bin/httpd, the Web server will be set up.

Browse <http://localhost:8080>, the resin default page should be shown.

C-2.2 Linux

Download the resin-2.1.6.tar.gz at the Web site: <http://www.resin.com/download-linux> and save it to /tmp/. As root, cd to /tmp and type tar -vzxf resin-2.1.6.tar.gz to install the resin-2.1.6 Web server. Execute resin-2.1.6/bin/httpd.sh to start the Web server. Browse <http://localhost:8080>, the resin default page should be shown.

C-3 Installing MySQL

C-3.1 Windows

Download MySQL from <http://www.mysql.com/download/mysql-3.23.html>.

Save the ZIP file. When the file is finished downloading, open it from the directory where it was saved and extract the zip file to a directory. You'll find the SETUP.EXE file in that directory. Run it to install MySQL.

C-3.2 Linux

Download MySQL from <http://www.mysql.com/download/mysql-3.23.html>.

If you download the RPM version, save them to /tmp, cd to /tmp and, as root, do an rpm -install on each of the files to install them. If you download the tar.gz version, following the install procedure provided by the installation document to install MySQL.

C-4 Installing JDBC for MySQL

Download the latest version of the MM.MySQL driver from

<http://sourceforge.net/projects/mmmysql>. Open a command prompt(Windows) or use the

command `cd /tmp(Linux)`. Run the following command:

```
jar xf mm.mysql-(version number)-you-must-unjar-me.jar
```

From the directory created by the above command, copy the file `mm.mysql-(version number)-bin.jar` to the `lib` directory of the Resin-2.1.6.

C-5 Installing the Web site files

Copy all `.jsp` files, `.html` files and images to the `Resion-2.1.6/doc`. Copy all `.java` files to the `Resion-2.1.6/doc/WEB-INF/classes`. Copy MySQL database files to the directory `mysql/data(Windows)` or `var/lib/mysl(Linux)`.