

NOTE TO USERS

This reproduction is the best copy available.

UMI[®]

**A ROLE BASED ACCESS CONTROL SYSTEM
FOR ELECTRONIC EDUCATION**

By

Hong Zhao

**SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN APPLIED SCIENCE
AT
SAINT MARY'S UNIVERSITY
HALIFAX, NOVA SCOTIA
OCTOBER 2004**

© Copyright by Hong Zhao, 2004



Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-98945-3

Our file Notre référence

ISBN: 0-612-98945-3

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

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

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

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

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.


Canada

SAINT MARY'S UNIVERSITY

Date: October 2004

Author: **Hong Zhao**

Title: **A ROLE BASED ACCESS CONTROL SYSTEM
FOR ELECTRONIC EDUCATION**

Department: **Mathematics and Computing Science**

Degree: **M.Sc.** Convocation: **May** Year: **2005**

Permission is herewith granted to Saint Mary's University to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.

Signature of Author

THE AUTHOR RESERVES OTHER PUBLICATION RIGHTS, AND NEITHER THE THESIS NOR EXTENSIVE EXTRACTS FROM IT MAY BE PRINTED OR OTHERWISE REPRODUCED WITHOUT THE AUTHOR'S WRITTEN PERMISSION.

THE AUTHOR ATTESTS THAT PERMISSION HAS BEEN OBTAINED FOR THE USE OF ANY COPYRIGHTED MATERIAL APPEARING IN THIS THESIS (OTHER THAN BRIEF EXCERPTS REQUIRING ONLY PROPER ACKNOWLEDGEMENT IN SCHOLARLY WRITING) AND THAT ALL SUCH USE IS CLEARLY ACKNOWLEDGED.

To my family

Table of Contents

Title Page	
Signatures of Examiners	ii
Table of Contents	v
List of Tables	ix
List of Figures	xi
Acknowledgements	xiii
Abstract	xiv
 Chapter 1. Introduction	 1
1.1 Motivation of the thesis study	1
1.2 Organization of the thesis	3
 Chapter 2. Access Control Techniques	 5
2.1. Overview.	5
2.2 Discretionary Access Control (DAC)	7
2.3 Mandatory Access Control (MAC)	10
2.4 Access Control List (ACL)	14
2.5 RBAC	17
2.5.1 Review of RBAC	17
2.5.2 RBAC Advantages	19
2.5.2.1 Time savings.	21
2.5.2.2 Cost savings.	22

2.5.2.3 Simplified Systems Administration	23
2.5.2.4 Reduced errors	23
2.5.2.5 Enhanced Systems Security and Integrity	23
2.6 Summary.	25
Chapter 3. Role Based Access Control Techniques	29
3.1 Overview of RBAC	29
3.1.1 History of RBAC	29
3.1.2 RBAC background	30
3.1.3 RBAC Characteristics and Policies	31
3.2 RBAC Concept	34
3.3 RBAC Components	36
3.3.1 Core RBAC	36
3.3.2 Hierarchical RBAC	39
3.3.3 Constrained RBAC	43
Chapter 4. Requirement and specification of RBAC based e-education system	46
4.1 E-education system requirements	47
4.2 E-education specification	50
4.2.1 User access function specification	50
4.2.2 Administrative function specification	52
4.2.3 E-education with RBAC	53
Chapter 5. Design of the RBAC based E-education System errors	57

5.1 RBAC based e-education system errors	57
5.1.1 Core RBAC based e-education system	58
5.1.1.1 System entities errors	58
5.1.1.1.1 USERS	58
5.1.1.1.2 ROLES	59
5.1.1.1.3 Objects	61
5.1.1.1.4 Operations	61
5.1.1.1.5 Permissions (needs examples for these simple paragraphs)	61
5.1.1.1.6 Sessions	61
5.1.1.2 User/role assignment	62
5.1.1.3 Permission/role assignment	62
5.1.2 Hierarchical RBAC based e-education system	64
5.1.3 Constrained RBAC based e-education system	66
5.2 E-education system interface design	67
5.2.1 Administrative support interfaces	69
5.2.2 Academic support interfaces	70
5.3 Systems functionality	71
5.3.1 Administrative functions	71
5.3.2 Supporting system functions	76
5.3.3 Review functions	78
5.4 Security and privacy	79
5.4.1 Security	79
5.4.2 Privacy	80

Chapter 6. Implementation	81
6.1 Implementation Environment Overview	81
6.2 Administrative Interfaces	86
6.3 User Interfaces	113
6.3.1 Logon Interfaces	113
6.3.2 Registration Interface	117
6.3.3 Time-Controlled Interfaces	118
6.4 Database implementation	122
6.5 System test and analysis	127
Chapter 7 Summary and future works	129
7.1 Summary	129
7.2 Future Work	132
REFERENCES	134
APPENDIX	138

List of tables

Table 2.1 Simple comparison of MAC, DAC and ACLs	17
Table 5.1 Roles and their permissions in Core RBAC model for e-education . .	63
Table 5.2 Roles and their permissions in Hierarchy RBAC model for the e-education system	65
Table 5.3 Z notation symbol explanation	73
Table 5.4 Explanation of an example function described in Z notation	74
Table 5.5 Definition and specification of administrative functions	74
Table 5.6 Definition of user-role and permission-role assignment functions . . .	75
Table 5.7 Definition and specification of session related functions	77
Table 5.8 Definition and specification of system review functions	78
Table 5.9 Three levels of administrators and their duties	79
Table 6.1 Registration information for the new users	89
Table 6.2 Users' table containing all the users' username, password, and SIN	89
Table 6.3 Data for all registered users	89
Table 6.4 The <i>roles</i> table containing all roles of the system	93
Table 6.5 The <i>sessions</i> table	93
Table 6.6 The <i>conflictRoles</i> table	94
Table 6.7 The <i>hierarchyRoles</i> table	94
Table 6.8 The <i>permissionRoles</i> table	94
Table 6.9 The <i>userRole</i> table	99
Table 6.10 The <i>academicRecords</i> table	99

Table 6.11 The <i>permissions</i> table	101
Table 6.12 The <i>objects</i> table	101
Table 6.13 The <i>operations</i> table	102

List of Figures

Figure 2.1 User and System Resources relationship in DAC	8
Figure 2.2 User and System Resources relationship in MAC	11
Figure 3.1 Core RBAC	36
Figure 3.2 Users are assigned to a role	38
Figure 3.3 Permissions are assigned to a role	38
Figure 3.4 Use - Role - Permission relationships	38
Figure 3.5 Hierarchical RBAC	40
Figure 3.6 a. Role 1 inherit role2 and role 3	40
Figure 3.6 b. Role 1 is an immediate descendant of role 2	40
Figure 3.7 General Role Hierarchy	41
Figure 3.8 Limited Role Hierarchy	42
Figure 3.9 Lattice Role Hierarchy	43
Figure 3.10 Static Separation of Duty relations (SSD)	44
Figure 3.11 Dynamic Separation of Duty relations (DSD)	44
Figure 4.1 Various users and roles, and their relations	54
Figure 4.2 Roles sharing common permissions	55
Figure 4.3 Complex RBAC model of this study	56
Figure 5.1 Role Hierarchical relationships in e-education system	64
Figure 5.2 User logon confirmation processes	68
Figure 6.1 Three tier architecture of e-education system	82
Figure 6.2 Middle tier - Orion server	83

Figure 6.3 Interface for Level I administrators	86
Figure 6.4 Interface for Level II administrators	87
Figure 6.5 Interface for Level III administrators	87
Figure 6.6 Processes of confirming new users' registrations	89
Figure 6.7 Role management interface	92
Figure 6.8 Processes of Role Management	93
Figure 6.9 Interface for User Management duty	98
Figure 6.10 Processes of User Management	98
Figure 6.11 Permission Management Interface	100
Figure 6.12 Processes of Permission Management	101
Figure 6.13 Interface for the role/user/permission assignment management .	105
Figure 6.14 Processes of Role/User/Permission Management	106
Figure 6.15 Showing the roles and sessions that the user e2651855 has . . .	107
Figure 6.16 Logon interface for regular users	114
Figure 6.17 Login process	115
Figure 6.18 Logon interface for top level administrators	116
Figure 6.19 Chief Administrators login process	116
Figure 6.20 Registration interface for new users	117
Figure 6.21 New users' registration process	118
Figure 6.22 Assignment interface with time control	119
Figure 6.23 A quiz interface with time control	121
Figure 6.24 Assignment/quiz/exam submission involved processes	122
Figure 6.25 Database tables and their relationships	126

Acknowledgement

First of all, I would like to thank my supervisor, Dr. Shyamala C. Sivakumar from the Finance & Management Science Department of Saint Mary's University, for her great support, and thoughtful/creative/stimulating advices through the entire journey of this thesis study. Thanks to her encouragement and trust in me. It has been a privilege to work with her. I cannot express enough appreciations to her.

I would also like to express my gratitude to all those who gave me the possibility to complete this thesis. I want to thank my committee members, Dr. Pawan Lingras, Dr. Stavros Konstantinidis, and Dr. William Phillips, for taking time to examine the study and for their comments and suggestions.

Also I want to thank the DEPARTMENT OF MATHEMATICS AND COMPUTING SCIENCE of Saint Mary's University for their constant support on this thesis study. In particular, Mr. Owen M. Smith, our department system manager, whose help in the early stage of this study in building the client-server environment and his trust in accessing the system server, I greatly appreciated these helps.

Thanks also go to Mr. Perry Sisk and Ms. Hanaa Aboushahla from Saint Mary's University for providing information on Saint Mary's University's information management system.

Abstract

A ROLE BASED ACCESS CONTROL SYSTEM FOR ELECTRONIC EDUCATION

By Hong Zhao

Role Based Access Control (RBAC) is an advanced and promising access control technology. RBAC associates users with roles, roles with permissions, and a user accesses a permission only when the user has an authorized role which is associated with that permission.

The thesis is a study of the NIST RBAC standard, and a demonstration of RBAC's application in e-education system. It gives a requirement analysis and specification for e-education system, and designs a hybrid RBAC model. This study also introduces two unique features: one is to create a privacy attribute and associate it with each permission; the other one is to have three levels of administrative functions. It is proved that these novel techniques not only reduce the number of conflict roles but also improve security and enforce privacy.

December 1, 2004

Chapter 1

Introduction

1.1 Motivation of the thesis study

Today, organizations' operation and management have started to heavily rely on computers and Internet. Their success largely depends on how reliable and secure their information resource system is. Effective and efficient access control to their information system is needed. At present, an organization's information resource is usually controlled through the Access Control Lists (ACLs), which specify, for each protected resource, a list of named individuals/users with their respective modes of access to these objects. The access lists are updated by system administrators as users' needs and permissions change [Gavrila and Barkley, 1998; Ferraiolo, et. al., 1999].

Though widely used, ACLs complicate matters by allowing the direct association of users with permissions. A large number of users each with many permissions implies a very large number of user/permission associations management. As such, when a user takes on different jobs within the organization, the system administrators may have to make many selective addition or deletion of user/permission associations on all systems. This can be seen as a waste of

time, a source for introducing errors, and thus a security risk [Saudhu Ravi, 2002]. This issue tends to get worse in organizations which have a large number of users.

Because of the potential problems associated with this lack of operational security assurance, organizations have resisted in making sensitive information and resources available online; this not only limits organizations' resource utilization, but also deprives the organizations of potential productivity gains [Gavrila and Barkley, 1998; Gallaher, 2002].

That is why Role Based Access Control (RBAC) has received increasing attention in recent years [Ferraiolo, et. al., 1999, 2001]. Based on the "Proposed NIST Standard for Role Based Access Control" [David F. Ferraiolo, et. al., 2001], users are not directly assigned to permissions for each application. Instead, users are assigned to roles and the roles are mapped to permissions, permissions are associated with roles and users are made members of appropriate roles. Roles are created based on the requirements of different job functions in an organization. In this way, RBAC allows administrators to use the natural structure of an organization to manage access control; it tends to reduce the cost of information system management while improving the enforcement of system security policies. The other advantages of RBAC over Access Control List are that it requires less storage and less maintenance, and less administration [Gallaher, 2002].

Under RBAC, roles can have overlapping responsibilities and privileges; that is, users belonging to different roles may need to perform common operations.

Some general operations may be performed by all employees. In this situation, it would be inefficient and administratively cumbersome to specify repeatedly these general operations for each role that gets created. Role hierarchies can be established to provide for the natural structure of an enterprise. A role hierarchy defines roles that have unique attributes and may contain other roles; that is, one role may implicitly include the operations that are associated with another role.

With the noticeable advantages, RBAC can be used in healthcare, bank, government, and university. Coupled with the continuous advances in Internet technology, RBAC has demonstrated its application potentials. In recent years, user communities have expressed great interests with RBAC, but its application potential has not been explored in e-education. This study is, therefore, carried out to explore the RBAC's application potential in e-education.

1.2 Organization of the thesis

Based on the "Proposed NIST Standard for Role Based Access Control" [Ferraiolo, et. al., 2001], this study introduces the standard RBAC component models: Core RBAC, Hierarchical RBAC, and Constrained RBAC which contains Static Separation of Duty Relations (SSD) and Dynamic Separation of Duty Relations (DSD) [Ferraiolo, et. al., 2001]. It gives a discussion of e-education

system requirement, a hybrid RBAC model for e-education system, and the design and implementations of the RBAC based e-education system.

The thesis is organized as follows. Chapter 2 overviews the traditionally and currently widely used access control techniques, including Discretionary Access Control, Mandatory Access Control, and Access Control List. It also gives a brief introduction of role Based Access Control and its advantages. Chapter 3 describes in detail the NIST standard concept and models of RBAC. Chapter 4 explores the potential of RBAC in e-education application, and gives an analysis and specification of e-education requirement. Chapter 5 demonstrates the detailed design of the e-education system based on the requirement/specification in Chapter 4 and on RBAC concept and models. The detailed implementation is given in Chapter 6. Chapter 7 gives a conclusion and discussion future directions for research.

Chapter 2

Access Control Technology

2.1 Overview

Access control is a set of procedures performed by hardware, software and administrators to monitor access, such as identify users' requesting access, record access attempts, and grant or deny access based on pre-established rules. It is the heart of security [Posulns, Shimonski and Faircloth, 2003]. It is used to protect an organization's data from unauthorized viewing, modification or copying, and protect a system from unauthorized use, modification or denial of service.

Access control technology evolved from the research and development in the 1960's and 1970's supported by the Department of Defense (US). That research and development resulted in two fundamental types of access control techniques: Discretionary Access Control (DAC) and Mandatory Access Control (MAC) [Ferraiolo and Kuhn, 1992]. DAC has been perceived as being technically correct for commercial and civilian government security needs, and has been applied to single-level military systems in which all users are in the same security level, but may have different access rights [Ferraiolo and Kuhn, 1992]. MAC is

mainly used for multi-level secure military systems and national security arenas in which different users are in different security levels based on their rank; its use in other applications is rare. In general, the two access control technologies were applied to prevent unauthorized access to classified information.

Since then, advanced computer technology has been deeply and widely involved in commercial and civilian government organizations. Civilian government and corporations started to rely heavily on information processing systems to meet their operational, financial, and information technology requirements. Online transactions and communications have become regular business operations in all industries, including education, government, commerce, manufacturing, finance, and health. The emergence of e-government (e.g. tax department) and e-business (e.g. online shopping) is an example. At the same time, it has been understood that once organizations' computer systems are connected by Internet and Intranet, the organizations' resources, plans, and data are at risk. Any security failure to the computer systems, including corruption, unauthorized disclosure, or theft of corporate resources, could disrupt the organization's operations and have immediate and serious impacts on finance, legal, human safety, personal privacy, and public confidence. Therefore, significant and broad sweeping security requirements exist outside the Defence systems. Access control policies are required not only by computer systems in military but also in all other industries and services that are computer systems facilitated. More sophisticated and effective access control technology therefore is needed.

The most popular model that has emerged as a full fledged model is RBAC, and it is as mature as traditional MAC and DAC concepts. It is becoming the leading access control model for information security management and is incorporated in major computer operating systems design [Ferraiolo et. al. 2001].

In addition to the above access control technologies, Access Control Lists (ACLs) technology has also been widely used in the major computer operating systems, such as Unix and Windows. This control technology was developed to facilitate advanced network communications. ACLs technology has been used to restrict multi-users' access to Network and Internet information system resources.

In the following subsections we give a brief description of the following access control technologies: I) Discretionary Access Control (DAC), II) Mandatory Access Control (MAC), III) Access Control List (ACL), IV) and Role-based Access Control (RBAC).

2.2 Discretionary Access Control (DAC)

DAC is generally used to restrict users' access to system objects and programs. It limits access to the system resources (objects and programs) based on the identity of subjects and/or groups to which the users belong. DAC allows users to grant and revoke access privileges to any of the system resources. As such, users are the owners of the objects and programs under their control.

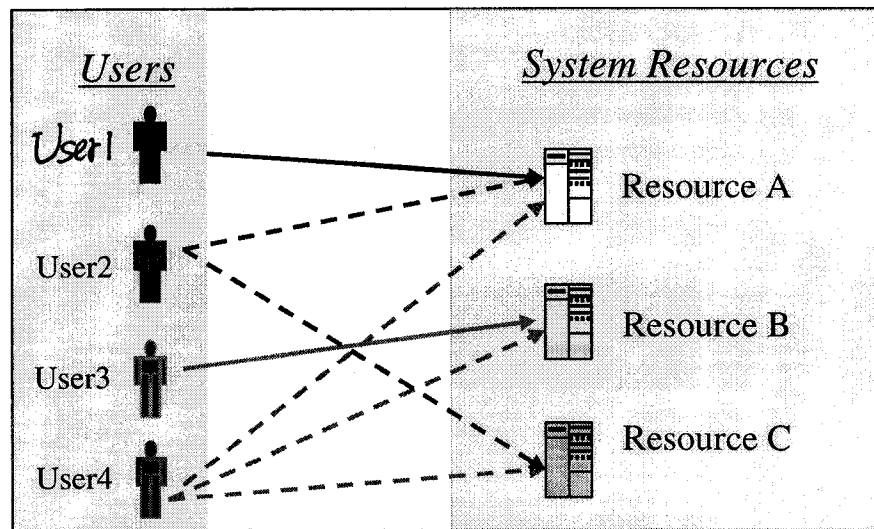


Figure 2.1 User and System Resources relationship in DAC

Figure 2.1 shows that the access rights for each user in DAC are different. User1 and User3 each can access one system resource, resource A and resource B, respectively. The user2 has the right to access both resource A and Resource C. The user4 can access all the system resources, Resource A, Resource B, and Resource C.

However, for many organizations, the end users do not own the information for which they are allowed access. It is the corporation or agency that is the actual owner of system objects and programs. Access privileges are controlled by the organization and are often based on employee functions rather than ownership.

For strong system security, the DAC mechanism is fundamentally inadequate. That is because DAC access decisions are only based on user's identity and ownership, and it ignores other security-relevant requirement such as the roles of

the user, the function and trustworthiness of the program, and the sensitivity of the objects (data and information). Each user has complete discretion over his objects; this makes it impossible to enforce a system-wide security policy.

Furthermore, every program run by a user is allowed to inherit all of the permissions granted to the user and is free to change access to the user's objects, so no protection is provided against malicious software [Loscocco and Smalley, 2001].

In addition, because the controls are discretionary, a user or process given discretionary access to a resource is capable of passing that information (resource) along to another subject [Jordan, 1987]. This may cause data integrity issues.

DAC has been used by Unix, NT, NetWare, Linux, Vines, etc. An example of using DAC is the standard Solaris system [Sun Trusted Solaris 7 Operating Environment, 2004]. Solaris system uses file permissions and optional access control lists to restrict access to information based on a user's identity or group membership. DAC is discretionary and allows a file's owner to change its permissions. But, in fact the permissions on system files can only be changed by the administrator who owns them. Therefore, in the Solaris environment DAC has to be used along with Mandatory Access Control to control access to system files [Sun Trusted Solaris 7 Operating Environment, 2004].

Before network technology was introduced, an organization's data and information were normally stored in a single computer system. Using DAC to control users' access to system resources made sense then.

2.3 Mandatory Access Control (MAC)

The need for a MAC mechanism arises when the security policy of a system dictates that protection decisions must not be decided by the object owners and the system must enforce the protection decisions, that is, the system enforces the security policy over the wishes or intentions of the object owner [Ferraiolo and Kuhn, 1992].

MAC enforces the corporate policy or security rules by comparing the sensitivity of the resource (e.g., file or storage device) with the clearance of the subjects (e.g., user or application). MAC mechanisms assign a security level to all resources, assign a security clearance to each user, and ensure that users only have access to those resources for which they have a clearance. An example could be: classified data may only be accessed by staff with a 'Secret' clearance level.

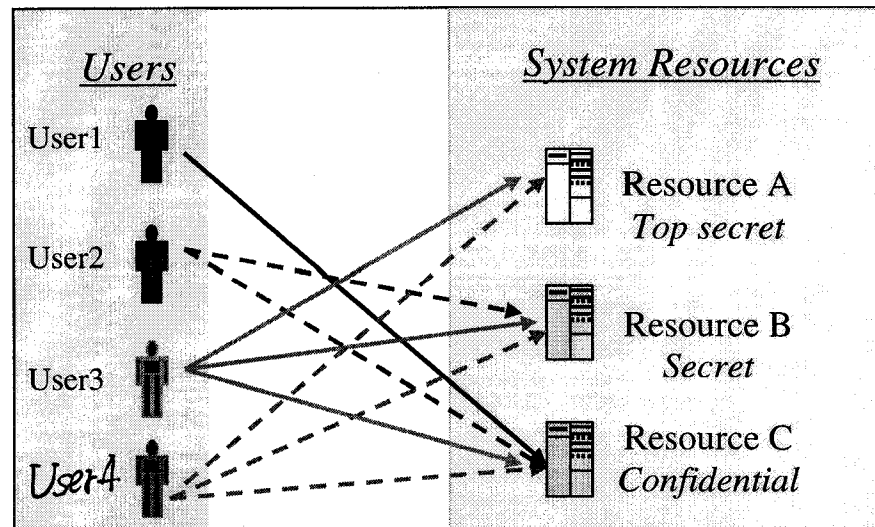


Figure 2.2 User and System Resources relationship in MAC

Figure 2.2 shows each resource has been labeled by security levels, either top secret, or secret, or classified. User1 has Confidential security level, so can only access resource C. User2 has Secret security level, so can access resource B and C. User3 and user4 both have Top Secret security level, so they can access all the resources.

MAC is usually appropriate for extremely secure systems including multilevel secure military applications or mission critical data applications. An MAC access control model often exhibits one or more of the following attributes.

- Only administrators can make changes to a resource's security label, and data owners cannot.
- All data is assigned security level that reflects its relative sensitivity, confidentiality, and protection value.

- All users can read from and post to a lower classification than the one they are granted (e.g., a "secret" user can read an unclassified document).
- All users can post/submit to a higher classification (e.g., a "secret" user can post information to a Top Secret resource).
- All users are given read/write access to objects only of the same classification (e.g., a "secret" user can only read/write to a secret document).
- Access is authorized or restricted to objects based on the time of day depending on the labeling on the resource and the user's credentials (driven by policy).
- In data communication Networks, access is authorized or restricted to objects based on the security characteristics of the HTTP client (e.g., SSL bit length, version information, originating IP address or domain, etc.)

Because MAC secures resources by assigning sensitivity labels on resources and comparing this to the level of sensitivity a user is operating at, in general, MAC mechanisms are more secure than DAC, but MAC has trade offs in performance and convenience to users [Curphey, et. al. 2002].

Adding the MAC mechanism in an information system, the vulnerabilities of DAC described in the previous section can be addressed. MAC access decisions are based on labels that can contain a variety of security-relevant information. MAC policy is defined by a system security policy administrator and enforced over all subjects (processes) and objects (e.g. files, sockets, network interfaces) in the

system. MAC can support a wide variety of categories of users on a system, and it can confine the damage that can be caused by flawed or malicious software.

MAC, as defined in the DoD's Trusted Computer Security Evaluation Criteria (TCSEC), is "A means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e. clearance) of subjects to access information of such sensitivity." These policies for access control are not particularly well suited to the requirements of government and industry organizations that process unclassified but sensitive information. In these environments, security objectives often support higher level organizational policies which are derived from existing laws, ethics, regulations, or generally accepted practices. Such environments usually require the ability to control actions of individuals beyond just an individual's ability to access information according to how that information is labeled based on its sensitivity.

In addition, the traditional MAC mechanism is typically closely coupled to a multi-level security policy which bases its access decisions on classifications for objects and clearances for subjects. It provides poor support for data and application integrity, separation of duty, and least privilege requirements. It requires special trusted subjects that act outside of the access control model. It fails to tightly control the relationship between a subject and the code it executes. This limits the ability of the system to provide protection based on the function and trustworthiness of the code, to correctly manage permissions required for execution, and to minimize the likelihood of malicious code execution.

MAC is mandatory because the labelling of information happens automatically, and ordinary users cannot change labels unless they are authorized by an administrator [Sun Trusted Solaris 7 Operating Environment, 2004].

In MAC it is the system that decides how the resources will be shared. Only the administrators, not object owners, may change the object level. It is used in systems where security is critical, i.e., military. MAC is also hard to program, configure, and implement. It relies on the system to control access. For example, if a file is classified as secret, MAC will prevent anyone from writing top secret information into that file.

2.4 Access Control List (ACL)

ACL is typically a file used by the access control system to determine who may access which programs and files, in what method and at what time. Different operating systems may have different ACL terms. The typical types of access include *read*, *write*, *create*, *execute*, *modify*, *delete*, and *rename*.

ACL specifies, for each protected resource, a list of named individuals/users with their respective modes of access to these objects. It is often considered as an object that is associated with a file and contains entries specifying the access that individual users or groups of users have to the file. ACL provides a

straightforward way of granting or denying access for a specified user or groups of users.

ACL is mostly used in computer operating systems and computer networks. In a computer operating system, ACL is designed as a table that tells the system which access rights each user has to a particular system object (such as an individual file or file directory). Each object has a security attribute that identifies its access control list. The list has an entry for each system user with access privileges. The most common privileges include the ability to read a file, to write to a file, and to execute a file (if it is an executable file or program).

Microsoft Windows NT/2000, Novell's Netware, Digital's OpenVMS, and Unix-based systems are among the operating systems that use access control lists. The list is implemented differently by each operating system. For instance, in Windows NT/2000, each ACL is associated with a system object. It has one or more access control entries that consist of the name of a user or group of users. For each of these users or groups, the access privileges are stated in a string of bits called an access mask. Generally, the system administrator or the object owner creates the access control list for an object.

In computer network, ACLs are rules that control access between networks. They provide security for the entire network. Access lists monitor network traffic by controlling whether packets are forwarded or blocked at the switch's interfaces [Technology Highlights, 2004]. Based on criteria specified within the ACL (source

address, destination address, upper-layer protocol, or other information), the switch examines each packet to determine whether to forward or drop it. With ACL, access to specific segments of the network can be controlled (e.g., Host A is allowed to access the Human Resources network while Host B is denied access to the same segment). Access lists can also be used to decide which types of traffic are forwarded or blocked at the switch interfaces. For example, you can permit e-mail traffic to be routed, while at the same time all Telnet traffic is blocked.

In general, these methods are effective in a static environment, but not in dynamic environments, where users enter and leave, or change positions within the organization often. The constant stream of changes and frequent updating of access permissions have to be made; these are often time-consuming, expensive and error-prone processes. A common security lapse with these approaches is administrators not making timely permission updates, enabling unauthorized users to access restricted data [RBAC Case study, 2004].

In summary, comparing the above three access control technologies (Table 2.1), it is clear that MAC enforces great security, but too much to configure and implement, and may not be applicable in organizations outside military. DAC is not secure enough. ACLs provide a good access mechanism, but for a large number of users, ACLs can be difficult to manage.

Table 2.1 Simple comparison of MAC, DAC and ACLs

	How to enforce access
DAC	Based on users' identity
MAC	Based on users' identity and objects' classification
ACLs	Attach a list of users to each object of the system

To address these security management issues, Role Based Access Control (RBAC) technology was developed. As described in the next section and Chapter 3, RBAC is a better solution for information security management.

2.5 RBAC

2.5.1 Review of RBAC

RBAC is an advanced access control mechanism. While traditional approaches (ACL, MAC, DAC) control access to system data and network resources by directly specifying detailed permissions for each user, RBAC uses roles or job responsibilities to organize access privileges.

In RBAC, rights and permissions are assigned to roles rather than to individual users. Users acquire these rights and permissions as they are assigned membership in appropriate roles. This simple idea greatly eases the administration of authorizations and improves information system security.

On the surface, basing access on roles or job descriptions may seem a bit restricting, but RBAC allows for granting groups multiple access permissions and even the ability to allow specific individuals elevated access privileges. For example, the accounting department staff would have access to financial systems and data, but their managers could also be granted access to human resource files and marketing projections. Roles can also be set up based on locations, projects and management level [RBAC Case study, 2004].

RBAC has the potential to be used in almost any organization that uses a computer network to limit access to particular pieces of information. Industries that will especially benefit from RBAC are those for which information security is a key, such as banking, health care, government, software development, and the military [RBAC Case study, 2004].

The efficiency and cost savings come from the diminished administrative need in maintaining a RBAC system. Employee turnover and assignment changes make it difficult to keep up with the constantly changing human resources landscape. That's not the case with roles, whose description usually doesn't change too often. By only having to add and remove users from role groups, the organization can cut down on the administrative costs and reduce the potential for error [RBAC Case study, 2004].

RBAC provides greater productivity on the part of security administrators, resulting in fewer errors and a greater degree of operational security. It can also be argued that RBAC actually simplifies information system management, administration and security. This is achieved by statically and dynamically controlling the actions of users by establishing and defining roles, role hierarchies, relationships and constraints [Ferraiolo, Cugini and Kuhn, 1995]. In an RBAC environment, the main administrative tasks tend to be adding and deleting users to and from roles. This differs from the more conventional and less intuitive process of directly managing lower level access control mechanisms, such as access control lists, on an object-by-object basis.

2.5.2 RBAC Advantages

In detail, RBAC has two unique advantages. First, RBAC is based on the user's role rather than the user's identity, direct and indirect administrative costs are greatly reduced. By assigning individuals to predefined roles, the administrative process of establishing privileges is streamlined and management time for reviewing privilege assignments is reduced. For example, if a user moves to a new function within the organization, the user can simply be assigned to the new role and removed from the old one, whereas in the absence of an RBAC model, the user's old permissions would have to be individually revoked, and new permissions would have to be granted. Secondly, RBAC provides greater security because it prevents users from obtaining inconsistent or incompatible privileges that can enable access violations [RBAC Case Study, 2004].

The major benefits of RBAC are the ability to express and enforce enterprise-specific security policies and to simplify the process of security management. RBAC is a framework of policy rich mechanisms that allow per-subject (role) as well as per-object access review. Its configuration is dependent on organizational policies. This allows RBAC to be adaptable (more so than other types of access control) to any organizational structure and means of conducting business. The policies implemented under RBAC can evolve over time as enterprise and organizational structure and security needs change. RBAC has been seen as a “commercial” and cost-effective alternative to the MAC concepts [A Review Paper of Role Based Access Control, 1999].

Additionally, in a distributed environment, administrator responsibilities can be divided among central and local protection domains [Ferraiolo, Cugini and Kuhn, 1995]. In other words, central protection policies can be defined at the enterprise level while leaving protection issues that are of local concern at the organizational unit level.

RBAC also has many operational benefits. One of the most significant is cost savings to an organization. Definite figures of RBAC cost savings are unavailable, but a survey of *Information Security* readers conducted by the Research Triangle Institute (RTI) on behalf of the National Institute of Standards and Technology (NIST) provides a glimpse of financial benefit this new technology can produce [Gallaher, O'Connor and Kropp, 2002].

Based on an ongoing RTI study, firms that implement a RBAC system yield two major benefits: reduced administrative overhead and improved employee productivity. With access based on roles, administrators aren't hampered by the laborious task of updating individual user privileges. Consequently, employees can gain faster access to systems critical to their jobs. To investigate the potential magnitude of these benefits, RTI compiled the responses of more than 100 *Information Security* readers on the administrative costs associated with various access control systems.

A recent report by SETA Corp., sponsored by NIST, asserted that organizations with certain staffing, data and organizational characteristics could reap tremendous benefits from deploying a RBAC system. These characteristics reflect the common problems found in many IT-dependant organizations: tremendous demand for services clashing with limited resources. According to SETA, organizations with a large number of staffs with high turnover rates, limited security resources, stable organizational structure and application, and maximum control over IT resources and data would benefit the most from a role based access system [A review paper of Role Based Access Control, 1999].

The quantifying examples that show the advantages of using RBAC, and other security improvements are listed in the following few subsections.

2.5.2.1 Time savings

The study [RBAC Case study, 2004] by RTI indicates that organizations using RBAC made significant timesaving over conventional user based access control mechanisms in assigning privileges to new users, and slightly better ability to update user privileges. Examining the number of times these tasks are performed on a daily basis and the number of employees in an organization, it is concluded that a RBAC system could save an organization 7.01 minutes per employee, per year in administration functions [RBAC Case study, 2004]. When take consideration of the average hourly salary of an IT administrator-\$59.27 per hour on average, this saving takes on significance. The annual cost savings ranges from \$6,924 a year for organizations with 1,000 employees to \$692,471 for organizations with 100,000 employees.

2.5.2.2 Cost savings

The survey [RBAC Case study, 2004] also shows a great cost-saving which is further amplified when combined with the reduced employee downtime. When new and transitioning employees receive their system privileges faster through a RBAC system, their productivity is increased. The average downtime for new employees while waiting for system access privileges was 26.4 hours with non-RBAC systems and 14.7 hours with RBAC systems. This means that using RBAC can result in an overall saving in average downtime of 11.7 hours per new employee. If the average employee hourly wage is \$39.27, the annual employee turnover rate is 13 percent, and the annual growth rate is 3 percent, the annual

productivity cost savings yielded by a RBAC system ranges from \$75,000 for an organization of 1,000 employees to \$7.4 million for an organization of 100,000 employees.

2.5.2.3 Simplified Systems Administration

In theory, if a role is attached with X number of users, and the role has Y number of permissions, it is obvious that the administration cost with RBAC is $(X + Y)$. This is less than the cost of traditional administration $(X * Y)$. When the number of permissions and users increases, this cost becomes very impressive.

2.5.2.4 Reduced errors

Because less maintenance is required, the chance to make mistakes is limited. RBAC minimizes each role's privileges, and gives each role minimum necessary permissions.

2.5.2.5 Enhanced Systems Security and Integrity

While security is a primary concern of many organizations, management more often sees it as a drain on financial resources than as a benefit to the bottom line. This perception persists until the inevitable happens -- the organization suffers from an insider security lapse. Various surveys, including those conducted by *Information Security*, have found that a significant number of organizations have

experienced an insider security lapse, costing an average of about \$250,000 (US) dollars per incident.

RBAC can reduce the impact from security violations in two ways. 1), RBAC decreases the likelihood that a security violation occurs; 2), if a security violation occurs, RBAC can limit the damage from the violation. Roles limit the possibility of internal security breaches from individuals who should not have access to the data and applications associated with each function. In addition, because privileges are not assigned manually, it is less likely that the administrator will make an error and inadvertently grant a user access to information or applications to which he or she would otherwise be prohibited. The greater control over users' access to information and resources will result in minimized potential for inside security violations.

The enhanced security provided by RBAC systems would result in further cost savings. Some respondents, without providing specifics, did indicate that RBAC systems reduced the number of security violations in their organizations.

In summary, the administrative savings, downtime savings and security benefits derived from RBAC systems will vary based on the size of an organization and its industry. The greater employee turnover, and in turn the number of people changing roles, the greater the cost savings of RBAC relative to other access control systems. Also, some organizations are very dynamic, and user roles and

permissions change quickly. In these environments, RBAC is more efficient in moving users in and out of given roles and changing the permissions of given roles than competing access control systems. This improved efficiency is observable in the decrease in labor hours that the computer network support team spends on administrative tasks.

Based on the survey [RBAC Case study, 2004], organizations of all sizes can yield a satisfying return on investment - both in direct and indirect costs -- by deploying an RBAC system. It is also observed that medium- to large-sized organizations would yield a quicker return of investment from deploying a RBAC system, since the savings increase with the number of users on a system.

2.6 Summary

ACLs is one of the most common access control models [Gallaher, O'Connor and Kropp, 2002]. When using ACLs, every piece of data or application has a list of users associated with it who are allowed access. In such a system, the security administrator will easily see which users have access to which data and applications. Changing access to the piece of information is straightforward: the administrator simply adds or removes a user from the ACL. Each set of data or application has its own ACL, but there may or may not be a corresponding list that gives the administrator information on all of the pieces of information to which a particular user has access. Only by examining each piece of data

individually and checking for access can the administrator find any potential security violations. If all accesses by a particular user need to be revoked, the administrator must examine each ACL, one by one, and remove the user from each list. When a user takes on different responsibilities within the organization, the problem gets worse. Rather than simply eliminating the user from every ACL, the administrator must determine which permissions need to be eliminated, left in place, or altered.

In DAC it is the individual who owns the data that controls the access to the data. ACLs is regarded as one implementation of DAC [Gallaher, O'Connor, and Kropp, 2002]. DAC governs access to information based on the user's identity and rules that specify which users have access to which pieces of information. Whereas ACLs are lists that specify which users can access a particular piece of data, DAC consists of a set of rules that specify which users are allowed to access the data [Gallaher, O'Connor, and Kropp, 2002].

When a user requests access to a particular piece of data, the system searches for a rule that specifies which users are allowed to access the information. If the rule is found, the user is given access; if not, the user is denied [Gallaher, O'Connor and Kropp, 2002]. For example, a rule may state that users from a certain group are not allowed to read a particular data file. Rule-based DAC is an improvement over ACLs, but it is still susceptible to human error and therefore suffers from potential security violations. DAC does not impose any restrictions on data access for a particular user. Once users can access data, they can

change or pass that information onto any other user without the administrator's knowledge [Sandhu and Samarati, 1994].

MAC is a departure from other access control mechanisms because it is based on hierarchical security labels and assigns each user and each piece of information or application a particular security level (e.g., confidential, secret, top secret). Two common principles are then applied to determine if a user has access to a particular piece of information: read down access and write up access [Gallaher, O'Connor and Kropp, 2002]. Read down access gives users the ability to access any piece of information that is at or below their own security level. If a user has a secret security level, they are able to access secret and confidential material but not top secret material. Write up access states that a subject's clearance must be dominated by the security level of the data or information generated. For example, someone with a secret clearance can only write things that are secret or top secret. With these two access control principles, information can only flow across security levels or up security levels.

RBAC is now considered a comprehensive mechanism (details are given in Chapter 3). It encompasses MAC and DAC as special cases and goes beyond them in providing a policy-neutral framework [Gallaher, O'Connor and Kropp, 2002]. And, in recent years, user communities have expressed great interests with RBAC but disenchantment with traditional MAC and DAC. This is because: 1) DAC is not applicable to the majority of personal information systems; 2) the most commonly used MAC is the multilevel security mechanism used by the US

Department of Defence, which associates information with such labels as *TOP SECRET, SECRET, and CONFIDENTIAL*. This type of MAC is not flexible enough for commercial use, nor is it adequate for the needs of personal information systems [A Review Paper of Role Based Access Control, 1999].

Chapter 3

Role Based Access Control Technology

3.1 Overview of RBAC

3.1.1 History of RBAC

RBAC was first introduced by Ferraiolo and Kuhn in 1992. Since then, in order to identify the true values of RBAC features in enterprises and the potential of its practical implementation, university researchers, vendors, and the National Institute of Standards and Technology have made great efforts. As a result of these efforts, a number of RBAC models and applications were proposed, and the potential benefits of RBAC technology have been recognized [Ferraiolo and Kuhn, 1992].

While these models and implementations are relatively similar on fundamental RBAC concepts, they differ in significant details. The models and applications come from different commercial and academic backgrounds, little consensus exists on what to call the different parts. Many models use different terminologies to describe the same concepts. There were no attempts at standardizing main RBAC features [Ferraiolo and Kuhn, 1992].

In order to address these issues of terminology and scope, and define a consensus standard, the first effort was made at the 2000 ACM Workshop on RBAC. Based on the follow-up comments and panel discussions at that workshop, a formal proposal titled *Proposed NIST Standard for Role-Based Access Control* was published in 2001 [Ferraiolo, et. al., 2001].

Through the past few years of research and practical implementations, RBAC's advantages and market values have been proven. On February 19, 2004, RBAC became an American National Standard - ANSI INCITS 359-2004 [National Institute of Standards and Technology].

3.1.2 RBAC background

Access control techniques are used to control the actions, functions, applications and operations of legitimate users and to protect the integrity of the stored information within an organization's computer system. The effectiveness of an access control technology is measured on two criteria: reliability of security and ease of administration [Andress, 2001]. RBAC can meet the two criteria, because RBAC has two main advantages as discussed in Chapter 2.

Because of the two advantages, and the unique feature that it can map to organizational structure, RBAC has been incorporated into major information technology vendors' product lines, ranging from defence to health care [Ferraiolo,

et. al., 2001; Role Based Access Control, 2003; A review paper of Role Based Access Control, 1999], and RBAC has now become the leading model for advanced access control, in particular, in large organizations where the natural role hierarchies and separation of duty are well known [Jaeger, Michailidis and Rada, 1999].

Though widely applied in many industries and services, RBAC has not been practiced in education environments. Some e-educations, such as Moodle, have started to use some of the RBAC features, but different from RBAC. That is why this study was undertaken; it is to explore the potential of RBAC in e-education system applications.

RBAC is an open architecture: its officially approved standards and specifications are available to the public for free, and anyone can design add-on features for it; it gives organizations the flexibility to meet their own specific needs.

3.1.3 RBAC Characteristics and Policies

RBAC is often described in terms of users, roles, role hierarchies, operations and objects. To perform an operation on an RBAC controlled object, a user must be active in some role. This assumes that the user is an authorized member of that role. RBAC enables administrators to place constraints on role authorization, role activation and operation execution. Constraints could include mutual exclusivity rules that can be applied on a role-by-role basis. Constraints can also be placed

on the authorization of an operation to a role and on operations being performed on objects, for example, time and location constraints.

The following list outlines some of the characteristics of RBAC [Ferraiolo, Cugini and Kuhn, 1995]:

- under the RBAC framework, a user is an individual, a role is a set of job functions, and an operation represents a particular mode of access to a set of one or more RBAC objects;
- the type of operations and objects that RBAC controls is dependent on the type of system in which it is implemented, for example, within a database management system, operations would take the form of and exhibit all the properties of a transaction, such as insert, update, and delete a record;
- roles can have overlapping responsibilities and privileges, that is, users belonging to different roles may need to perform some common operations. As a result, RBAC supports the concept of role hierarchies;
- a role hierarchy defines roles that have unique attributes and that may contain other roles, for instance, that one role may implicitly include the operations, constraints, and objects that are associated with another role;
- role authorization means association of user with a role, it can be subject to the following:
 - the user can be given no more privileges than is necessary to perform his/her job (principle of least privilege);

- the role in which the user is gaining membership is not mutually exclusive with another role for which the user already possesses membership (static separation of duty);
- role activation involves the mapping of a user to one or possibly many roles. A user initiates a session during which the user is associated with a subset of roles for which that user has membership. A particular role for a user can be activated if the followings are meet: the user is authorized for the role being proposed for activation; the activation of the proposed role is not mutually exclusive with any other active role(s) of the user; the proposed operation is authorized for the role being proposed for activation;
- role execution of an operation can take place only if the user is acting within an active role, that is, once it is determined that a role is part of the authorized role set for the user;
- dynamic separation of duty can be provided with RBAC as long as the following rule is satisfied: a user can become active in a new role only if the proposed role is not mutually exclusive with any of the roles in which the user is currently active;
- operation authorization can only be granted to a user if the operation is authorized for the user's proposed active role;

As shown above, RBAC is a framework of robust security policies. It can map to various organizational structures and means of conducting business. Over time, the security policies implemented under RBAC can evolve as enterprise and security requirements change.

RBAC supports several well known security principles and policies critical to any information system [A Review Paper of Role Based Access Control, 1999].

These include the enforcement of the concept of *Least Privilege* for administrators and general users, and the enforcement of conflict of interest rules that involve duty assignment and dynamic and static separation of duties. These policies can be enforced (1) at the time operations are enforced for a role, (2) at the time users are authorized as members of a role, (3) at the time of role activation during a user's active session, or (4) when a user (or an application) attempts to perform an operation on an object [A Review Paper of Role Based Access Control, 1999]. Which of the four strategies will be used depends on individual system's need and design.

3.2 RBAC Concept

RBAC simplifies administration by statically and dynamically regulating users' actions through the establishment and definition of roles, role hierarchies, relationships and constraints. This is in contrast to the conventional and less intuitive process of administering lower level access control mechanisms directly on an object-by-object basis.

Based on the standard *Proposed NIST Standard for Role Based Access Control* [Jaeger, Michailidis and Rada, 1999], the basic concept of RBAC is that: users are assigned to roles, and permissions are assigned to roles; a user acquires

permissions by becoming a member of one or more roles. A large organization may have a few thousands users, but only a few hundred roles, and some permissions of the system are common to most of the users. RBAC groups the users in different roles, which makes it simple for administrators to control the users and permissions, and reduces management operations which in turn reduce operational mistakes.

The RBAC standard [Jaeger, Michailidis and Rada, 1999] includes three models: Core RBAC, Hierarchical RBAC, and Constrained RBAC. Core RBAC is the basic model as it contains all the essential aspects of RBAC. Hierarchical RBAC is introduced to accommodate the natural role hierarchy structure found within an organization. Constrained RBAC includes Static Separation of Duty Relations and Dynamic Separation of Duty Relations. The aim of Constrained RBAC is to solve the issues relevant to duty/permission conflict of interests. Once the Core RBAC model is understood, the other two models (Hierarchical RBAC, Constrained RBAC) can be understood and deployed by adding extra features.

It is noted that not all RBAC features are appropriate for all environments nor do vendors necessarily implement all RBAC features.

The following section is the descriptions of the RBAC concept and models based on the standard [Jaeger, Michailidis and Rada, 1999].

3.3 RBAC Components

3.3.1 Core RBAC

Based on the proposed standard [Jaeger, Michailidis and Rada, 1999], Core RBAC consists of the essential aspects of RBAC, and includes a set of elements: **USERS**, **ROLES**, **OBS**, **OPS**, **PERMISSIONS**, **SESSIONS**, and a number of assignment relationships as shown in Figure 3.1.

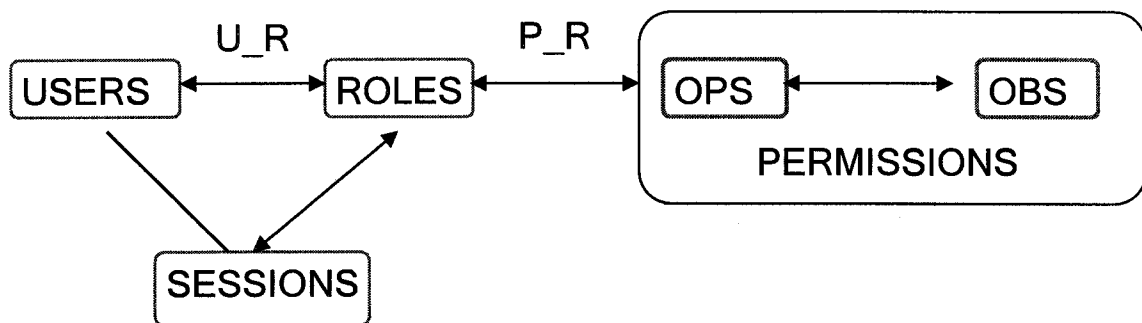


Figure 3.1 Core RBAC

- A *user* is a human being. For instance, Jen is an employee of a bank, so she is a user of the banking computer system. The concept of a *user* can also be extended to include a machine, a network, or an intelligent autonomous agent, though this is not considered in this study.

- A *role* is a job function that is specified within an organization with some associated semantics regarding the authority and responsibility. For example, account manager is a role in a bank.
- An *object* is a computer resource, such as a file, a directory, a program, a printer or software.
- An *operation* is an executable image of the object. Upon invocation, it executes some functions for the user. The types of operations and objects that RBAC controls are dependent on the type of application systems in which they will be implemented. It includes read, write, and execute.
- A *permission* is an approval to perform an operation on one or more system protected objects. For example, *reading the file x* is a permission.
- A session is a set of active roles. It is a subset of the whole set of roles assigned to a user. One user has one session. Only one session can be active at a time. A session allows selective activation and deactivation of roles. During a session, the user can successfully perform any operations permitted by these roles' duties.

The important part of RBAC is the user - role assignment and permission - role assignment relations (Figure 3.2 and Figure 3.3). Figure 3.2 shows a number of users are assigned to a single role, and figure 3.3 shows a number of permissions are assigned to a single role.

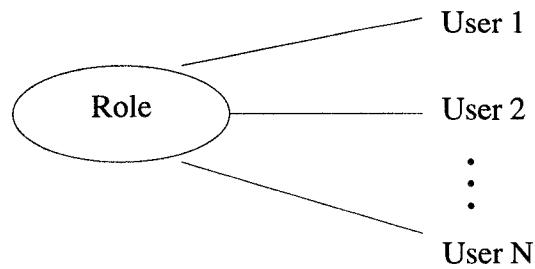


Figure 3.2 Users are assigned to a role

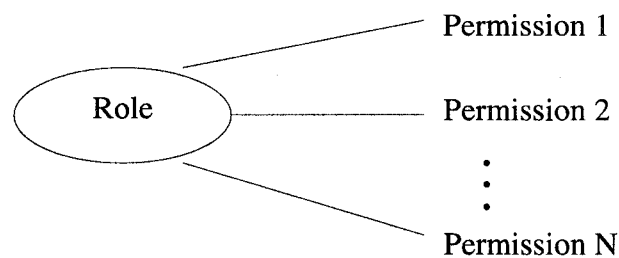


Figure 3.3 Permissions are assigned to a role

Both user - role and permission - role assignment relationships can be many-to-many (Figure 3.4), that is, the same user can be assigned to many roles, and a single role can have many users. In the same way, a single permission can be assigned to many roles, and a single role can have many permissions.

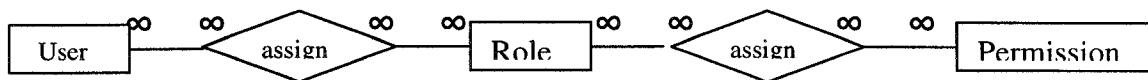


Figure 3.4 User - Role - Permission relationships

A session is a mapping of one user to a set of roles. Each session is associated with a single user and each user is associated with one or more sessions. A user

may be assigned to many roles, but not all of those roles are necessarily put in a session. Only the roles that are active are in a session, that is, a session contains the activated roles of a user.

3.3.2 Hierarchical RBAC

Hierarchies are a natural means of structuring roles to reflect an organization's authority and responsibility. It has been noticed that in an organization roles can have overlapping responsibilities, that is, users belonging to different roles may have some common permissions. In other words, within an organization there might be a number of general permissions that are performed by different roles. As such, it would be inefficient and administratively cumbersome to specify repeatedly their general permission - role assignments. To support an organization's operational structure and improve efficiency, the proposed RBAC standard incorporates role hierarchies; Hierarchical RBAC becomes a key aspect of RBAC models (Figure 3.5).

Role hierarchies define an inheritance relation between roles (Figure 3.6a). For example, r_1 and r_2 are two roles in an organization, if all permissions of r_2 are also permissions of r_1 , then it is called r_1 inheriting r_2 . If no role lies between r_1 and r_2 , it is said that r_1 is an immediate descendant of r_2 (Figure 3.6b).

Role hierarchies allow senior roles to acquire the permissions of their juniors. The proposed RBAC standard recognizes two types of role hierarchies. They are the General Role Hierarchy and Limited Role Hierarchy.

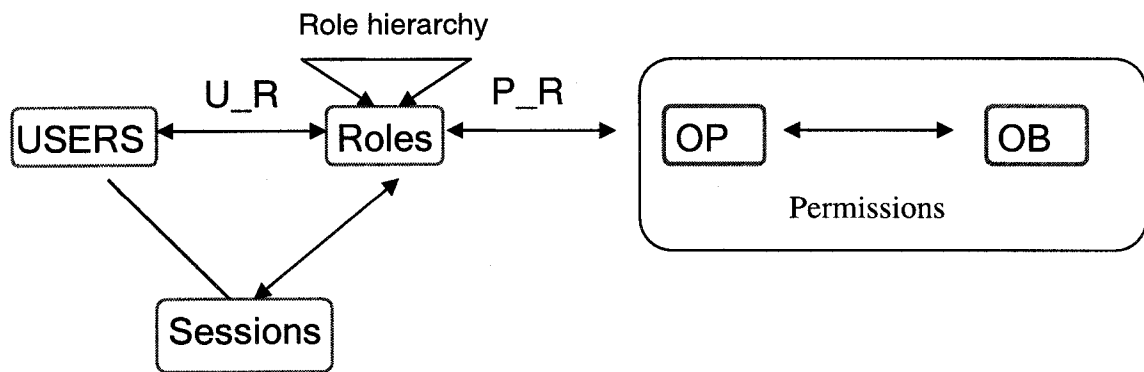


Figure 3.5 Hierarchical RBAC

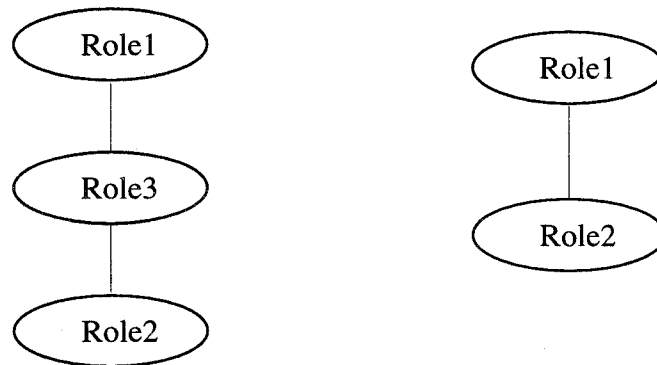


Figure 3.6a. Role1 inherits role2 and role3 b. Role1 is an immediate descendant of role2

The General role hierarchy supports multiple inheritances. It allows inheritance of permissions from two or more roles and inheritance of user membership from two or more roles (Figure 3.7). In this hierarchy, role 1 (R1) has permission P_{R1} , and role 2 (R2) has permission P_{R2} , but there is no common permission between R1 and R2; so there can be no sharing of resources between two roles which are on different branches of a tree structure. It is also noted that the top role may obtain too much power because it can inherit all permissions and membership from other individual roles.

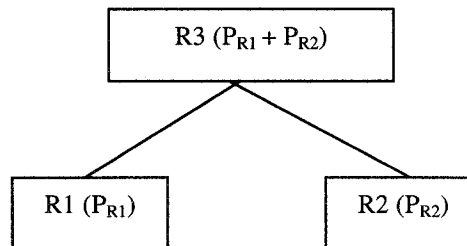


Figure 3.7 General Role Hierarchy

Limited role hierarchy imposes restrictions resulting in a simpler and inverted tree structure (Figure 3.8). In this tree structure, a role may have one or more immediate ascendants, but is restricted to a single immediate descendant. In Figure 3.8, department's permission (P_D) is basic information for all users in the department; role1 and role2 have their own permissions (P_{R1} and P_{R2} , respectively) and have common permission P_D . The same ideas are for the role3.

To maximize the use of hierarchy structures, the lattice hierarchy structure is introduced; it is a combination of the general role hierarchy and limited role hierarchy. The lattice structure takes advantage of the above two hierarchy structures. It prevents one role and its associated users from having monopoly power of access, and also allows reasonable amount of inheritance. Figure 3.9 is an example of the lattice hierarchy structure; it shows role 5 (R_5) has most of permissions from the roles in the department, but not all. Role 5 is on the top of the structure, but it just inherits the necessary functions for it to perform its duties. Role 5 shares the permissions of role 1, 3, and 4, as well as the basic ones of the department, but it does not have the permissions that are designed for the role 2. This hybrid structure will be used in this thesis to design an e-education system.

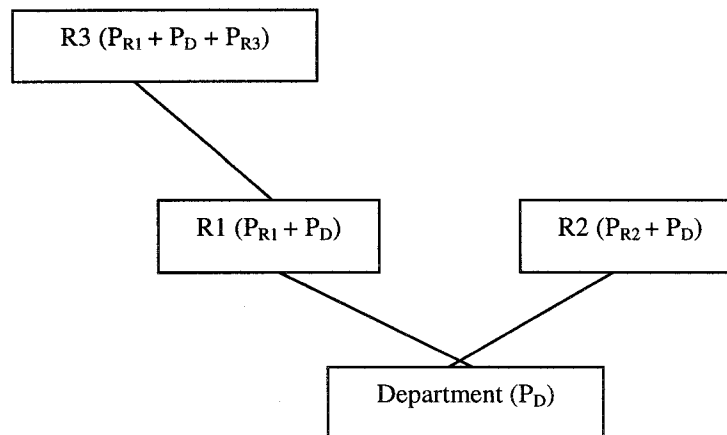


Figure 3.8 Limited Role Hierarchy

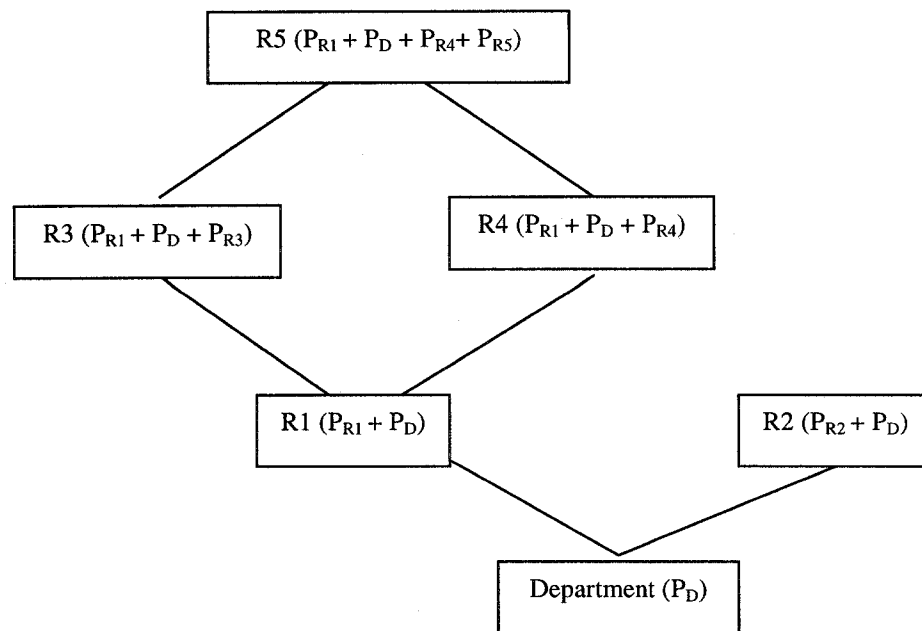


Figure 3.9 Lattice Role Hierarchy

3.3.3 Constrained RBAC

Within an organization certain roles may be mutually exclusive. Normally, the conflict of interest roles would not be allowed to be assigned to the same user, but sometimes, some key users are allowed to have mutually exclusive roles. For example, an employee of a bank can also be a client of this bank; in a hospital, a doctor can be patient in his or her hospital. When a user acts as a client of the bank, the user can only view his/her financial records, but not allowed to modify them; when the user acts as an employee of the bank, he/she can not access their own bank account, even their family member's, but he or

she will have the right to modify customers' financial records. In order to address these types of issues, the Constrained RBAC is introduced (Figure 3.10 and Figure 3.11).

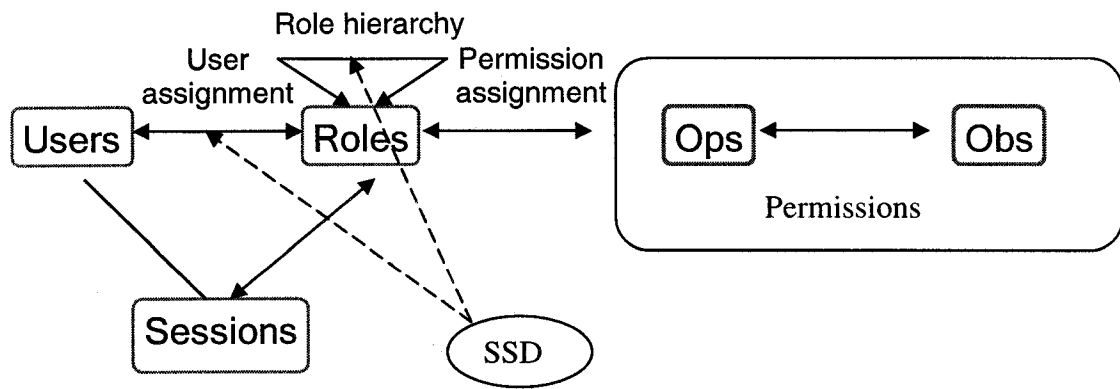


Figure 3.10 Static Separation of Duty relations (SSD)

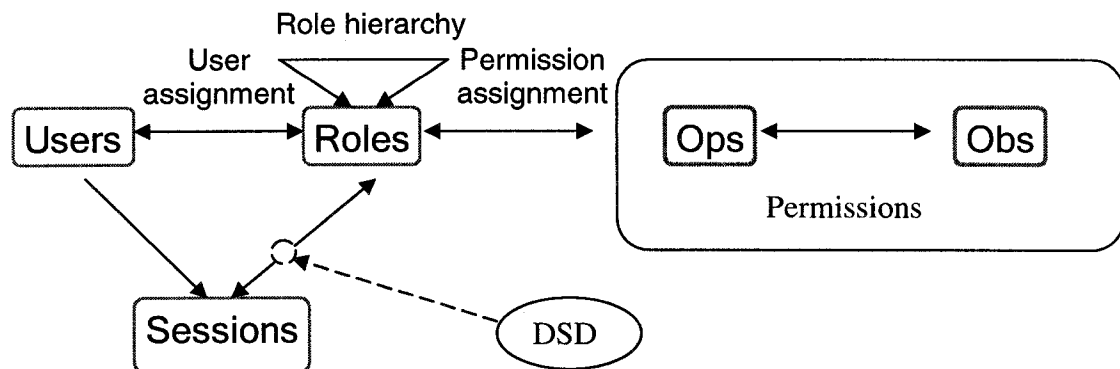


Figure 3.11 Dynamic Separation of Duty relations (DSD)

The Constrained RBAC is based on the Core RBAC and Hierarchical RBAC. Conflict of interest in a role-based system often arises as a result of a user

gaining permissions associated with conflicting roles as indicated in the above example. The Constrained RBAC proposes two common ways to prevent the conflict. 1) Static Separation of Duty (SSD) (Figure 3.10) enforces constraints on the assignment of users to roles: one user can not have mutually exclusive roles at all. 2) Dynamic Separation of Duty (DSD) (Figure 3.11) limits the permissions that are available to a user by placing constraints on the roles that can be activated within or across a user's sessions. Some users can have mutually exclusive roles but in different sessions, and only one session can be invoked at the same time. It is observed that DSD is more flexible than SSD.

In summary, the Core RBAC model consists of the essential components, and it provides the most basic elements for any RBAC based system. Hierarchy RBAC and Constrained RBAC models are built based on the Core RBAC. Users can incorporate the three models and build their own RBAC model, as shown in Chapter 4 of this study where a hybrid RBAC model is built for an e-education system.

Chapter 4

Requirements and specification of RBAC based e-education system

The World Wide Web and Internet have changed the way people acquire and share information and knowledge. People become comfortable enough to take online courses, called e-education, e.g., WebCT and Moodle. It is a professional tool allowing students and instructors to involve in courses through Internet.

An e-education system can be considered as a virtual university. It has most of the framework of a traditional education organization, except for the real classrooms, campus, and face-to-face communication. Compared to traditional education, e-education has the following advantages:

- Students do not need to physically attend lectures or labs
- Students can study at different times
- They can review course materials many times
- They can take the study in different locations in the world. E-education makes it possible for many people, who may not be able to attend the traditional education, to get training
- The same flexibility and benefits are applicable to education instructors

- Save costs for both students and the virtual university, because no real campus and classrooms are required

Due to the above advantages and continuing advances in Internet technology, e-education is becoming more and more popular.

A typical e-education system has a variety of components, such as roles, users, and education resources. In order to effectively manage the complex relationships among these components, a sophisticated access control mechanism is required. From the discussion of RBAC characteristics in Chapter 3, it is clear that RBAC models are the most optimal solution for e-education system's access control. The e-education system designed and implemented with RBAC concept can have a dramatic improvement in resources management cost and security.

The objective of this study is, therefore, to design and develop an e-education system using the RBAC concept and models, and to demonstrate that RBAC can provide efficient and secure management for e-education system.

4.1 E-education system requirements

An e-education system is a simplified version of a real university. It has a less complex organizational structure. The essential entities of an e-education

system include students, instructors, system administrators, and accountants. Some less crucial entities could be teaching assistant, etc. These entities of the e-education system have the same operational functions as those in a real university.

An e-education also has system resources that include students and instructors' personal information, student academic records, course curriculums, financial status, etc.

An e-education system should be able to accept new users' registration through Internet. It will give different groups of users different access rights to the system resources. For instance, an individual student's personal data can only be accessed by that student; one particular course material can only be accessed by the students who have registered for the course within a specific period of time, and their instructors. The system should be able to provide reliable and secure online services. It should be available anytime and anywhere via Internet.

In summary, these requirements may be expanded into the following functions:

- (1) **Online registration:** the e-education system must allow new users to register; the registration form must be available to take new users' name, mailing address, phone, e-mail address, payment, etc.
- (2) **Remote access via Internet:** the system must be available for users to access anytime, anywhere, 24 hours a day, 7 days a week through a web

browser. The access control should be enabled via the authorized login ID and password.

- (3) **Resource access for learning:** once a user logs on to the system, the system should display corresponding data for that user. Similarly groups of users can only access the resources allocated to that groups. For instance, student users are only allowed to view the information that is related to them; they are not allowed to view the information of the courses that they have not registered for; students are not allowed to view other students' personal data. In the same way, an instructor can only view and evaluate the data of students who are registered for his/her courses. The system must also be able to identify users' roles; if the login user is an administrator, the system will direct the user to the corresponding administrative WebPages; if the user is a student, the system will direct him or her to the student learning sites.
- (4) **Create and post course materials:** the system should allow instructors to create and post course curriculums, assignments, quizzes, etc., and distribute the assessed works back to students.
- (5) **Submit work for evaluation:** once completing a work, such as assignment, quiz, and exam, a student should be able to submit it to the appropriate instructors for an assessment.
- (6) **Manage system resources:** the system resources, such as, personal records, course curriculums, assignments, etc. must be managed and maintained up-to-date by administrators.

4.2 E-education specification

To meet the above requirements, the following sections give explicit specification of the e-education system. In general, there are two streams of specifications.

One stream specifies how users, including students, instructors, teaching assistants, and accountants, access the e-education system. The other stream specifies how the system administrators manage the system resources.

4.2.1 User access function specification

Students, teaching assistants (TA), and instructors can only access their own personal information, academic data and payment status. Students are allowed to get information (assignments, handouts, quizzes, etc.) from only the courses for which they have registered. Instructors create course materials (handouts, quizzes, assignments, exams, etc.), and make them available to the registered students. The following tasks are performed by instructors, TA, and students, respectively.

- Instructors:
 - Create course activities, including references and tutorial materials, homework, quizzes, and examinations
 - Read students responses
 - Grade students responses
- Students:

- Select a course to register in
- Create responses to activities posted by instructors
- Submit completed responses to such activities
- View their own grades/marks/payments
- TA
 - Read students assignments/quizzes/exams in designated courses
 - Write students grade for the designated courses

Given these tasks, the e-education system must protect the privacy and integrity of these course data, and the following access control requirements must be upheld.

- Instructors:
 - Can only access data associated with their own courses
 - Can read/write activities that are under development
 - Can read/write activities that have been assigned
 - Can assign students to activities
 - Can only read responses that have been submitted by students
 - Can read/write the assigned courses grades
- Students:
 - Can only access data associated with own courses
 - Can read/write activities that are under development for their courses
 - Can read/write activities that have been assigned for this course
 - Can only read responses that have been submitted

- Can not write responses that have been submitted for assessment
- Can read own grades
- Can not write their own grades
- TA
 - Can read students grades for the designated/assigned courses.
 - Can read activities that have been designated/assigned
 - Can grade activities that have been designated/assigned

4.2.2 Administrative function specification

Administrators play a critical role in the e-education system management. Their functions are to ensure efficient and effective data communications among different users (students, instructors, accountants, and TA), and to ensure and maintain integrity and security of system data. For instance, course materials must be protected; unnecessary access is not allowed. Each course material should be strictly protected and only allowed access by the designated instructors, TAs, and registered students.

Administrators are required to confirm a new user's registration information (e.g., names, address, payment, courses interested) and accept the new user's registration based on online provided data. If the new user's registration is confirmed and accepted, the administrators will issue a set of logon username and password to this user. Based on the request at the time the user registered,

administrators assign the user the requested role, which will be the user's default role.

In summary, administrators' tasks/functions should include the following:

- Take new users' registration requests
- Create and maintain users' profiles
- Confirm and add system resources
- Control users' access to system resources

In order to provide a high quality and user satisfied service (reliable, secure, available anytime and anywhere), the good system design is very critical, though Internet traffic condition and network connectivity (reliability and bandwidth) are also important for a successful e-education, which are beyond the scope of this study. From the discussion of RBAC characteristics in previous chapters, it is obvious that incorporation of RBAC models into the e-education system design can provide the power for an efficient and effective system. The following section is a discussion of e-education system specification with RBAC concept and models.

4.2.3 E-education with RBAC

As described in Chapter 3, RBAC has five basic components: users, roles, objects, operations, and permissions. These components and their functional relationships can be fully demonstrated in the e-education system. In the e-

education system, all individual participants are users. Student, faculty, TA, administrator, account manager are the roles in this system. Different users can be assigned to these roles (Figure 4.1). The various resources, such as course assignments, quizzes, etc. are the set of objects. Read/write/execute these objects are the set of operations.

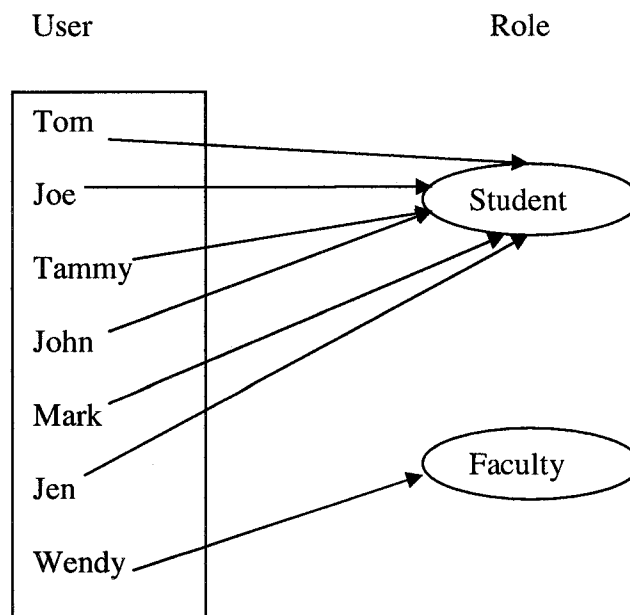


Figure 4.1 Various users and roles, and their relations

The e-education system has roles that may have common permissions (Figure 4.2). If these common characteristics can be inherited among the roles, the design and implementation will become more efficient and simpler. As shown in Chapter 3, this inheritance requirement is one of the key features of hierarchical RBAC.

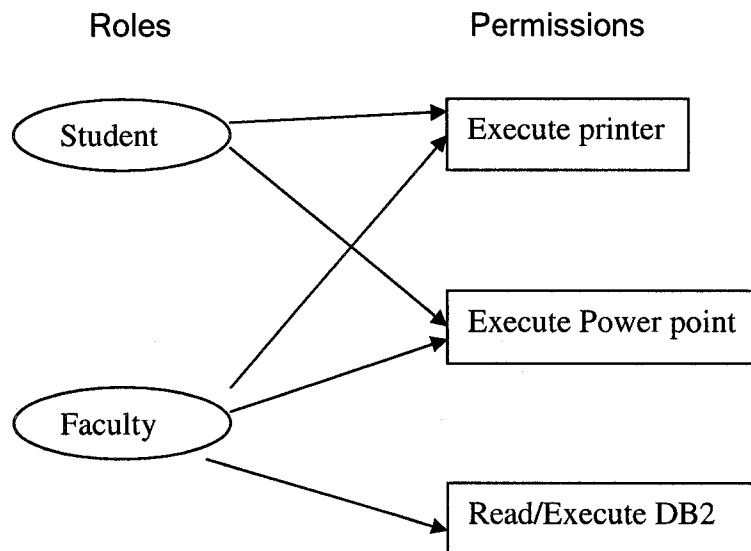


Figure 4.2 Roles sharing common permissions

In the e-education system, a great deal of permission access and inheritance should be constrained. For example, TA can inherit student role permissions, it is allowed to access some of the student role's permissions, but is not allowed to access students' personal information. Similarly, a system administrator can inherit faculty's role permission, but is not allowed to access faculty's salary information. Constraint RBAC can be used to address these types of issues.

The e-education system is required to handle a large number of users' access information and also maintain a large amount of course data. Based on the discussion in Section 4.1, it is clear that RBAC can be used to improve e-education system application development.

Based on the above requirement analysis and the discussions in Chapter 3, this study proposes a hybrid model, it is a combination of the standard Core RBAC model, Dynamic Separation of Duty Constrained RBAC model, and Hierarchy RBAC model. Figure 4.3 illustrates this proposed model. It is most similar to the standard Constraint RBAC model (dynamic separation of duty) (see the details in Chapter 3). Comparing to the standard Constraint RBAC model, the model here also puts constraints on the role hierarchy relations. It categorizes the permissions associated with a role into two groups: private permissions and public permissions. Only the public permissions can be inherited. This not only improves privacy but also reduces the number of conflict roles.

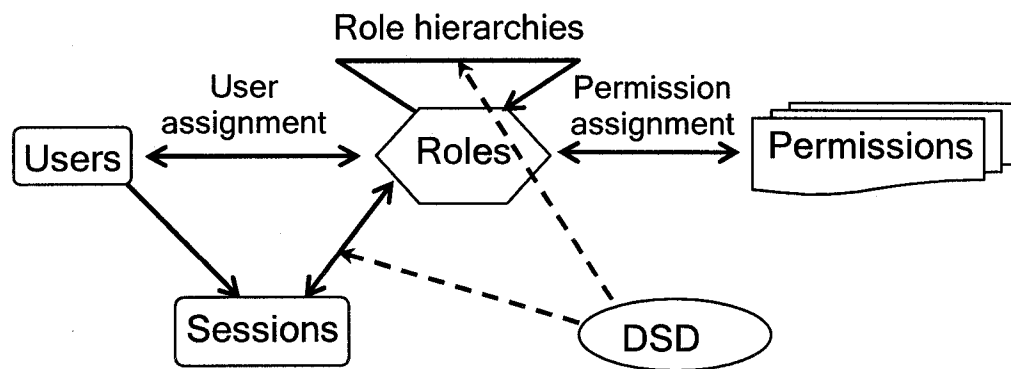


Figure 4.3 Complex RBAC model of this study

In summary, this Chapter gives a requirement analysis and specification for e-education. The requirement analysis and specification are based on the standard RBAC concept and models. It proposes a hybrid RBAC model for the e-education. In the following chapter a detailed design will be given based on this model and the above requirement analysis and specification.

Chapter 5

Design of the RBAC based E-education System

The effectiveness, robustness and efficiency of an e-education system largely depend on system design. In this study the e-education system is designed and implemented under the RBAC framework and is expected to provide enhanced security and improved management.

This chapter gives a description of the design of the e-education system, based on the analysis of requirements and specifications discussed in the previous chapter. This chapter contains three parts: the first part illustrates the incorporation of the three RBAC models in the e-education system, including the enforcement of security and privacy, the second part discusses the design of user interfaces, and the last section discusses the design of system functions.

5.1 RBAC based e-education system

This section gives step by step explanations on how the e-education system is designed based on RBAC concept and models. The design starts with the Core

RBAC model, followed by the Hierarchical RBAC model and the Constrained RBAC model in sequence.

5.1.1 Core RBAC based e-education system

5.1.1.1 System entities

With the Core RBAC model, the e-education system is designed to have five basic groups of elements called users (USERS), roles (ROLES), objects (OBS), operations (OPS), and permissions (PRMS). The combination of an object and an operation forms a permission entity. In order to control users' access to the system resources, a user is required to be a member of one or more roles, and a permission is assigned to one or more roles.

5.1.1.1.1 USERS

The USERS represents the individual participants in the e-education system. Any individual who has a set of valid logon username and password is a user of the system, such as Jen, Wendy, Tom, Mark, Frank, Joe, etc., who has a valid logon username and password. Different users have different access rights to system resources depending on their functions in the education system. A large number of users will use the e-education system in the student role. A relatively small group of users will perform in the instructor role. A few users will be in the administrator role to perform system management functions.

5.1.1.1.2 ROLES

The ROLES in the e-education system may include student, academic administrator (department head, dean), registrar's officer, system administrator, faculty, teaching assistant (TA), and account manager. Because an e-education system normally has some resources that should be accessed by all users, a global role is needed to oversee the public resources, such as the introduction of education programs. Every user in the education system will at least be assigned to this global role, such that the public resources can be shared.

Because this study is an implementation of the RBAC concept, the following six roles are chosen and designed. They are the most important roles in the e-education, and they can cover all the RBAC concept (hierarchy, constraints).

Global user - this role allows all users to share the e-education system's basic information. This is the most basic role. On a hierarchical tree relationship it will be at the bottom of the tree, thereby allowing inheritance. For example, the student role can inherit this role's permissions.

Student - a student role is an abstract representation of users who join the e-education system to learn. Users who are assigned this role can access more resources than a user who is assigned only the *global role*, for example, read the specific courses' information and submit their response.

Faculty - this role is an abstract representation of participants who deliver information and knowledge to the students, including students' assignments, marks, etc. Relatively, this role will have a smaller number of users than the student role.

Teaching assistant (TA) - this role is designed to help faculty to evaluate students' activities, such as marking assignments and quizzes.

Account manager - this role represents a group of users who monitor students' payment status (debit and credit), faculty and administrators' payroll balances. This role is restricted to accessing only financial data of the system's users. For example, an account manager can only access students' financial accounts and modify students account's balance but can not view students' academic records.

Administrator - this role represents a group of users who maintain and update all the RBAC components and assignment of relationships among these components. It has the right to modify the users and roles information, set up permissions and assign these different roles.

Among these roles, student, faculty, account manager, and administrator are the key roles of this study. Other roles can be flexible. Different e-education systems may focus on different interests; therefore, they may have a different set of roles.

5.1.1.1.3 Objects

Objects in the e-education system include individual courses, assignments, quizzes, personal records, etc. Their existence depends on the nature of the education program. If the education program offers more courses and has a large number of users, it is highly likely that the system will have more objects. In this design, for simplicity, the objects are individual web pages/links, i.e., HTML files.

5.1.1.1.4 Operations

Operations are the executions of read, write, and modify. They are the actions performed on different objects. For instance, reading an assignment and modifying a grade are both actions.

5.1.1.1.5 Permissions

Permissions represent access rights that a user is allowed to perform on certain system objects. Any permission is a combination of an operation and an object. It indicates what action can be performed on an object.

5.1.1.1.6 Sessions

For test purposes, three sessions are set up for the e-education system, though many more can be set if required. A default session will be created at the time the user is accepted for registration. All the active roles assigned to the user will be put in the default session. A second session will not be established unless two or more conflict of interests roles are required to be activated. For instance, a user has a default session with the student role, and the student role and administrator role are in conflict of interests; when the user is assigned the TA role and requires this TA role to be active, this TA role will be put in the default session, but when the user is assigned the administrator role and requires to activate this role, this role is not allowed to put in the current default session, a new session must be created to contain this role.

5.1.1.2 User/role assignment

In the education system, each user must have at least one role. The user-role assignments are performed by administrators. One user can have many roles, and one role can be assigned to many users. For example, while the user Jen is assigned to a student role, she can also be assigned to a TA role.

5.1.1.3 Permission/role assignment

In this study, each role is assigned at least one permission. Table 5.1 shows the assignment relationships between permissions and roles.

Table 5.1 roles and their permissions of the e-education system

ROLES	Objects accessed
Global users	<ol style="list-style-type: none"> 1. basic information about the e-education system 2. course information
Students	<ol style="list-style-type: none"> 1. basic information about the e-education system 2. course information 3. assignment 4. quiz 5. handout 6. exam 7. grade (private)
Teaching Assistant (TA)	<ol style="list-style-type: none"> 1. basic information about the e-education 2. course information 3. assignment 4. quiz 5. handout 6. exam 7. students marks
Faculty	<ol style="list-style-type: none"> 1. basic information about the e-education 2. course information 3. assignment 4. quiz 5. handout 6. exam 7. students marks 8. evaluate students works
Account manager	<ol style="list-style-type: none"> 1. basic information about the e-education 2. course information 3. students account
Administrator	<ol style="list-style-type: none"> 1. basic information about the e-education 2. course information 3. assignment 4. quiz 5. handout 6. exam 7. students marks 8. can modify the assignment due date 9. students account 10. management duty

5.1.2 Hierarchical RBAC based e-education system

In the e-education system, some public permissions are shared by all roles, for example, the basic information about the e-education. It would be better to have this kind of information inherited by all roles. It is also noted that certain roles have hierarchical relationships, for instance, a hierarchical relation between the faculty role and TA role. These observations are not handled in the Core RBAC model. Therefore, hierarchical RBAC is needed in the system design. Figure 5.1 shows the hierarchical relationships among the six roles.

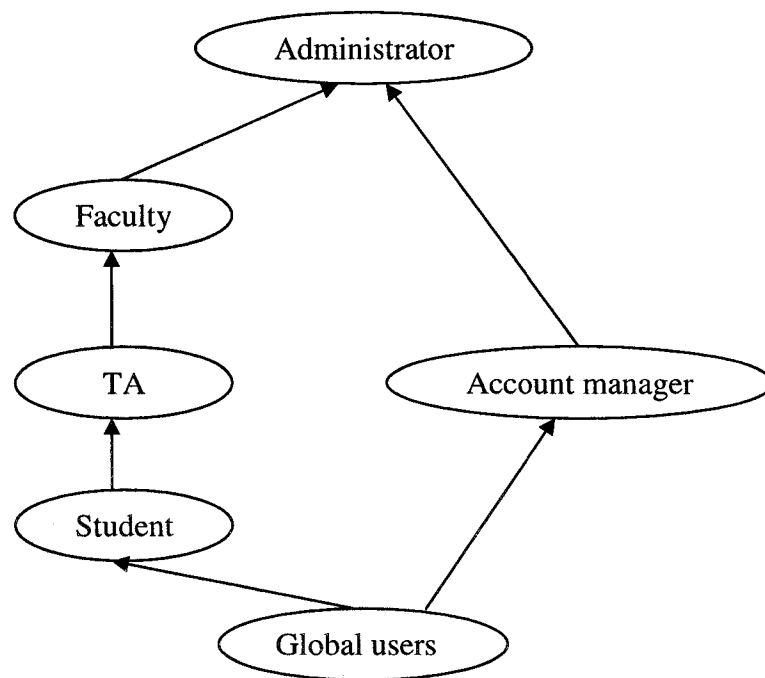


Figure 5.1 Role Hierarchical relationships in e-education system

With the role hierarchy being established, the amount of permissions/roles assignments made by administrators is reduced. The less the assignment of permissions to roles means the less the chance for administrators to make mistakes, and in turn, the less the chance for users to maliciously access resources. As a result, system security will be enhanced. The permissions/roles assignments in Hierarchical RBAC are included in table 5.2.

Table 5.2 Roles and their permissions in hierarchy RBAC model for e-education

ROLES	PERMISSIONS
Global users	1. basic information about the e-education (read) 2. course information (read)
Students	1. Global users' permissions (inherited) 2. response for assignments (read/write) 3. response for quizzes (read/write) 4. handout (read) 5. response for exams (read/write) 6. grade (private) (read)
Teaching Assistant (TA)	1. Student's permissions (inherited) 2. Students marks (read/write)
Faculty	1. TA's permissions (inherited) 2. the assignment due date (read/write)
Account manager	1. Global users' permissions (inherited) 2. Students account (read/write)
Administrator	1. All role's permissions (inherited) 2. Management duty (read/write)

As shown in the above table, the basic difference between the Core RBAC based system design and Hierarchical RBAC based system design, is to introduce hierarchical relationships, which results in less assignment processes.

5.1.3 Constrained RBAC based e-education system

The constrained RBAC model is introduced into this study because of three issues: privacy, security, and conflict of interest roles.

It is recognized that some permissions should not be inherited. For example, student marks and grades are private data for each student, TA and faculty should not view them all after certain period of time. It is better that this information is kept private and not inherited.

In order to prevent an administrator from having monopoly actions in managing the education system resources, the most important management processes, such as permissions and roles assignments, are required to be performed by two or more administrators. Less important administrative duties, such as assigning basic role to new users, are allowed to be conducted by one administrator. This issue is addressed by designing three levels of administrator roles. The level one role is allowed to perform the basic administrative duties; the level two role gains more administrative duties; the level three role is allowed to carry out all the administrative duties, but it requires two three level administrator users to perform the duties (detailed discussion is available in Chapter 6).

Normally a user has a set of active roles and they are managed by one session. But sometimes, it is natural that a user may have some roles that are mutually exclusive in functions. The organization policy of a well designed RBAC system

should not allow the user to access the conflict of interest roles at the same time. These roles cannot be placed in the same session; they should be put into a separate session. During one specific login, the user can only perform the duties related to one session, and not the other session. The user can only access the other session through a new login.

This Constrained RBAC education system design will be implemented in Chapter 6 of this study.

5.2 E-education system interface design

A universal interface is designed. It represents the homepage to the e-education system. The interface includes an introduction to the e-education, the login frame through which all users enter the system, and the register link that will take a new user to a registration page. All users, including administrators, students, faculty, etc., enter the system through the login frame.

Then, based on the system requirements, two classes of interfaces are designed. One is for the administrators to manage the system resources, and the other one is for the regular users (students, faculty, TA, Account Manager) to use the system.

When an existing user enters her/his user name and password, the system will confirm the user's identity. If the logon information is correct, the corresponding interface will be returned with available resources; if the information is not correct, the system will ask the user to logon by entering correct logon information, or suggest that user register. When a user has the role of performing the level three administrative duties, the user will be directed to the level three administrative duties logon interface (Figure 5.2).

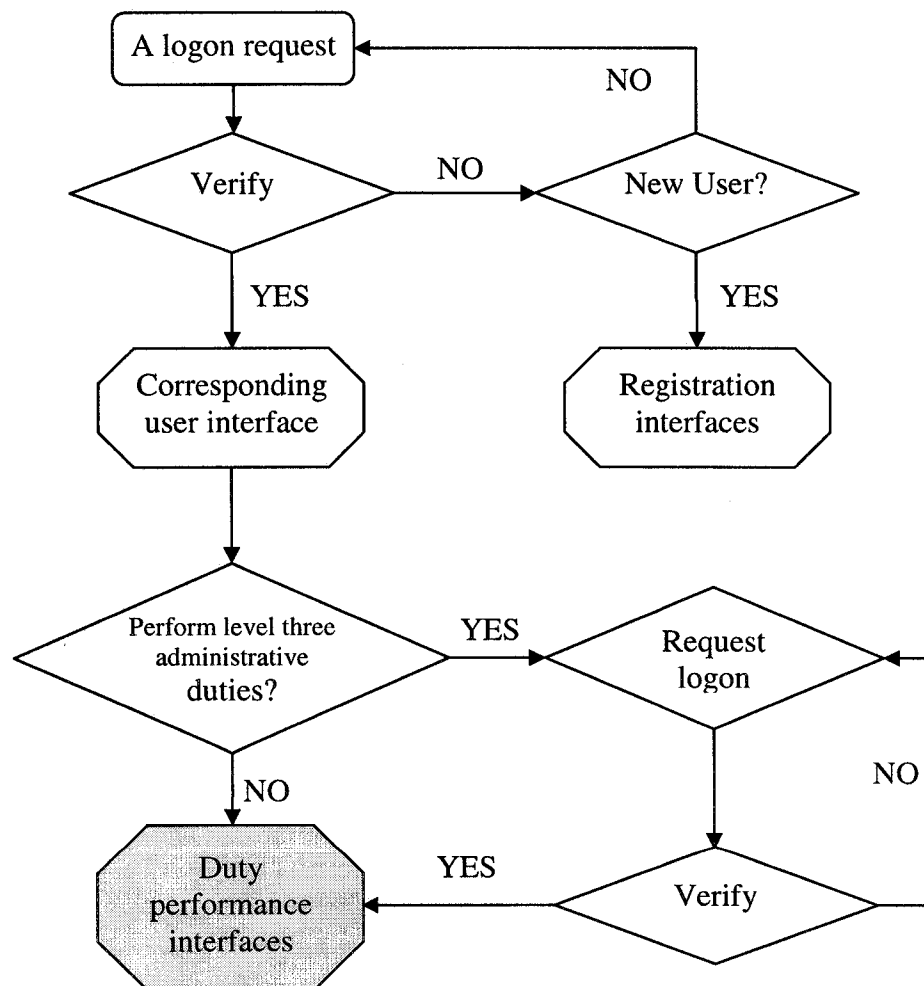


Figure 5.2 User logon confirmation processes

Each user has one initial role and one default session. After a user logon, the system will display all the sessions and roles the user has. If the interface shows more than one session, that means the user has conflicting roles that are in separate sessions. For one particular logon, the user can only choose to go with one session. If the user wants to perform duties upheld in another session, the user has to logout and make a new logon request.

5.2.1 Administrative support interfaces

To address the security concern discussed in section 5.1.3, a special login interface is designed for level three administrators role. This interface is for administrators to manage the most important system resources. It provides two sets of usernames and passwords, and requires two level three role administrators to appear and logon at the same time.

Because of the required duties, administrator interfaces include: accept/confirm new user registration, user management, role management, user/role assignment and permission/role assignment, and session management (detailed descriptions are available in Chapter 6).

The administrative interfaces will allow administrators to accept the new users, place new users' registration information into a database table, and at the same time create a session for the user.

Administrators can assign a role to a user. In order to make the role active, the administrator will put the role into one of the user's sessions where no conflict of interest roles exist.

The administrative interfaces also provide tools for administrators to assign permissions to role, add/remove roles to/from the system, add/remove permissions to/from roles, and create hierarchy relations and conflict role relations.

5.2.2 Academic support interfaces

The interfaces for regular users are simple and straightforward. They display the data that a registered user requests and entitles. They allow users to view course information, personal academic and financial records, and feedbacks/evaluations from faculty; they also allow users to submit assignments and exams for evaluation. All the interfaces for regular users will be *html*/files.

For the student role, three basic types of interfaces are designed. One is a very basic interface, allowing students to view course information, and faculty's up-to-date postings. The second type of interfaces are about assignments; a time limit is set for each assignment, and each assignment has a fixed due date. Students can only submit the answers to the assignment before the due date. The third

one is about quizzes and exams; students must complete and submit the answers to the exams within a specified period of time. After the specified time limit, submission is disabled. The time is counted from the time users open the pages.

5.3 Systems functionality

5.3.1 Administrative functions

The administrative functions include creation and maintenance of system entities: USERS, ROLES, OPS (operations), OBS (objects), and PRMS (permissions). Of these elements, the existence of OPS and OBS is essentially for PRMS. In the e-education system, OBS represents the individual webpage, for example, a webpage showing the assignment of a particular course, or a webpage showing the personal records of a student. OPS represents read/write/modify actions to the web pages.

Administrators create and delete OPS, OBS, USERS, and ROLES, and establish and maintain relations between OPS and OBS, forming PRMS, between roles and users, and between roles and permissions.

Below are a formal definition of the above entities and relationships that are used in this thesis.

- $USERS$, $ROLES$, OPS , OBS , and $SESSIONS$ represent the set of users, roles, operations, objects, and sessions respectively.
- $PRMS = 2^{(OPS \times OBS)}$, the set of permissions.
- $U_R \subseteq USERS \times ROLES$ is a many-to-many assignment mapping between users and roles.
- $P_R \subseteq PRMS \times ROLES$ is a many-to-many assignment mapping between permissions and roles.
- $U_{assigned} : (r : ROLES) \rightarrow 2^{USERS}$, the mapping of role r onto a set of users.
Formally, $U_{assigned}(r) = \{u \in USERS \mid (u, r) \in U_R\}$
- $P_{assigned} : (r : ROLES) \rightarrow 2^{PRMS}$, the mapping of role r onto a set of permissions. Formally, $P_{assigned}(r) = \{p \in PRMS \mid (p, r) \in P_R\}$
- $U_S : (u : USERS) \rightarrow 2^{SESSIONS}$, the mapping of user u onto a set of sessions.
- $S_R : (s : SESSIONS) \rightarrow 2^{ROLES}$, the mapping of sessions s onto a set of roles.

Required administrative functions for different entities of the e-education system are listed in the Table 5.5. The notation used in the following specifications is basically the Z notation. The representation of a schema is: Schema-Name (declaration) \triangleleft Predicate1; ...; PredicateN \triangleright . An explanation of the symbols is listed in Table 5.3 below:

Table 5.3 Z notation symbol explanation

Symbol	Example	Meaning
\in	$A \in B$	Membership. Is true if object A is a member of B.
\notin	$A \notin B$	Non – membership
\mapsto	$A \mapsto B$	Maplet. A graphic way of expressing the ordered pair (A,B)
\emptyset	\emptyset	Empty set. It has no members
\setminus	$A \setminus B$	Set difference. The members of $A \setminus B$ are those objects which are members of A but not of B
\cup	$A \cup B$	Set union. The members of the set $A \cup B$ are those objects which are members of A or B or both
\forall	$\forall s \bullet P$	Universal quantifier. Is true, if whatever values are taken by the variables introduced by s which make the property of S true, the predicate P is true as well
'	$USERS'$	Represent the <i>after</i> state of <i>USERS</i>
\Rightarrow	$A \Rightarrow B$	Implication
\subseteq	$A \subseteq B$	Subset relation. A is a subset of B
\wedge	$A \wedge B$	Conjunction
:	$A : B$	A is a member of set B
Note: the symbols and explanations are from Spivey (1989)		

An explanation of the informal notation of an example function is briefed in Table 5.4 below. The function is to delete a user from the system. It first checks if the user (u) exists in the system. If so, the system will check all the sessions which belong to this user and delete these sessions. Then the system updates the sets:

U_R , $U_{assigned}$, and *USERS*.

Table 5.4 Explanation of an example function described in Z notation

The function: delete a user	
$\triangleleft u \in USERS$	Precondition: check if u exists in system
$U_S' = U_S \setminus \{u \mapsto U_S(u)\}$	Remove the user-sessions mappings of this user u , and update the U_S
$U_R' = U_R \setminus \{r : ROLES \bullet u \mapsto r\}$	Post and pre states of U_R
$U_{assigned}' = \{r : ROLES \bullet r \mapsto (U_{assigned}(r) \setminus \{u\})\}$	Post and pre states of $U_{assigned}$
$USERS' = USERS \setminus \{u\} \triangleright$	Post and pre states of $USERS$
<ul style="list-style-type: none"> - variables without ' describe the <i>pre</i> state - variables with ' describe the <i>post</i> state of an operation 	

Table 5.5 Definition and specification of administrative functions

Entities	Description	Functions
USERS	addUser (u): to add the user u , the system updates $USERS$ and U_S	$\triangleleft u \notin USERS$ $USERS' = USERS \cup \{u\}$ $U_S' = U_S \cup \{u \mapsto \phi\} \triangleright$
	deleteUser (u): to delete the user u , the system updates $USERS$, U_R , U_S , $U_{assigned}$	$\triangleleft u \in USERS$ $U_S' = U_S \setminus \{u \mapsto U_S(u)\}$ $U_R' = U_R \setminus \{r : ROLES \bullet u \mapsto r\}$ $U_{assigned}' = \{r : ROLES \bullet r \mapsto (U_{assigned}(r) \setminus \{u\})\}$ $USERS' = USERS \setminus \{u\} \triangleright$
PRMS	addObject (ob): to add the object ob , the system updates OBS	$\triangleleft ob \notin OBS$ $OBS' = OBS \cup \{ob\} \triangleright$
	deleteObject (ob): to delete the object ob , the system updates OBS and $PRMS$	$\triangleleft ob \in OBS$ $OBS' = OBS \setminus \{ob\}$ $PRMS' = PRMS \setminus \{ob : OBS \bullet op \mapsto ob\} \triangleright$
	addOperation (op): to add the operation op , the system updates OPS	$\triangleleft op \notin OPS$ $OPS' = OPS \cup \{op\} \triangleright$
	deleteOperation (op): to delete the operation op , the system updates OPS and $PRMS$	$\triangleleft op \in OPS$ $OPS' = OPS \setminus \{op\}$ $PRMS' = PRMS \setminus \{op : OPS \bullet op \mapsto ob\} \triangleright$

	makePermission(op, ob): to create the permission, the system updates <i>PRMS</i>	$\triangleleft op \in OPS; ob \in OBS$ $PRMS' = PRMS \cup \{(op \mapsto ob) \notin PRMS\} \triangleright$
	removePermission(op, ob): to remove the permission, the system updates <i>PRMS</i> and <i>P_R</i>	$\triangleleft p \in PRMS$ $PRMS' = PRMS \setminus \{p\}$ $P_R' = P_R \setminus \{p : PRMS \bullet p \mapsto r\} \triangleright$
	addRole (r): to add the role r, the system updates <i>ROLES</i> , <i>U_{assigned}</i> , and <i>P_{assigned}</i>	$\triangleleft r \notin ROLES$ $ROLES' = ROLES \cup \{r\}$ $U_{assigned}' = U_{assigned} \cup \{r \mapsto \phi\}$ $P_{assigned}' = P_{assigned} \cup \{r \mapsto \phi\} \triangleright$
	deleteRole (r): to delete the role r, the system updates <i>ROLES</i> , <i>U_R</i> , <i>P_R</i> , <i>S_R</i> , <i>U_{assigned}</i> , and <i>P_{assigned}</i>	$\triangleleft r \in ROLES$ $U_R' = U_R \setminus \{u : USERS \bullet u \mapsto r\}$ $U_{assigned}' = U_{assigned} \setminus \{r \mapsto (U_{assigned}(r))\}$ $P_R' = P_R \setminus \{p : PRMS \bullet p \mapsto r\}$ $P_{assigned}' = P_{assigned} \setminus \{r \mapsto P_{assigned}(r)\}$ $[\forall s \in SESSIONS \bullet r \in S_R(s)$ $\Rightarrow S_R \setminus \{s \mapsto S_R(s)\} \cup \{s \mapsto (S_R(s) \setminus \{r\})\}]$ $ROLES' = ROLES \setminus \{r\} \triangleright$

The main functions required to establish and maintain the user-role and permission role relations are listed in Table 5.6.

Table 5.6 Definition of user-role and permission-role assignment functions

Relations	Description	Functions
USERS/ ROLES	assignUser (u, r): to assign the user u to the role r, the system updates <i>U_R</i> and <i>U_{assigned}</i>	$\triangleleft u \in USERS; r \in ROLES; (u \mapsto r) \notin U_R$ $U_R' = U_R \cup \{u \mapsto r\}$ $U_{assigned}' = U_{assigned} \setminus \{r \mapsto U_{assigned}(r)\} \cup$ $\{r \mapsto (U_{assigned}(r) \cup \{u\})\} \triangleright$
	deassignUser (u, r): to de-assign the user u from the role r, the system updates, <i>U_R</i> , and <i>U_{assigned}</i>	$\triangleleft u \in USERS; r \in ROLES; (u \mapsto r) \in U_R;$ $\forall s \in SESSIONS \bullet s \in S_R(u) \Rightarrow$ $dropActiveRole(u, r, s)$ $U_R' = U_R \setminus \{u \mapsto r\}$ $U_{assigned}' = U_{assigned} \setminus \{r \mapsto U_{assigned}(r)\} \cup$ $\{r \mapsto (U_{assigned}(r) \setminus \{u\})\} \triangleright$

PRMS/ ROLES	grantPermission (p, r): to grant the permission p to the role r, the system updates P_R and $P_{assigned}$	$\triangleleft p \in PRMS; r \in ROLES;$ $P_R' = P_R \cup \{p \mapsto r\}$ $P_{assigned}' = P_{assigned} \setminus \{r \mapsto P_{assigned}(r)\} \cup$ $\{r \mapsto (U_{assigned}(r) \cup \{p\})\} \triangleright$
	revokePermission (p, r): to revoke the permission p from the role r, the system updates P_R and $P_{assigned}$	$\triangleleft p \in PRMS; r \in ROLES;$ $(p \mapsto r) \in P_R;$ $P_R' = P_R \setminus \{p \mapsto r\}$ $P_{assigned}' = P_{assigned} \setminus \{r \mapsto P_{assigned}(r)\} \cup$ $\{r \mapsto (P_{assigned}(r) \setminus \{p\})\} \triangleright$

5.3.2 Supporting system functions

The supporting system functions are required for session management and in making access control decisions. Each user is associated with a default session which consists of a number of active roles. A user may have been assigned many roles, but only the roles that are in a session can be accessed. The composition of a session can be altered by adding or deleting active roles.

To create a new session with a given user as owner and an active role, the function (Table 5.7) is valid if and only if:

- The user is a member of the USERS, and
- The active role is a role that has been assigned to that user.

The function that adds a role as an active role to a session is valid if and only if:

- The user is a member of the USERS data set,
- The role is a member of the ROLES data set,

- The session is a member of the **SESSIONS** data set,
- The role is assigned to the user, and
- The session is owned by that user.

The schema that describes this function is given in Table 5.7.

To delete a role from the active role set of a session owned by a given user, the function (Table 5.7) is valid if and only if the user is a member of the **USERS**, the session is a member of the **SESSIONS**, the session is owned by the user, and the role is an active role of that session.

Table 5.7 Definition and specification of session related functions

Descriptions	Functions
createSession(u, ar, s): to create the session s for the user u with the role ar, the system updates SESSIONS , U_R , and S_R	$\triangleleft u \in \text{USERS};$ $ar \subseteq \{r : \text{ROLES} \mid (u \mapsto r) \in U_R\};$ $s \notin \text{SESSIONS}$ $\text{SESSIONS}' = \text{SESSIONS} \cup \{s\}$ $U_S' = U_S \setminus \{u \mapsto U_S(u)\} \cup \{u \mapsto (U_S(u) \cup \{s\})\}$ $S_R' = S_R \cup \{s \mapsto ar\} \triangleright$
addActiveRole(u, r, s): to add the role r as the active role to the session s of the user u, the system updates S_R	$\triangleleft u \in \text{USERS}; s \in \text{SESSIONS};$ $r \in \text{ROLES}; s \in U_S(u)$ $(u \mapsto r) \in U_R; r \notin S_R(s)$ $S_R' = S_R \setminus \{s \mapsto S_R(s)\} \cup \{s \mapsto (S_R(s) \cup \{r\})\} \triangleright$
dropActiveRole(u, r, s): to delete the active role r from the session s of the user u, the system updates S_R	$\triangleleft u \in \text{USERS}; s \in \text{SESSIONS};$ $r \in \text{ROLES}; s \in U_S(u)$ $(u \mapsto r) \in U_R; r \in S_R(s)$ $S_R' = S_R \setminus \{s \mapsto S_R(s)\} \cup \{s \mapsto (S_R(s) \setminus \{r\})\} \triangleright$

5.3.3 Review functions

After user-to-role assignment and permission-to-role assignment have been made, the system should be able to display the contents of these assignment relations from all the three perspectives: role, user, and permission. For example, the administrators should be able to view all the users that have been assigned to a given role, all the permissions that have been assigned to a given role, all the roles assigned to a given user, and all the roles assigned to a given permission. The Table 5.8 below lists these review functions.

Table 5.8 Definition and specification of system review functions

Descriptions	Functions
assignedUsers (r, result): to view the users that have been assigned to the role r	$\triangleleft r : ROLES$ $result = \{u : USERS \mid (u \mapsto r) \in U_R\} \triangleright$
assignedUserRoles (u, result): to view the roles that have been assigned to the user u	$\triangleleft u \in USERS;$ $result = \{r : ROLES \mid (u \mapsto r) \in U_R\} \triangleright$
assignedPermissions (r, result): to view the permissions that have been granted to the given role r	$\triangleleft r : ROLES$ $result = \{p : PRMS \mid (p \mapsto r) \in P_R\} \triangleright$
assignedPrmnRoles (p, result): to view the set of roles that are associated with the given permission p	$\triangleleft p \in PRMS$ $result = \{r : ROLES \mid (p \mapsto r) \in P_R\} \triangleright$

5.4 Security and privacy

5.4.1 Security

It is often heard that most security threats are from inside intruders. In order to enhance the system security, this e-education system comes up with a design with three levels of system administration. The entry level administrators are only allowed to perform the basic duties. The middle level administrators have more privileges than the entry level; they have more responsibilities. The top level has all the privileges to manage the system. To avoid security failures from monopoly access to the system resources, two individual administrators must be present at the same time to perform their top privileged duties.

The duties for each level are summarized in Table 5.9.

Table 5.9 Three levels of administrators and their duties

LEVEL	DUTY
Level one	<ol style="list-style-type: none"> 1. confirm a new user's information 2. delete user from the system
Level two	<ol style="list-style-type: none"> 1. Have all duties of level one 2. create/delete new permissions and objects 3. assign/design a role to a user, but not activate them
Level three	<ol style="list-style-type: none"> 1. have all duties of level one and two 2. set up conflict role 3. set up relationship for inheritance 4. delete a role or add a role to the system 5. change permissions for a role <p>(To do the duty, two administrators login in together at same time)</p>

5.4.2 Privacy

In order to enforce users' privacy, a set of logon username and password is issued to each user. Username is set as a string of one letter combining seven digits. A username starts with one letter then seven digits. It is hard to guess a user's real name and role based on the username - a string of letter and digits. As a result, users' privacy is protected. Compared to the traditional way in which username is set as a user's first name initial plus last name, this way of setting up username gives users more privacy.

A password is randomly generated as a string mixed with ten digits and 26 letters. Like Social Insurance Number or student ID number, it may not be easy to remember at the beginning, but after a while, people will get used to it.

Privacy concerns are also considered when role inheritance is designed. It is reasonable and understandable that every role has something that should be kept private. For instance, the privilege that students view their own academic grades should not be inherited by TA role. Therefore, in this study each role's permissions are divided into two groups. One group can be inherited, but the other group should not be inherited, which only belongs to this role itself.

Having a private attribute for each role's permissions is a unique design of this study. It improves privacy and reduces the number of conflict of interest roles. And it is demonstrated in the following chapter that this design is effective.

Chapter 6

Implementation

The implementation is carried out using software and connectivity drivers that are free of cost and widely distributed from Internet. They are compatible to each other and suitable for client/server environment deployment. The computer programming languages chosen are the most popular and available ones. The use of these implementation tools in this study will allow other researchers to repeat and compare their study with the results of this thesis.

6.1 Implementation Environment Overview

In order to implement the RBAC based e-education system, this study used a three-tier client server architecture environment (Figure 6.1). The three tiers are client, web server, and database server.

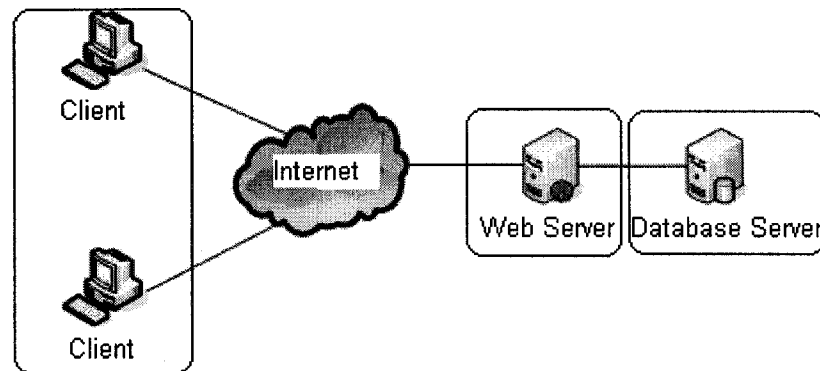


Figure 6.1 Three tier architecture of e-education system

The client tier is individual Internet browsers. It allows users to access online education information, view assignments, take online exams, submit answers, and display feedback information from the education system, as well as allows administrators to maintain and update the system resources.

The database server is a MySQL database. It contains the e-education system information including *users*, *roles*, *permissions*, *objects*, *operations*, *sessions*, *hierarchical relations*, and *conflict role relations*. For example, it maintains users' personal data, academic records, payment statements, as well as usernames and passwords. All the resources are stored as relational database tables.

The web server serves as the middle tier; it is an Orion server (Figure 6.2). It hosts Java Servlet and JDBC driver. The function of the Java Servlet is to process client requests and server feedbacks. The Servlet takes client requests, processes the requests, passes the processed requests over to the server tier, and then takes server's feedbacks and sends back to the client.

All Java Servlet files are held and maintained in the directory *Orion/default-web-app/WEB-INF/classes* of the Orion server. All HTML files are also stored and maintained in the Orion server, but in the directory *Orion/default-web-app*.

In the three tiers communications, the connections between the client tier (web browsers) and the middle tier (the web server) are made by the Internet, and the connections between the middle tier and database server tier (mySQL) are made by the JDBC driver.

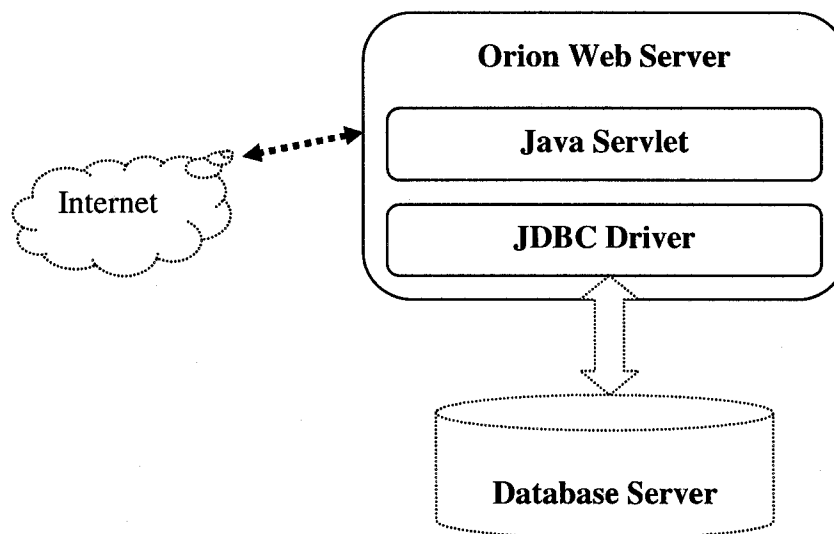


Figure 6.2 Middle tier - Orion server

The rationale for using MySQL is that MySQL is the most popular open source SQL database management system [MySQL Manual, 2003]. Open source means that anyone can use and modify the software, can download the MySQL

software from the Internet and use it without paying anything. Anyone may study the sources code and change it to suit their needs. The MySQL database server is fast, reliable, and easy to use. It offers a variety of useful functions. Its connectivity, speed, and security make it highly suitable for accessing databases on the Internet. It works well in client/server or embedded systems.

The rationale for using an Orion server is that the Orion server is its own web server, and its use eliminates the hand-off that usually occurs between the application server and a web server like Apache. Another reason is that it is free to download, with well-documented expert instructions for installation.

Because these two application systems (MySQL and Orion) are widely distributed and easily available, other researchers can easily implement one approach using them. This study can be repeated and verified by other researchers using the same techniques.

The implementation of the RBAC based e-education system is essentially a client-server application development. The initial implementation phase includes the following tasks:

- Downloading, installing, and configuring the Orion server
- Downloading and installing the MySQL database
- Installing and configuring the JDBC (server software tools)

Both Orion server and MySQL are downloaded from the Internet, and attached to the Computer Science server of Saint Mary's University. After the client-server environment is properly setup, the implementation starts. The following sections describe the detailed implementations of the RBAC based e-education system.

The implementation is done in Java computer programming language, Java Database Connectivity (JDBC), Java Server Development Kit (JSDK), HTML, Java Script, and MySQL database. Coding and testing are conducted in the UNIX operating system environment.

In order to make the system a thin client side application, HTML is used as much as possible. Java Script codes are added when they are needed. For example, Java Script is used for client side intelligence such as the timing device and the auto-commit controls of a time-limited test and assignment (see section 6.2.2 for details).

The prototype implementation of the e-education system contains about 3000 lines of Java source codes. Among them, about 2000 lines are for data communications between tiers; about 1000 lines are for the user interfaces.

The implemented e-education system is virtually platform independent. It can be compiled and run on any platform that supports Java and JDBC.

6.2 Administrative Interfaces

As discussed in Chapter 5, three levels of administrator roles are designed and implemented. They are shown in Figures 6.3 to 6.5. As seen from Figure 6.3, the level one (entry level) has the following functions: confirming new users' registration, and user management. As seen from Figure 6.4, in addition to the entry level duties, the middle level administrator role has its own duties which are permission management, and user/role assignment, and permission/role assignments. The top level administrator role inherits all the duties from the level one and level two, plus its own unique duty - role management; administrators of this level can perform all the duties that are relevant to system management (confirming new users' registration, user management, permission management, role management, and user/role assignment and permission/role assignments) (Figure 6.5). Table 5.7 in Chapter 5 lists detailed descriptions of these functions.

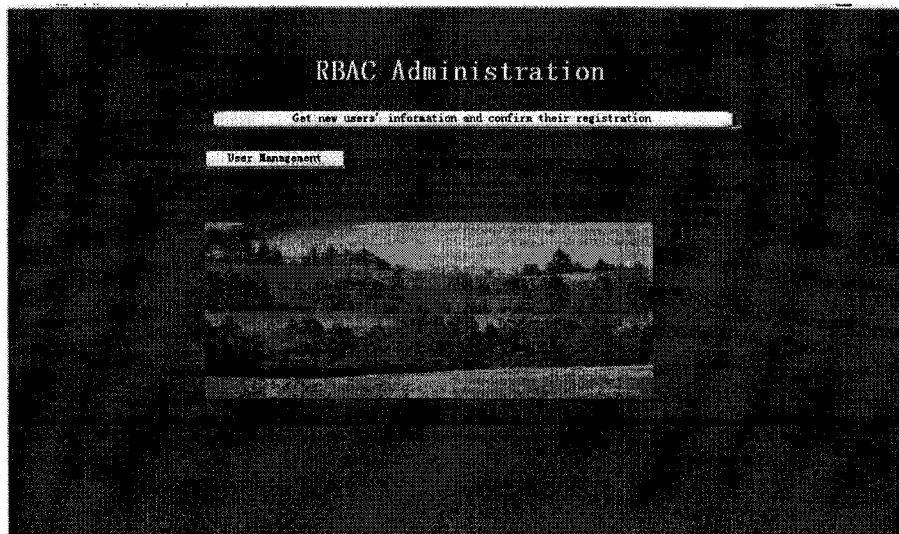


Figure 6.3 Interface for Level I administrators

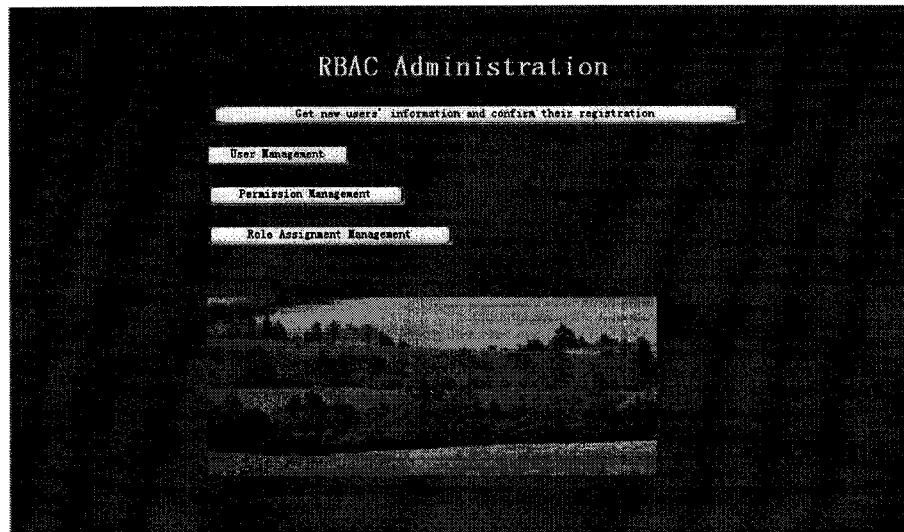


Figure 6.4 Interface for Level II administrators

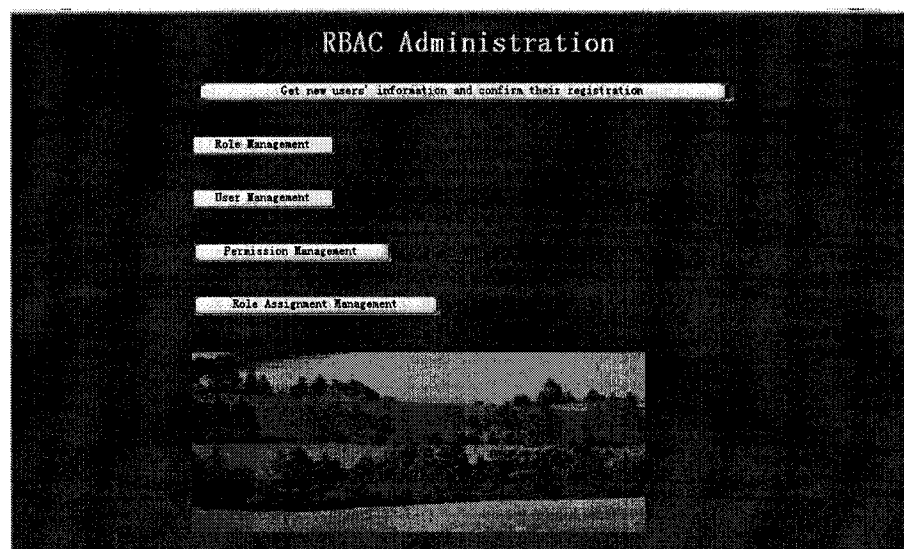


Figure 6.5 Interface for Level III administrators

As seen from Figure 6.5, the level three administrators' interface, the overall system administrative duties are divided into five groups: verification of new users' registration, role management, user management, permission management, and role/user and role/permission assignment management. The

five duties are represented and accessed by five buttons on the interface. The following are discussions of the five groups of interface implementation.

1) To confirm new users' registration. Before administrators confirm new users' registration, all the registration information (name, address, phone, payment, etc.) is stored in the *newUsers* table (see section 6.3.1 for more discussion on new users' registration). Figure 6.6 shows the steps for administrators to perform this duty. First, access the *newUsers* table (Table 6.1) in the database, and get the registration information of all the recently registered users. In the second step, based on the returned data, create username and password for each new user, and store the usernames and passwords in the *users* table (Table 6.2); copy new users' registration data to the *allUsers* table (Table 6.3); empty the *newUsers* table for future users registration. Then, send a copy of the newly created username and password to the corresponding users.

Figure 6.6 also shows the communications among the three tiers. The highlighted users, servlet, and database tables represent the parts that are invoked in the corresponding processes. In this duty performance, *administrators* users initiate the process. The work is done by the servlet *acceptUsers*. The table *allUsers*, *newUsers*, and *users* are passively involved in the processes.

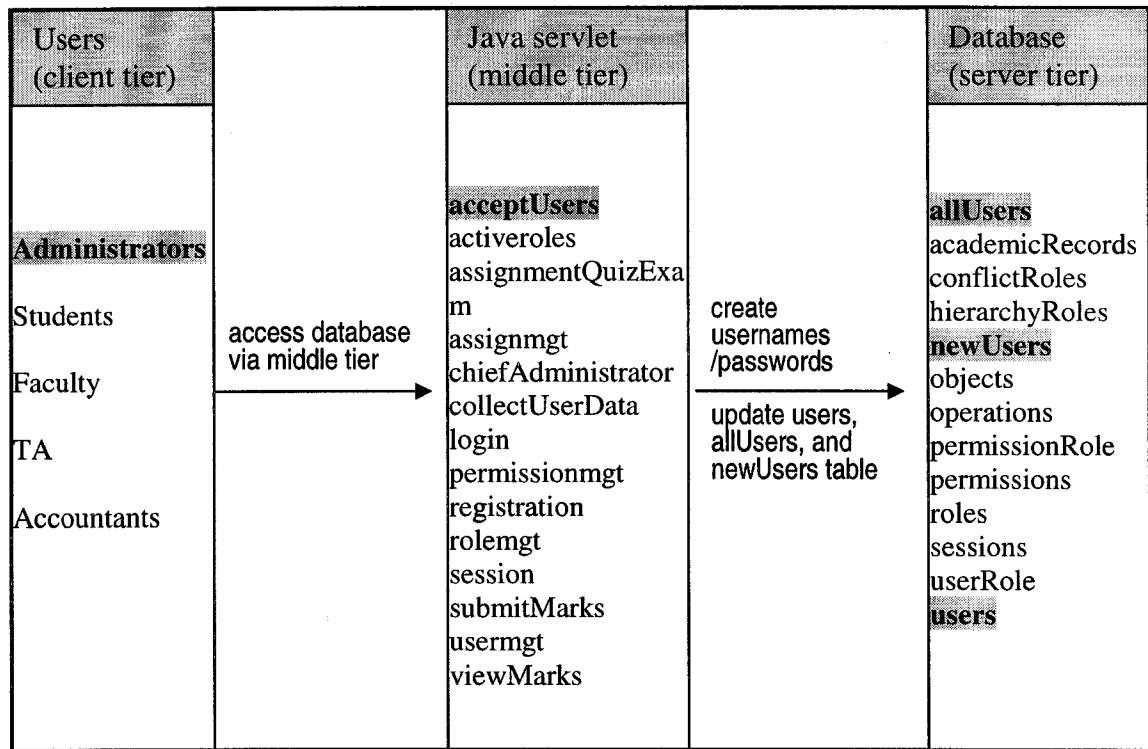


Figure 6.6 Processes of confirming new users' registrations

Table 6.1 Registration information for the new users

lastname	firstname	SIN	rolename	address	Phone	Email	payment
...
Wendy	User1	123456789	TA	26 Tower Road	(902)4968152	uw@yahoo.ca	\$200
...

Table 6.2 Users' table containing all the users' username, password, and SIN

Username	Password	SIN
...
e2316986	Xxxxxxx	123456789
...

Table 6.3 Data for all registered users

lastname	firstname	SIN	rolename	address	Phone	email	payment
...
Wendy	User1	123456789	TA	26 Tower Road	(902)4968152	uw@yahoo.ca	\$200
...

The key codes for the above process are given below.

a) to connect to the *newUsers* table.

...

```
try {Class.forName("com.mysql.jdbc.Driver").newInstance();}
catch (Exception e) {toClient.println("Failed to load JDBC/ODBC driver.");}

try {Connection con = DriverManager.getConnection("jdbc:mysql://localhost/h_zhao", "h_zhao",
"nRy2xN");

        Statement st = con.createStatement();

        st.executeUpdate("use h_zhao;");

        ResultSet result = st.executeQuery("select * from newUsers;");

        int fCol = result.findColumn ("lastname");
        int lCol = result.findColumn ("firstname");
        int sinCol = result.findColumn ("SiN");
        int rCol = result.findColumn ("rolename");
        int adCol = result.findColumn ("address");
        int phCol = result.findColumn ("phone");
        int emCol = result.findColumn ("email");
        int paCol = result.findColumn ("payment");
```

b) to create username and password

...

```
String y = "1234567890"; //for username
```

```
String x = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890"; //for
password

String username = "e";

String pw = "";

    for ( int j=0; j<7; j++ ) {

        username = username + String.valueOf( y.charAt((int)( Math.random() * y.length() )) ) );

    }

    for ( int i=1; i<7; i++ ) {

        pw = pw + String.valueOf( x.charAt((int)( Math.random() * x.length() )) );

    }

```

c) to return the username and password to corresponding users

...

```
toClient.println("Your username is: " + username + ", password is: " + pw);
```

d) to update the *users*, *userRole*, *sessions*, *newUsers*, and *allUsers* tables

...

```
st.executeUpdate("insert into users values('"+username+"','"+pw+"');");
st.executeUpdate("insert into userRole values('"+username+"','"+rnm+"');");
st.executeUpdate("insert into sessions values('"+username+"','S1','"+rnm+"');");
st.executeUpdate("insert into allUsers
values('"+lnm+"','"+fnm+"','"+sin+"','"+rnm+"','"+ad+"','"+ph+"','"+em+"','"+pa+"');");
st.executeUpdate("delete from newUsers;");
```

2) To perform the duty - role management. In this duty performance, the following interface (Figure 6.7) is used, where administrators can add/remove roles to/from the system, assign/remove conflict role relations, and setup/remove hierarchical relations between roles.

The screenshot displays the 'ROLE MANAGEMENT' interface with the following sections:

- Enter a new role:** A text input field followed by an 'Add a Role' button.
- Roles:** A dropdown menu showing 'assistant administrator' and a 'Remove the selected role' button.
- Roles:** A dropdown menu showing 'account manager'.
- Roles:** A dropdown menu showing 'account manager'.
- Conflict Role Pairs in the RBAC System:** A section with a dropdown showing 'student <--> account manager', an 'Add a Conflict Role pair' button, and a 'Remove the selected conflict role pair' button.
- Role Hierarchical relationship in the RBAC System:** A section with a dropdown showing 'account manager --> basic user', an 'Add a hierarchy relation' button, and a 'Remove the selected hierarchy relationship' button.

Figure 6.7 Role management interface

Administrators can enter a role name in the text box, and add it into the system by pressing the *Add a Role* button; they can also select a role from the list, and remove it from the system by pressing the button *Remove the selected role*.

In this process (Figure 6.8) the *rolemgmt* servlet is invoked. When adding a role to the system, the *roles* table (Table 6.4) is updated. When removing a role from the system, the tables, *roles*, *sessions* (Table 6.5), *conflictRoles* (Table 6.6), *hierarchyRoles* (Table 6.7), and *permissionRole* (Table 6.8), are all updated. This servlet (*rolemgmt*) will go to these tables and check if any record is related to

this role. Any entry and/or relation maintained in these tables, which is related to this role, will be confirmed and removed from these tables.

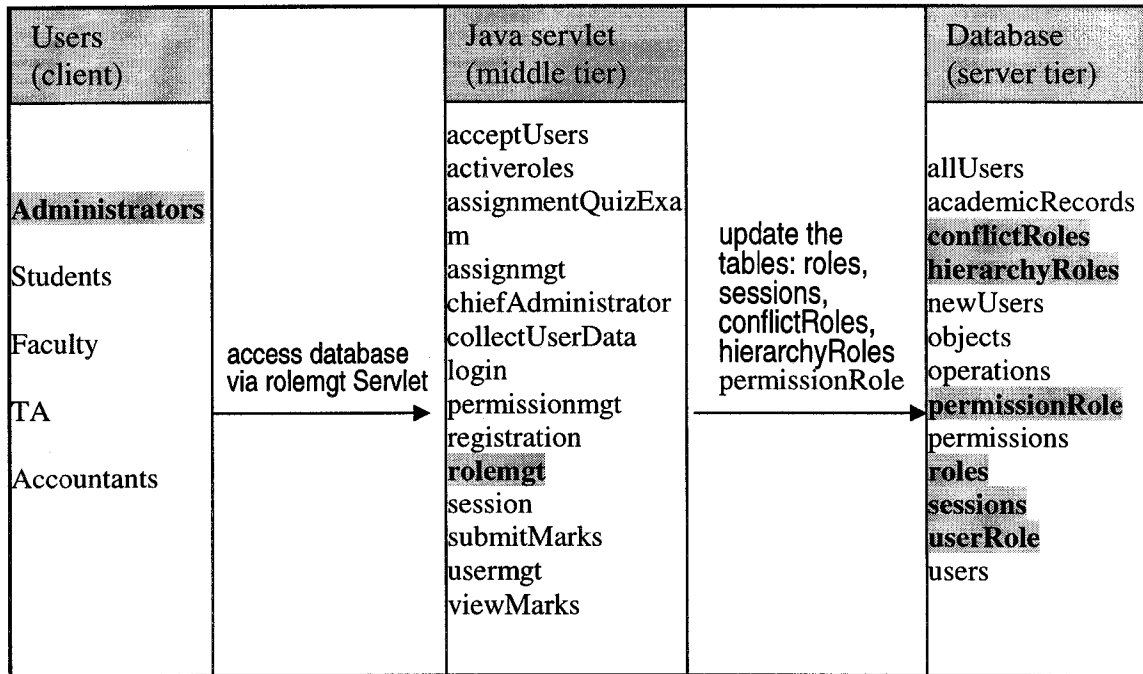


Figure 6.8 Processes of Role Management

Table 6.4 The *roles* table containing all roles of the system

rolename
account manager
administrator
assistant administrator
basic user
chief administrator
faculty
student
TA

Table 6.5 The *sessions* table

username	sname	activeroles
...
e2651855	SI	TA
...

Table 6.6 The *conflictRoles* table

rolename	itsconflictroles
...	...
student	Account manager
...	...

Table 6.7 The *hierarchyRoles* table

parent	child
...	...
faculty	TA
TA	Student
...	...

Table 6.8 The *permissionRoles* table

permissionname	rolename	Privacy
...
Read → studentMarks	TA	No
...

To assign/remove conflict role relations, administrators select two roles from the two separate role lists, and press the appropriate buttons; the *conflictRoles* table will be updated (Figure 6.8). In a similar way, administrators can establish a hierarchical relation by selecting two roles from the two separate role lists and pressing the button *Add a Conflict Role Pair*, or remove a relation by selecting a relation pair from the list and pressing the *remove* button. In this process the *hierarchyRoles* table will be updated (Figure 6.8).

The key codes for this duty performance are shown below.

a) to add a new role to the system

...

```
try {st.executeUpdate("insert into roles values
('"+(req.getParameter("newrole")).trim()+"");"}
catch (SQLException sqle){
    toClient.println("<font color = 'red'>");
    toClient.println("duplicate entry, the role already exists");
    toClient.println("</font>");
}
```

b) to remove a role from the system

...

```
else if (req.getParameter( "debtn" ) != null){
    st.executeUpdate("delete from roles where
rolename='"+(req.getParameter("rolelist")).trim()+"';");

    st.executeUpdate("delete from userRole where
rolename='"+(req.getParameter("rolelist")).trim()+"';");

    st.executeUpdate("delete from permissionRole where
rolename='"+(req.getParameter("rolelist")).trim()+"';");

    st.executeUpdate("delete from conflictRoles where
rolename='"+(req.getParameter("rolelist")).trim()+"' or
itsconflictrole='"+(req.getParameter("rolelist")).trim()+"';");
```

```

        st.executeUpdate("delete from hierarchyRoles where
        parent='"+(req.getParameter("rolelist")).trim()+"' or
        child='"+(req.getParameter("rolelist")).trim()+"';");

        st.executeUpdate("delete from sessions where
        activeroles='"+(req.getParameter("rolelist")).trim()+"';");
    }

```

c) to add/delete conflict roles

...

```

else if (req.getParameter( "addbtnC" ) != null){
    st.executeUpdate("insert into conflictRoles values
    ('"+(req.getParameter("rolelist1")).trim()+"', '"+
    (req.getParameter("rolelist2")).trim()+"');");
}

else if (req.getParameter( "debtnC" ) != null){
    String thePair = (req.getParameter("crolelist")).trim();
    int sp = thePair.indexOf("<");
    String r1 = thePair.substring(0,sp-1);
    String r2 = thePair.substring(sp+6,thePair.length());

    st.executeUpdate("delete from conflictRoles where rolename='"+r1+"' and
    itsconflictrole = '"+r2+"';");
}

```


d) to add/remove hierarchy roles

...

```

else if (req.getParameter( "addbtnH" ) != null){

    st.executeUpdate("insert into hierarchyRoles values

    ("+(req.getParameter("rolelist1")).trim()+"", "+
    (req.getParameter("rolelist2")).trim()+"");

}

else if (req.getParameter( "debtnH" ) != null){

    String thePair = (req.getParameter("hrolelist")).trim();
    int sp = thePair.indexOf("-");
    String r1 = thePair.substring(0,sp-1);
    String r2 = thePair.substring(sp+5,thePair.length());

    st.executeUpdate("delete from hierarchyRoles where parent='"+r1+"' and
    child = '"+r2+"";

}

```

3) To perform the duty - user management, the interface (Figure 6.9) is used, where administrators can remove users from the system. In the interface all the current users are listed; administrators choose a user from the select list and press the remove button, then the user will be removed.

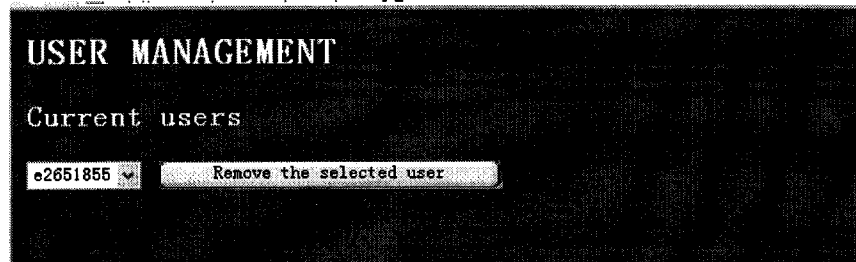


Figure 6.9 Interface for User Management duty

In this process (Figure 6.10), the *usermgt* servlet is invoked, and the tables, *users* (Table 6.2), *userRole* (Table 6.9), *sessions* (Table 6.5), and *academicRecords* (Table 6.10), are updated.

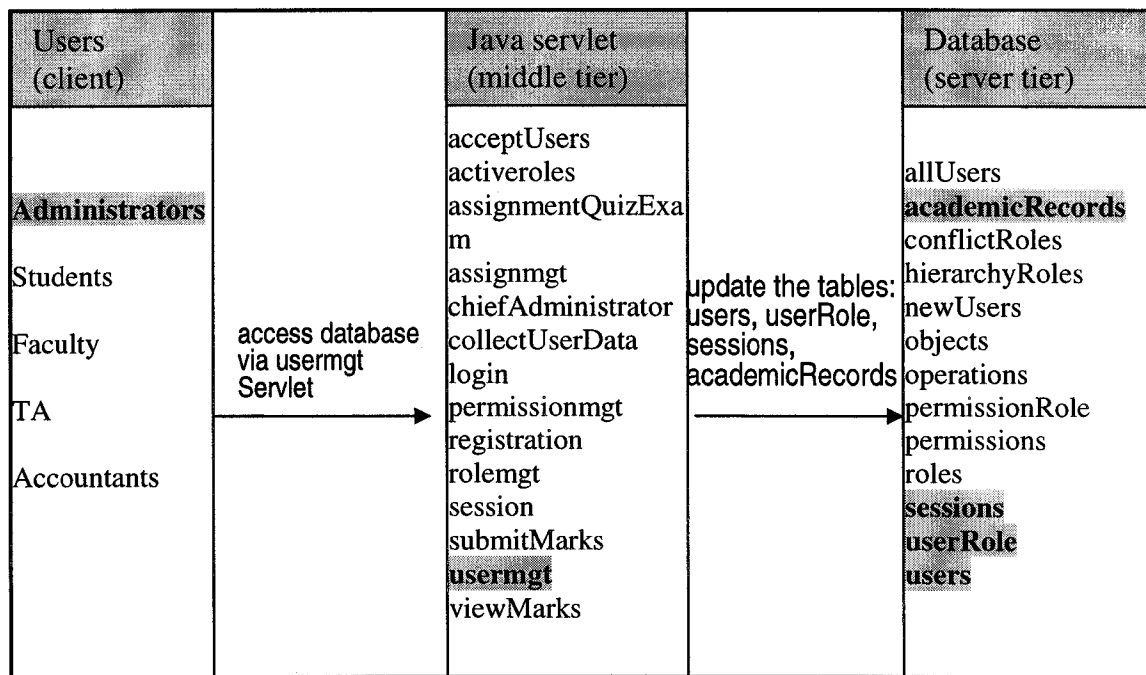


Figure 6.10 Processes of User Management

Table 6.9 The *userRole* table

username	rolename
...	...
e2651855	TA
...	...

Table 6.10 The *academicRecords* table

studentID	instructorname	coursename	term	year	aqe	ans	marks	comments
...
e3275344	Jen	Commerce	Spring	2004	Quiz1	RBAC stands for role based access control	100	Very good
...

The key codes are given below. They are to update the tables where the users related information are stored.

```

if (req.getParameter( "debtn" ) != null){

    st.executeUpdate("delete from users where username="+req.getParameter( "userlist"
    )).trim()+"");

    st.executeUpdate("delete from userRole where username="+req.getParameter( "userlist"
    )).trim()+"");

    st.executeUpdate("delete from sessions where username="+req.getParameter( "userlist"
    )).trim()+"");

    st.executeUpdate("delete from academicRecords where studentID="+req.getParameter(
    "userlist" )).trim()+"");

```

}

4) To perform the duty - Permission Management, the interface (Figure 6.11) is used. Administrators enter an object or operation, press the Add buttons to create them; they can select an object or operation to delete them from the system. When creating a permission, administrators select one object and one operation from the two individual lists, and press *Make ...* button. When removing a permission from the system, administrators must select a permission from the list, and press the *Remove ...* button.

The screenshot displays a web-based interface titled "PERMISSION MANAGEMENT". It is organized into several sections:

- Enter a new operation name:** A text input field followed by an "Add an Operation" button.
- Operations:** A dropdown menu with a downward arrow and a "Remove the selected operation" button.
- Enter a new object name:** A text input field followed by an "Add an Object" button.
- Objects:** A dropdown menu showing "administratorDuties" with a downward arrow, and a "Remove the selected object" button.
- Create a new permission:** A section containing two dropdown menus. The first is empty, and the second shows "administratorDuties". Below these is a "Make the selected operation - object pair a permission" button.
- read - administratorDuties:** A dropdown menu showing this selected pair, with a "Remove the selected permission" button next to it.

Figure 6.11 Permission Management Interface

In this process (Figure 6.12) the permissionmgt servlet is invoked; the four tables, permissions (Table 6.11), permissionRole (Table 6.8), objects (Table 6.12), and operations (Table 6.13), are updated.

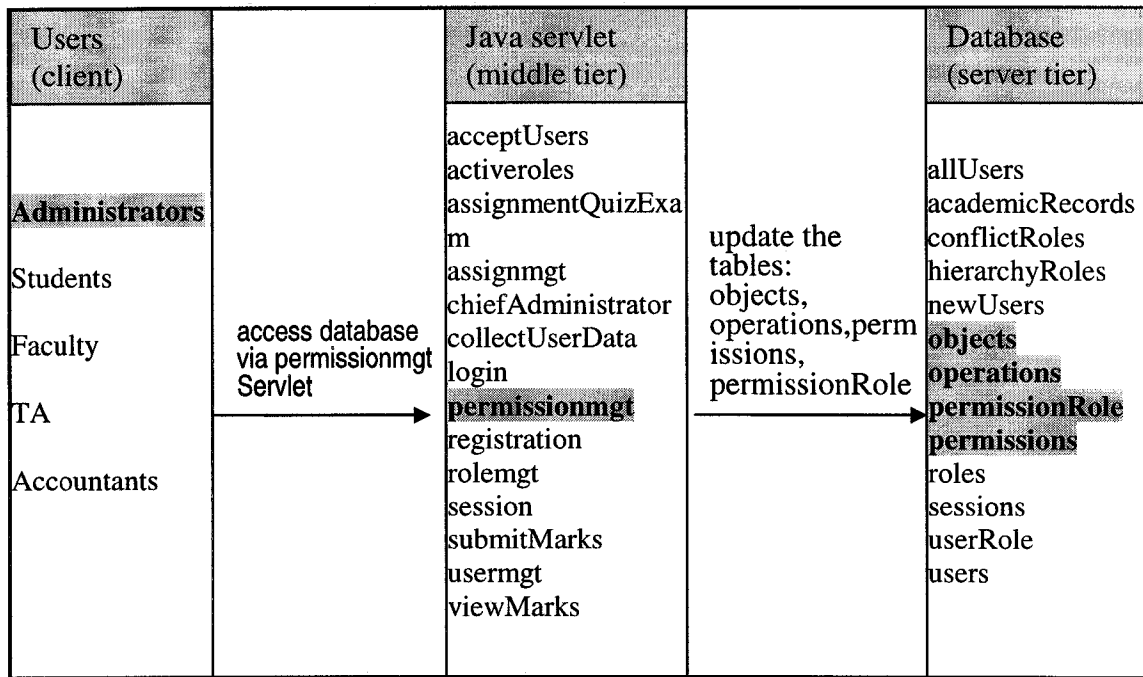


Figure 6.12 Processes of Permission Management

Table 6.11 The *permissions* table

operation	object
...	...
read	grades
...	...

Table 6.12 The *objects* table

objectname
...
chiefAdministratorDuties
...

Table 6.13 The *operations* table

operationname
...
Read
...

Following are the key codes that are used to perform the above duties.

a) to add an operation

```
if (req.getParameter( "addbtnOP" ) != null){
    try {st.executeUpdate("insert into operations values
        ("'+(req.getParameter("newoperation")).trim()+"'");"}
    catch (SQLException sqle){
        toClient.println("<font color = 'red'>");
        toClient.println("duplicate entry, the operation already exists");
        toClient.println("</font>");
    }
}
```

b) to delete an operation

```
else if (req.getParameter( "debtnOP" ) != null){
    st.executeUpdate("delete from operations where
        operationname='"+(req.getParameter("operationlist")).trim()+"'");

    st.executeUpdate("delete from permissions where
        operation='"+(req.getParameter("operationlist")).trim()+"'");
}
```

c) to add an object

```

else if (req.getParameter( "addbtnOBJ" ) != null){
    try {st.executeUpdate("insert into objects values
        ("+(req.getParameter("newobject")).trim()+"");"}
    catch (SQLException sqle){
        toClient.println("<font color = \"red\">");
        toClient.println("duplicate entry, the objects already exists");
        toClient.println("</font>");
    }
}

```

d) to delete an object

```

else if (req.getParameter( "debtnOBJ" ) != null){
    st.executeUpdate("delete from objects where
        objectname="+(req.getParameter("objectlist")).trim()+"");

    st.executeUpdate("delete from permissions where
        object="+(req.getParameter("objectlist")).trim()+"");
}

```

e) to add a permission

```

else if (req.getParameter( "addbtnPERM" ) != null){
    String op = (req.getParameter("operationlist1")).trim();
    String obj = (req.getParameter("objectlist1")).trim();
}

```

```

        try {st.executeUpdate("insert into permissions values
        ('"+op+"','"+obj+"');");}
        catch (SQLException sqle){
            toClient.println("<font color = 'red'\>");
            toClient.println("duplicate entry, the permission already exists");
            toClient.println("</font>");
        }
    }
}

```

f) to delete a permission

```

else if (req.getParameter( "debtnPERM" ) != null){
    String thePair = (req.getParameter("permissionlist")).trim();
    int sp = thePair.indexOf("-");
    String op = thePair.substring(0,sp-1);
    String ob = thePair.substring(sp+3,thePair.length());

    st.executeUpdate("delete from permissions where operation='"+op+"' and
    object = '"+ob+"';");
}

```

5) To perform the duty - Role/User/Permission Assignment Management includes: assign/remove a user to/from a role, put a role into a session and remove a role from a session, and assign/remove a permission to/from a role (Figure 6.13). The duty also includes: display the users that a role is associated, and display the roles that a user has.

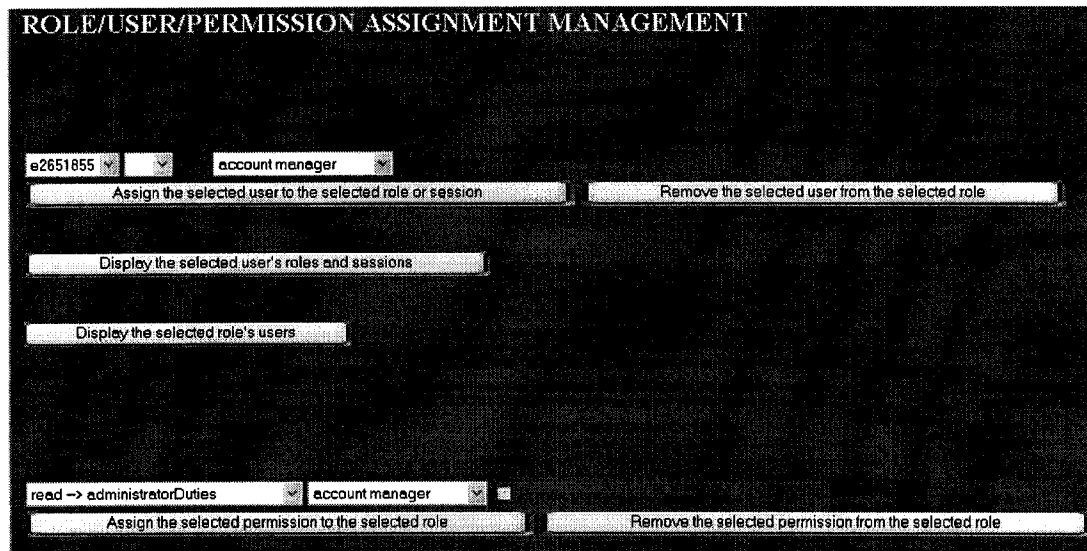


Figure 6.13 Interface for the role/user/permission assignment management

In this process the *assignmgt* servlet is used, and the *sessions*, *userRole*, and *permissionRole* tables are involved (Figure 14). All the current users, roles and permissions are listed on the interface (Figure 6.13).

Administrators make a user-role assignment relation by selecting a role and a user, leaving the *sessions* list blank, and pressing the button *Assign ...* At the same time the *userRole* table (Table 6.9) is updated. When removing a user-role assignment relation, administrators will select a role and a user, leaving the *sessions* list blank, and press the *Remove ...* button. In this deassign process both the *userRole* table and *sessions* table (Table 6.5) are updated. When deciding to make a user's roles become active or de-active, administrators can repeat the above steps, plus choose a session (S1, S2 or S3) from the *sessions* select list. When doing this, only the *sessions* table is involved and updated; the *userRole* table remains intact.

When performing the permission/role assignment, administrators select a permission and a role, and press the *Assign ...* button. In role hierarchy relations it is determined that certain permissions of a role are not allowed to be inherited, that is, certain permissions are only designated for certain roles. Therefore, when doing the permission-role assignment, users must indicate whether or not this permission is private for the role by ticking the *Private permission* box. When removing such an assignment relation, users can simply select the role and permission, and press the *Remove ...* button. The *permissionRole* table (Table 6.8) will be updated accordingly.

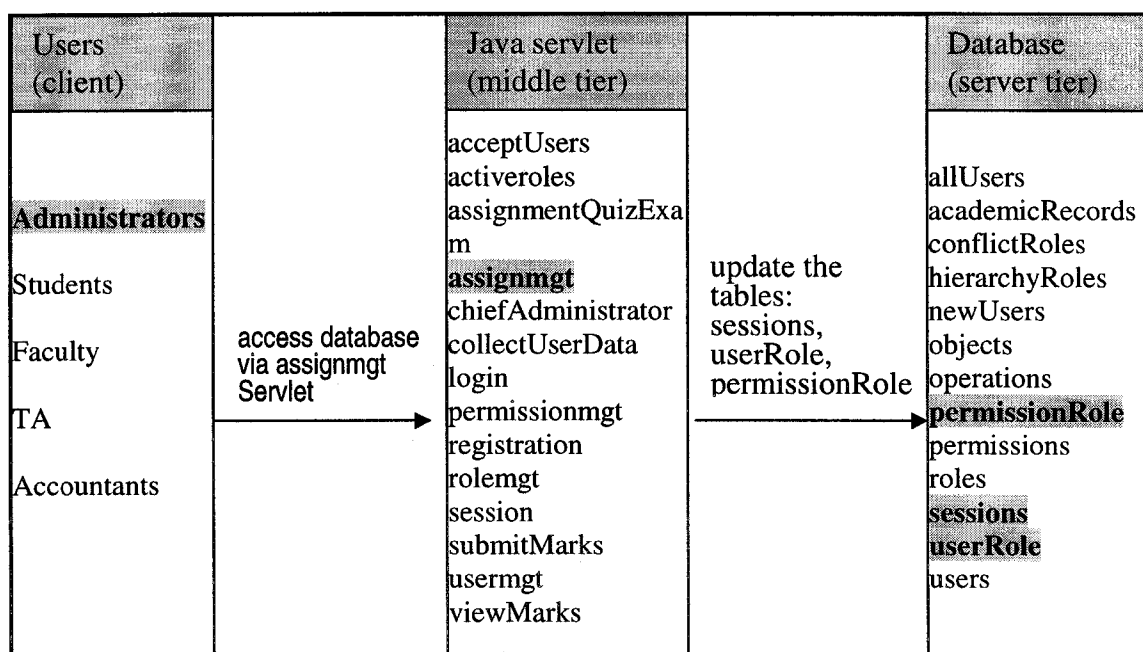


Figure 6.14 Processes of Role/User/Permission Management

To show how many roles and sessions including their active roles that a user has, or show how many users a role is associated with, administrators can press the

Display ... roles and session button, or the *Display ... users* button. For example, the Figure 6.15 shows the user e2651855 has two roles: account manager and TA, and one session which has an active role of TA.

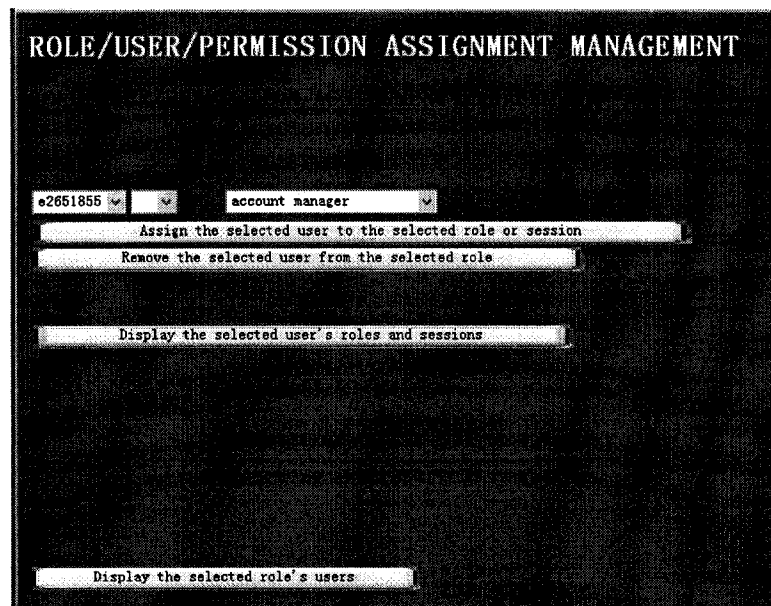


Figure 6.15 Showing the roles and sessions that the user e2651855 has

In the user/role/permission assignment management interface (Figure 6.13), three sessions are available. Normally one user can have many sessions. It is up to the organization that uses the RBAC concept to make a decision on the number of sessions for each user. The reasons that this study implements three sessions are: first, this study is a test of the RBAC concept and models, three sessions are enough to fulfill the purpose of this study; second, if a user has more than three sessions, that means the user has more than three roles that are in conflict of interests, if this is the case, the organization needs to consider the

creation of a separate role, rather than maintain a number of sessions for one user.

For each user, its initial role will be put into the default session - the session one. Any other role that has been assigned to this user and has no conflict of interests with the role in the session one can be put in this session. If the role is in conflict of interests with any role in the session one, then this role must be put into a new session.

The key codes for this duty performance are:

a) to create a user/role relation

```
if (req.getParameter( "addbtnUserRole" ) != null){
    String u = (req.getParameter("userlist")).trim();
    String s = (req.getParameter("sessionlist")).trim();
    String r = (req.getParameter("rolelist")).trim();

    if(s.compareTo("")==0) { // this is simply a user and role assignment
        try {st.executeUpdate("insert into userRole values ('"+u+"','"+r+"');");}
        catch (SQLException sqle){
            toClient.println("<font color = \"red\">");
            toClient.println("duplicate entry, user-role assignment already exists");
            toClient.println("</font>");
        }
    }
}
```

b) to put a role into a session

```

else { // this is a session creation, first have to check if the user has such a role,

    // if not, can not assign the role to the session

    boolean theUserHasTheRole = false;

    ResultSet sresult = st.executeQuery("select * from userRole;");

    int Colu = sresult.findColumn ("username");

    int Colr = sresult.findColumn ("rolename");

    String unnm="";

    String rnm="";

    while(sresult.next()) {

        unnm = sresult.getString(Colu);

        rnm = sresult.getString(Colr);

        if (unnm.compareTo(u) == 0 && rnm.compareTo(r) == 0) {

            theUserHasTheRole = true;

            break;

        }

    }

    if (theUserHasTheRole == true) {

        try {

            st.executeUpdate("insert into sessions values ('"+u+"','"+s+"','"+r+"');");

        }

        catch (SQLException sqle){

            toClient.println("<font color = \"red\">");

```

```

        toClient.println("duplicate entry, this session-role already exists");

        toClient.println("</font>");
    }
}

else

    toClient.println("The user does not have such a role! Please assign the
    role to the user, then you can add this role to a session!");

}

```

c) to remove a user/role relation

```

else if (req.getParameter( "debtnUserRole" ) != null){

    String u = (req.getParameter("userlist")).trim();

    String s = (req.getParameter("sessionlist")).trim();

    String r = (req.getParameter("rolelist")).trim();

    if (s.compareTo("")==0) { // this is to remove the user from the role

        st.executeUpdate("delete from userRole where username='"+u+"' and
        rolename='"+r+"',");

        st.executeUpdate("delete from sessions where username='"+u+"' and
        activeroles='"+r+"',");

    }

    else // just remove the role from the user's this session

        st.executeUpdate("delete from sessions where username='"+u+"' and
        sname='"+s+"' and $

}

```

d) to add a permission/role relation

```

else if (req.getParameter( "addbtnPermRole" ) != null){

    String p = (req.getParameter("permlist")).trim();

    String r = (req.getParameter("prolelist")).trim();

    String ck = req.getParameter("yesnocheck"); // important! don't trim THIS ONE


    String privacy = "no";


    if ( ck != null)

        privacy = "yes";


    try { st.executeUpdate("insert into permissionRole values ('"+p+"','"+r+"','"+privacy+"');")$
    catch (SQLException sqle){

        toClient.println("<font color = \"red\">");

        toClient.println("duplicate entry, this permission-role assignment already exists");

        toClient.println("</font>");

    }

}

```

e) to remove a permission/role relation

```

else if (req.getParameter( "debtbtnPermRole" ) != null){

    String p = (req.getParameter("permlist")).trim();

    String r = (req.getParameter("prolelist")).trim();

    st.executeUpdate("delete from permissionRole where permissionname='"+p+"' and
    rolename='"+r+"';");

}

```

f) to display a user's sessions and roles

```

else if (req.getParameter( "showbtnUserRole" ) != null){

    String ur = (req.getParameter("userlist")).trim();

    // add the user name into the first spot of the vector
    userRole_vector.addElement(ur);

    userRole_vector.addElement("roles");

    ResultSet uresult = st.executeQuery("select * from userRole;");

    int Colu = uresult.findColumn ("username");
    int Colr = uresult.findColumn ("rolename");

    String nmu="";
    String nmr="";

    while(uresult.next()) {

        nmu = uresult.getString(Colu);
        nmr = uresult.getString(Colr);

        if (nmu.compareTo(ur) == 0) {

            userRole_vector.addElement(nmr);

        }

    }

    // show the user's sessions and roles
    sessionRole_vector.addElement(ur);

    sessionRole_vector.addElement("sessions/roles");

    sessionRole_vector1.addElement(ur);

    ...

}

```


g) to display a user's sessions and roles

```

else if (req.getParameter( "showbtnRoleUser" ) != null){

    String rl = (req.getParameter("rolelist")).trim();

    // add the role name into the first spot of the vector
    roleUser_vector.addElement(rl);

    roleUser_vector.addElement("roles");

    ResultSet rresult = st.executeQuery("select * from userRole;");

    int Colu = rresult.findColumn( "username");
    int Colr = rresult.findColumn( "rolename");

    String nmu="";
    String nmr="";

    while(rresult.next()) {

        nmu = rresult.getString(Colu);

        nmr = rresult.getString(Colr);

        if (nmr.compareTo(rl) == 0) {

            roleUser_vector.addElement(nmu);

        }

    }

}

```

6.3 User Interfaces**6.3.1 Logon Interfaces**

The public login interface (Figure 6.16) is implemented as an HTML file. It is for all the users, including those who have roles of students, faculty, TA, administrators, and accountants. All the users' login is through this interface. It has the following functions:

- To take a user's username and password, and send them to the server through middle tier for confirming
- To provide a new user with a registration form
- To provide users with information on e-education and programs
- To allow users to address their concerns to relevant personnel

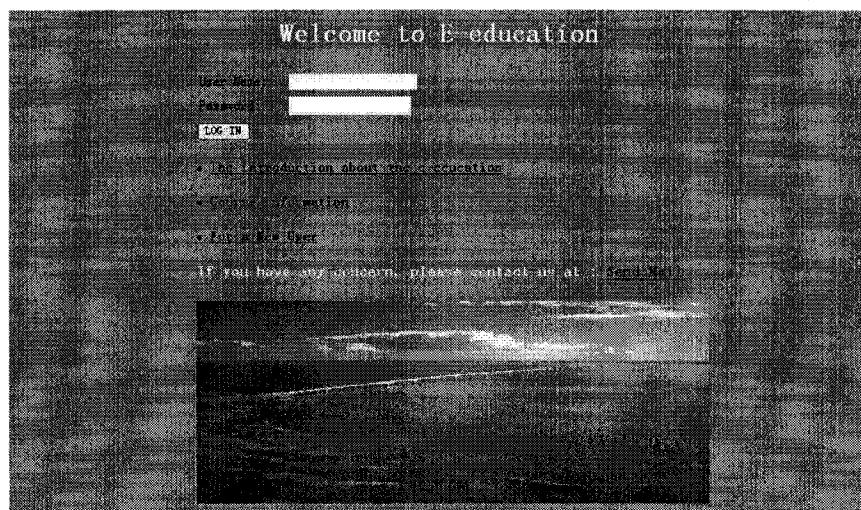


Figure 6.16 Logon interface for regular users

In this process the login servlet is used (Figure 6.17). When a user enters his/her username and password, the e-system will first check the *users* table (Table 6.2),

which contains each user's username, password, and SIN, to verify the user's username and password.

If the logon is successful then the system will check the *sessions* table (Table 6.5), to find if the user has any session and related active role, and return a web page, which contains the appropriate resources for this user. The system will get the records from the *sessions* table based on its primary key, username.

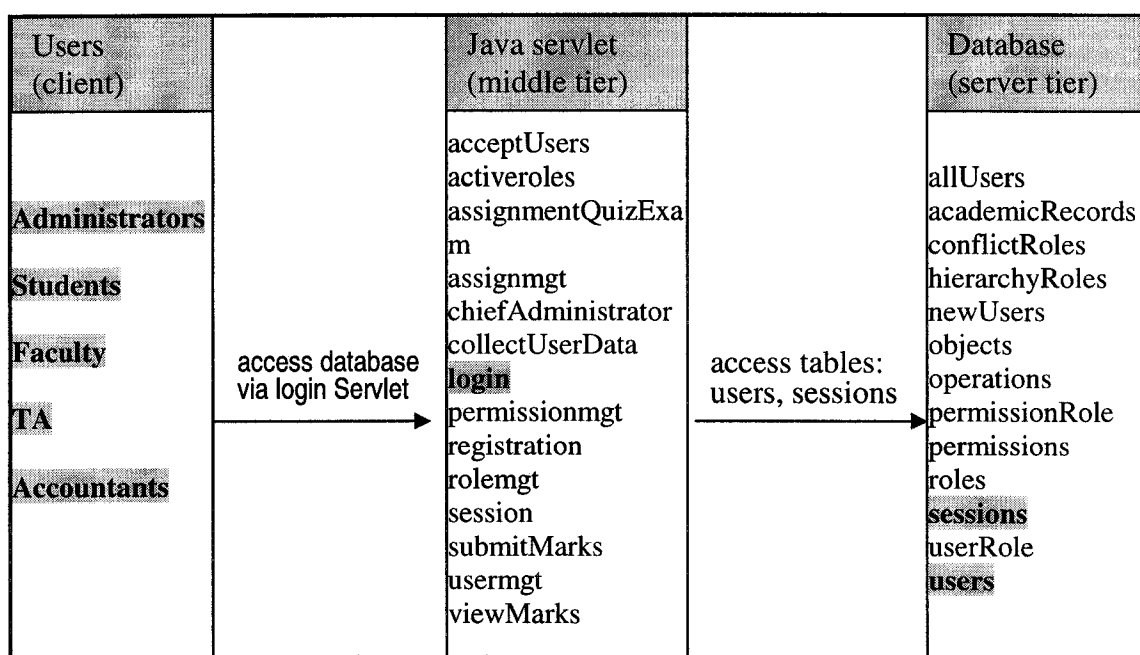


Figure 6.17 Login process

When logon user has the top level administrator role and tries to perform this role's duty, the system will direct the user to the more secured logon interface (Figure 6.18). This interface requires two chief administrators to logon at the same time.

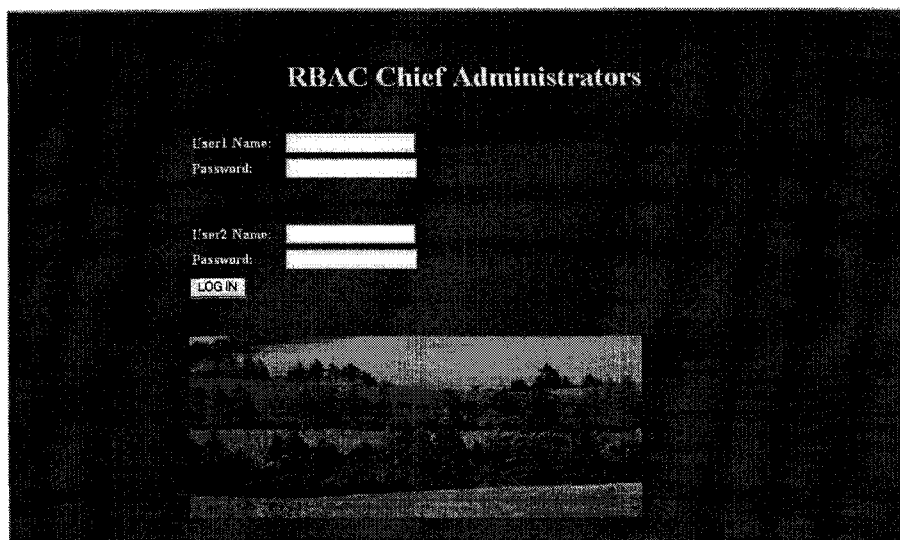


Figure 6.18 Logon interface for top level administrators

In this process, the required servlet is chiefAdminstrator (Figure 6.19).

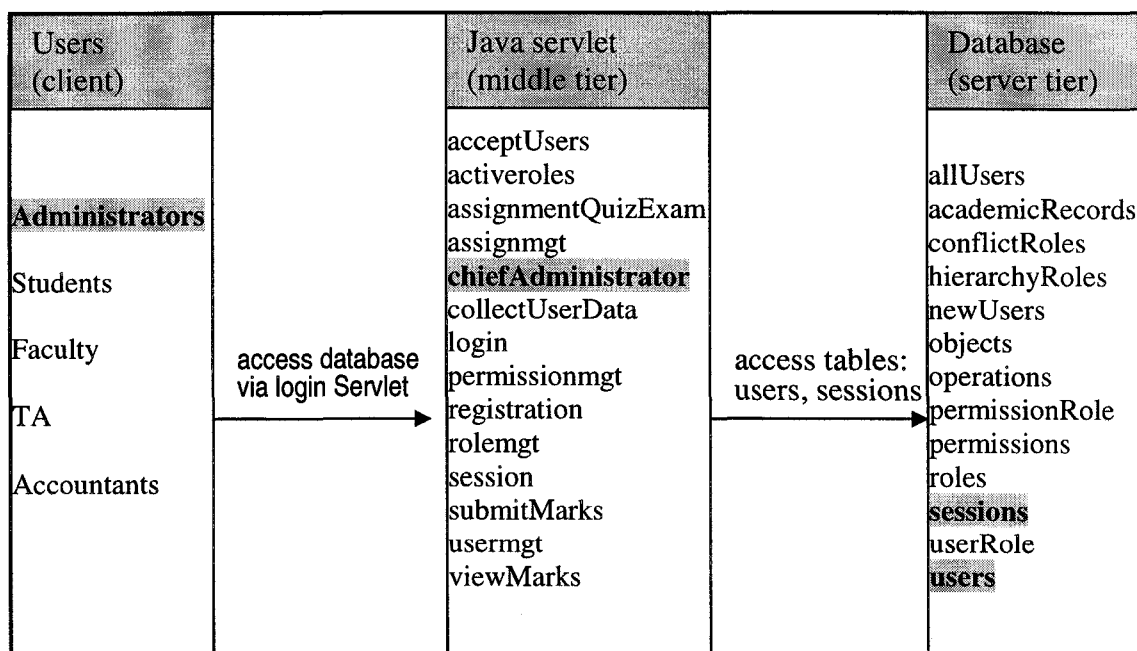


Figure 6.19 Chief Administrators login process

6.3.2 Registration Interface

The registration interface (Figure 6.20) is an HTML file too. It takes a new user's information, such as name, mailing address, designated role request, etc. and submits to the e-education system for acceptance. The submitted information is stored in the *newUsers* table (Table 6.3). In the following demonstration, User1 will be used as an example. Table 6.1 contains User1's registration data.

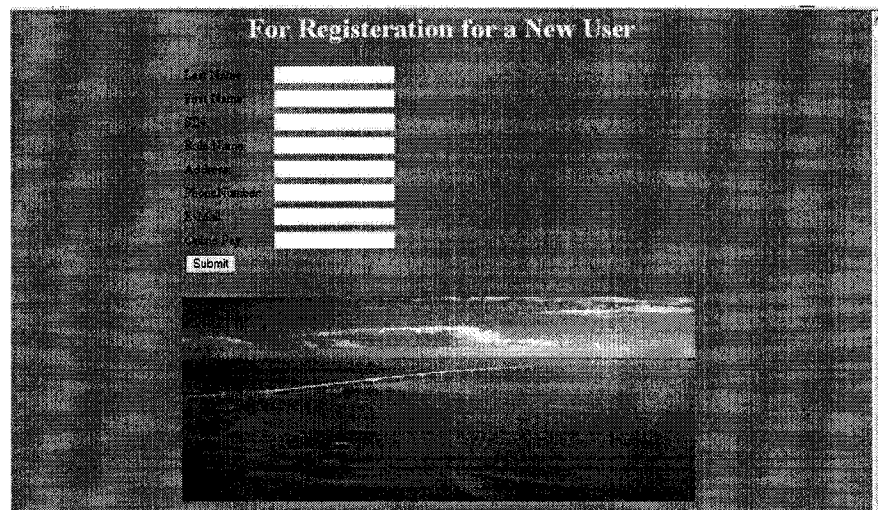
The image shows a web browser window with a dark, textured background. At the top, the title "For Registration for a New User" is displayed. Below the title, there is a list of labels on the left and corresponding input fields on the right. The labels are: Last Name, First Name, ID#, School Name, Address, Phone Number, E-mail, and Date of Birth. Each label has a corresponding white rectangular input field. Below the input fields is a "Submit" button. At the bottom of the form, there is a small rectangular image showing a landscape with a body of water and a bright light source, possibly the sun or moon, reflecting on the water.

Figure 6.20 Registration interface for new users

In this process the *collectUserData* servlet is used. The *newUsers* table is updated (Figure 6.21).

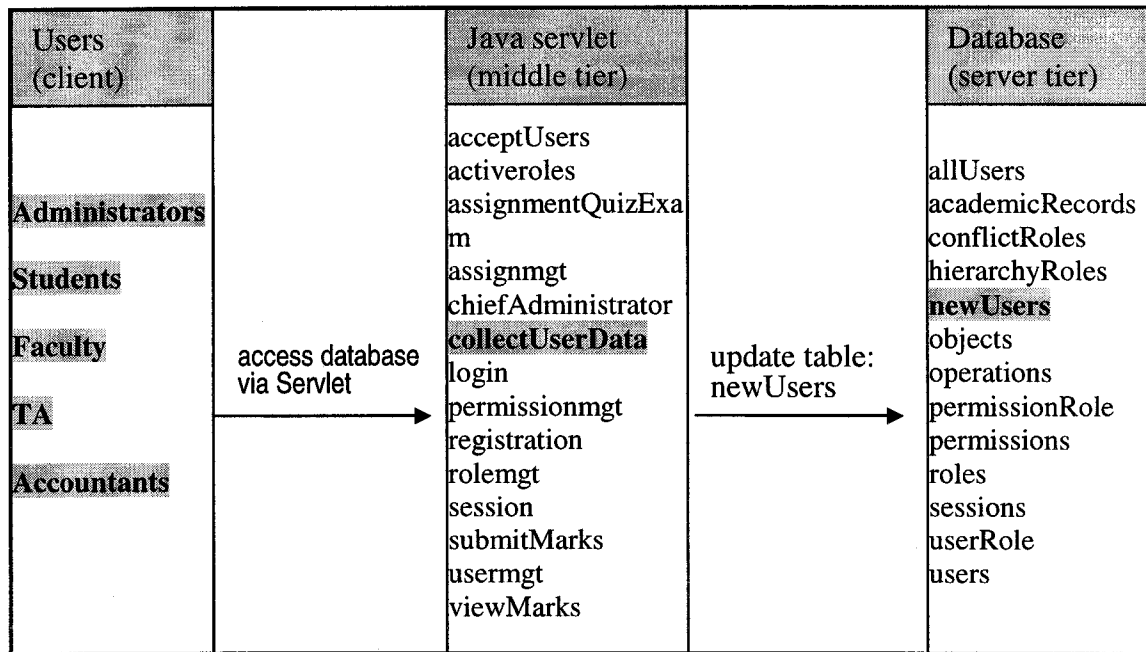


Figure 6.21 New users' registration process

6.3.3 Time-Controlled Interfaces

The availability of an assignment is controlled by its due date. After the due date, the submit button on the assignment interface will not be visible/available.

Students must submit the assignment before the due date. An example interface is shown in Figure 6.22.

This assignment posted on 05/18/2003
AND will be due on 04/09/2004

Your Student ID:

Instructor name:

Course name:

Assignment/Exam title:

Term:

Year:

1. What does RBAC stand for?

2. What is the advantage of RBAC?

3. What are the basic elements for RBAC?

Please put your answers in the following area:

Figure 6.22 Assignment interface with time control

Part of the code for this interface is given as below.

```
<html>

...

<SCRIPT LANGUAGE="JavaScript">

var goLiveDate = "20030618"; // the post date
var expireDate = "20040920"; // the due date

var expireYear = expireDate.substring(0,4)
var expireMonth = expireDate.slice(4,-2)
var expireDay = expireDate.slice(6)
var liveYear = goLiveDate.substring(0,4)
var liveMonth = goLiveDate.slice(4,-2)
var liveDay = goLiveDate.slice(6)
```

```

var nowDate = new Date();

var day = nowDate.getUTCDate();

var month = nowDate.getUTCMonth();

var month1 = month + 1;


if (month1 < 10){ month1 = "0" + month1; }

if (day < 10){ day = "0" + day; }


var year = nowDate.getYear();

var GMTdate = year + "" + month1 + "" + day

if ((GMTdate < expireDate) && (GMTdate >= goLiveDate)) {

    document.write(" <div align='center'><b>This assignment posted on "

+liveMonth+ "/" +liveDay+ "/" +liveYear+ "<br>and will due on "

+expireMonth+ "/" +expireDay+ "/" +expireYear+ ". </b> <br><br> "

+ "Your username: <input type='text' name='username'> <input name='search' type='submit'

value='Submit'> </div>")

}

</script>

...

</html>

```

A quiz must be completed within a specific period of time, starting from the time the user opens the quiz interface. An example of such interface is shown in Figure 6.23.

This quiz must be completed in 30 minutes! WATCH OUT YOUR TIME!

Your username:

Instructor name:

Course name:

Assignment/Quiz/Exam and number:

Term:

Year:

1. List three access control technologies?

2. List the advantage and disadvantages for each of them?

3. What is the major difference between RBAC and other access control?

Please enter your answers in the following area:

Figure 6.23 A quiz interface with time control

Below is part of the code of this quiz interface.

```
...
<body bgcolor=8470FF text="#000000" link="#044d84" vlink="#044d84" alink="#a2937a">
<meta http-equiv="Refresh" content="15; URL=http://cs.stmarys.ca:8000/login.html">
<form action=http://cs.stmarys.ca:8000/servlet/quiz1_servlet method=post>
...
```

In the above two interfaces, when the submit buttons are pressed, the *assignmentQuizExam* servlet will be used, and *academicRecords* table is involved (Figure 6.24).

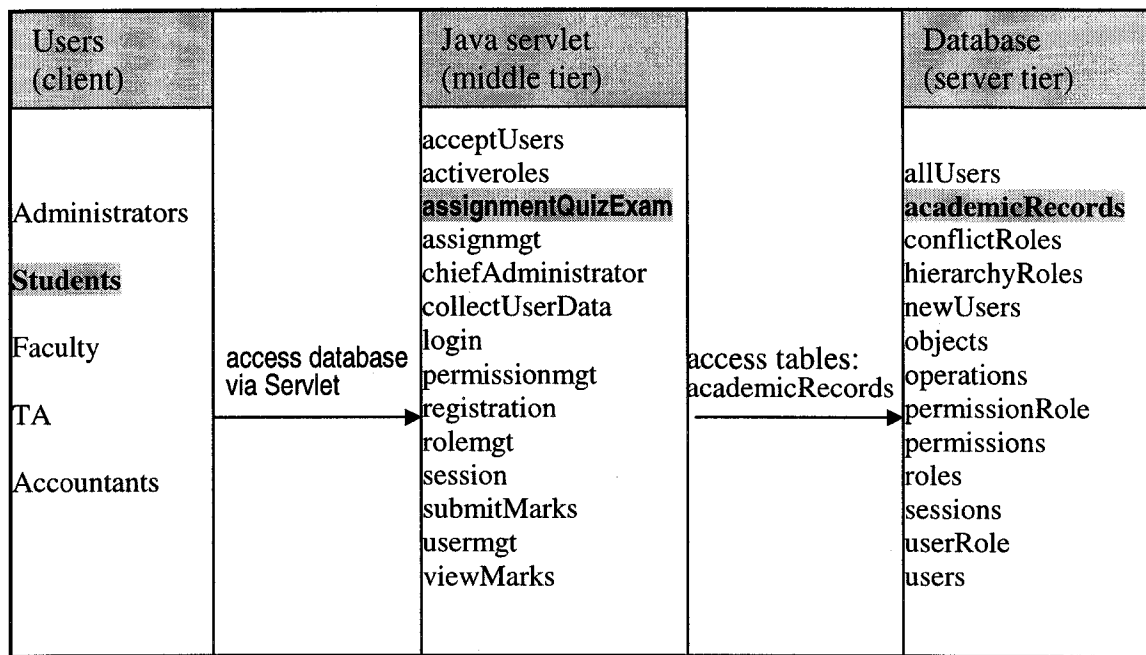


Figure 6.24 Assignment/quiz/exam submission involved processes

6.4 Database implementation

In order to store the system resources, thirteen MySQL database tables are created. This section gives descriptions of these tables.

The table *newUsers* holds new user's registration information. Every time when an administrator logons and confirms the new users' information, all the registration information of the latest new users will be transferred into the table *allUsers* for a permanent storing. Right after the transferring, the table *newUsers* will be emptied for the next batch of new users' registration. The primary key for each of the two tables is SIN (Social Insurance Number).

```
table newUsers (lastname varchar(30), firstname varchar(30), SIN
varchar(9), rolename varchar(30), address varchar(100), phone
varchar(20), email varchar(60), payment varchar(30), PRIMARY KEY
(SIN))
```

```
table allUsers (lastname varchar(30), firstname varchar(30), SIN
varchar(9), rolename varchar(30), address varchar(100), phone
varchar(20), email varchar(60), payment varchar(30), PRIMARY KEY
(SIN))
```

Once confirmed, the new user will be issued a set of username and password; this logon information will be stored in the table *users*. This table will be checked every time a user requests a logon.

```
table users (username varchar(30), password varchar(30), SIN varchar(9)
PRIMARY KEY (SIN))
```

The system roles, objects, operations, and permissions are maintained in the table *roles*, *objects*, *operations*, and *permissions*, respectively.

```
table roles (rolename varchar(30))
```

```
table objects (objectname varchar(60))
```

```
table operations (operationname varchar(30))
```

```
table permissions (operation varchar(30), object varchar(60), PRIMARY  
KEY (operation,object))
```

The *sessions* table, at the beginning is implemented with the fields: user name, session name, and a string of active roles, with user name and session name together as primary key. Later, it is noticed that the coding will not be efficient to remove an active role from the table; it needs a loop to check the active role string, taking more time. Eventually, the *sessions* table was implemented with the fields: username, session name, and individual active role, and all the three fields together becomes the primary key. This makes the adding and removing of records more efficient.

```
table sessions (username varchar(30), sname varchar(30), activeroles  
varchar(236), PRIMARY KEY (username,sname,activeroles))
```

The user-role and permission-role assignment relationships are maintained in the table *userRole* and *permissionRole*.

```
table userRole (username varchar(30), rolename varchar(30), PRIMARY  
KEY (username, rolename))
```

```
table permissionRole (permissionname varchar(97), rolename
varchar(30), privacy varchar(5), PRIMARY KEY (permissionname,
rolename))
```

The conflict role pairs and role hierarchy relations are stored in the following two tables, respectively.

```
table conflictRoles (rolename varchar(30), itsconflictrole varchar(30),
PRIMARY KEY (rolename, itsconflictrole))
```

```
table hierarchyRoles (parent varchar(30), child varchar(30), PRIMARY
KEY (parent, child))
```

The table *academicRecords* holds student academic records. The field title *aqe* represents the title of either assignments, or quizzes, or exams. The field *ans* contains the submitted answers to the assignments/quizzes/exams.

```
table academicRecords (studentID varchar(20), instructorname
varchar(20), coursename varchar(50), term varchar(9), year varchar(4),
aqe varchar(20), ans varchar(227), marks varchar(9), comments
varchar(127), PRIMARY KEY (studentID, coursename, instructorname,
aqe, term, year));
```

The following figure is an overview of all the above database tables and their relationships.

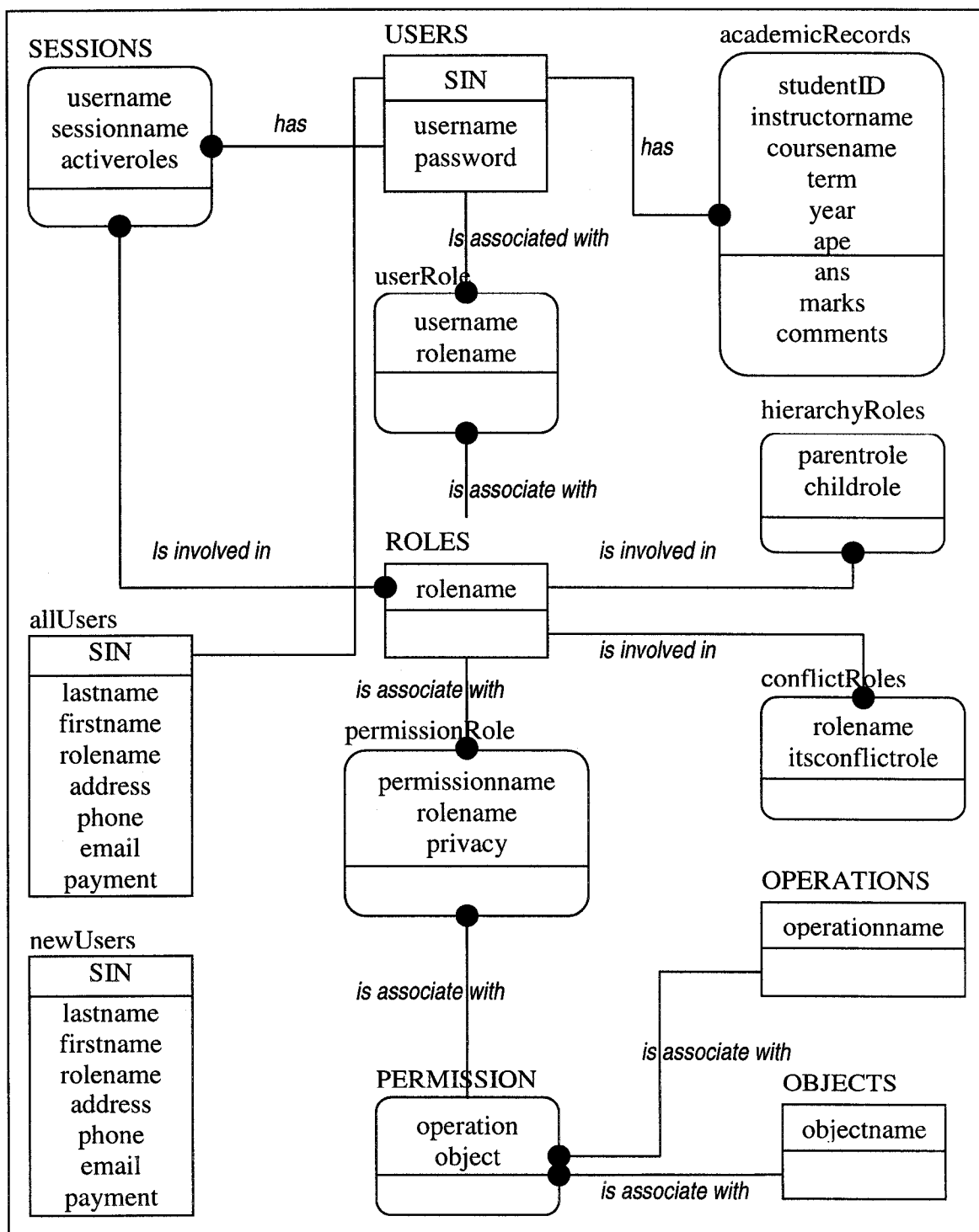


Figure 6.25 Database tables and their relationships

6.5 System test and analysis

To test the capability, efficiency and effectiveness of the e-education system, 1000 users, seven roles and 20 permissions (see Chapter 5) were created and stored in the system. Because the student is the most common role in the system, 900 of the 1000 users are assigned to this role.

With these numbers of users, roles, permissions, and relevant assignment relationships, the system works smoothly: administrators can easily add/remove and display any entity and its relations with other entities. For example, the interface shows that the student role is associated with the 900 users.

To test how efficient and effective the system is when removing a role, we purposely removed the student role from the system. Removing a role means that all the users associated with this role will have no access to the permissions that are associated with this role. As a result there are 900 user-role assignment relations that need to be updated. The system needs to check all the tables that are role-related, such as the tables: *users*, *sessions*, *userRoles*, *rolePermissions*, *hierarchyRoles*, and *conflictRoles*. It takes less than one second for the system to do all the modifications. However, if this is implemented using ACLs that does not have roles but maintains a list of users for each system object, the administrators have to manually check all the user lists and make proper

modifications. This will take much longer time, may produce mistakes inevitably, and also delay the system operations.

In summary, this chapter gives a detailed description of the implementation of the RBAC based e-education system. The implementation includes: building a three tier client-server environment, developing the administrative interfaces and regular user interfaces, developing the system logon and new user registration interfaces, and creating a database to store system resources. The next chapter will give a summary/conclusion of the entire study.

Chapter 7

Summary and future work

7.1 Summary

This thesis studied the Role Based Access Control (RBAC) technology, and explored its application potential in an educational environment. As a result of this study, an RBAC based e-education system is designed and specified, a three tier client-server environment is built, and a prototype of this design is implemented.

The RBAC based e-education system is a software package that can be used by any education organization to manage its daily business operations through the Internet. All users including designated administrators, students, faculty, TA, and account managers have pure online access to the education system. Through the Internet, administrators perform system management duties; students and faculty learn and teach.

To use this RBAC based e-education system, an organization is simply required to install a web server and database server. The e-education system provides two types of interfaces. One is for system administrators to perform

administrative duties, and the other one is for all the users to carry out their own activities. Once they logon, the users will be taken to their corresponding interfaces, which contain the resources they are entitled to. A set of administrative interfaces are provided for the three levels of administrators. Personal data, academic records, assignments, exams, and course curricula are stored in the database server, and they are available via the middle tier - Java Servlet.

The designed model of this study is a hybrid of three RBAC standard models: Core RBAC, Hierarchical RBAC, and Constrained RBAC. It includes all the essential parts of the three standard models. To take privacy issues into consideration, this hybrid model has a unique design: it sets an additional attribute for each permission, that is, each permission is associated with a privacy attribute, either private or non-private. If it is private, this permission will not be inherited, otherwise, the permission can be inherited.

Different from the proposed standard RBAC, this study created a privacy attribute for the system entity - role. The creation of this unique concept can dramatically reduce the number of conflict of interests in the system, also improve privacy.

This study encompasses every step of software development processes: requirement analysis, specification, design, implementation, test, and documentation. It also includes the set-up of a client-server environment, and installation and configuration of software tools and servers.

This study has also demonstrated one of the unique features of RBAC that is most appealing to this study: users' access rights are based on the roles that individual users have as part of an organization and RBAC uses an organization's natural business operation structure. Users are assigned to roles, and permissions are assigned to roles. A user can only access the permissions that are assigned to its role. Compared to a traditional access control technology by which a user's access right is based on the user's identity, RBAC effectively enforces enterprise-specific security policies and streamlines the security management process.

To test this education system, 1000 users, 7 roles, and 20 permissions are set for the system. When a user takes a new or different job (role), administrators will, through the administrative interface, remove the user from the role and assign the user to the new role. It takes less than a second for the administrators to update the change. While in a system which uses ACLs, to update this change, administrators will have to look at all the system objects and update their attached access control lists. Comparing to RBAC system, it will take much longer for the administrators to make the updates in a system that uses ACLs. The analysis obviously shows the advantage of RBAC - reduced management cost.

From this study, it is observed that RBAC has two major advantages: increased security and reduced management cost. The feature of increased security can

be demonstrated in both small scale and large scale application environments, while the feature of reduced management cost is better revealed in a large scale system.

It is also learned that more and more people are now involved in the RBAC research and application development. The impact of RBAC application is potentially great; the use of RBAC applications will result in huge commercial value, particularly in large organizations.

7.2 Future Work

Though the study results have not been used in real world cases, the study is an effort to apply RBAC in e-education system, and demonstrates the potential values of RBAC in e-education application. The groundwork laid by this effort can be used in future development of a commercial software system.

In any future work, it would be reasonable to add more users, roles, courses, and programs in the system to reflect one of the RBAC features - reduced management cost.

In order to make it more realistic, future research can focus on more elaborate interfaces with more interactive functions.

To further enhance the system security, future work may add a monitoring or training program to the e-education system. This program will keep track of the three-levels of administrators' performances, and maintain a record including any mistakes an administrator made and any complaints a user made to an administrator. Based on the record, the administrators will be given a performance evaluation. Only those administrators who perform outstandingly will be promoted to a higher level of permissions in the RBAC system.

Administrators play a vital role in the system security. In the future, the e-education system may use fingerprints to force administrators to logon to the system to perform their administrative duties. This issue is a concern due to the fact that most security breaches are from employees within the organization. With the continuous advances in technology and the demand to tighten security, this type of enforcement would become feasible and applicable in the near future.

References:

1. A Review Paper of Role Based Access Control. 1999.
<http://www.isrc.qut.edu.au/resource/techreport/qut-isrc-tr-1999-004.pdf>
2. Andress Mandy. 2001. "Reach Out and id Someone Access Control".
Information Security. April 2001.
3. Curphey M. and other 16 authors. 2002. "A Guide to Building Secure Web
Applications". The Open Web Application Security Project.
4. Ferraiolo D.F. and Kuhn R. 1992. "*Role Based Access Control*". Proceedings
of the NIST-NSA National Computer Security Conference, pp 554-563.
5. Ferraiolo D.F., Cugini J.A. and Kuhn D.R. 1995. "Role-Based Access Control
(RBAC): Features and Motivations", 11th Annual Computer Security
Applications Proceedings, New Orleans, LA, December 13-15, pp 241-248.
6. Ferraiolo D.F., Barkley J.F. and Kuhn D.R. 1999. "A Role Based Access
Control Model and Reference Implementation Within a Corporate Intranet".
ACM Workshop on Role-Based Access Control.

7. Ferraiolo D.F., Sandhu R., Gavrila S., Kuhn D.R. and Chandramouli R. 2001. "Proposed NIST standard for role based access control". ACM Transactions on Information and System Security, Vol. 4, No. 3, pp. 224-274.
8. Gallaher M.P., O'Connor Alan C. and Kropp Brian. 2002. "The Economic Impact of Role-Based Access Control. RTI Project". March 2002.
9. Gallaher Michael P. 2002. "The economic impact of role-based access control. Report to the National Institute of Standards and Technology". Pages 145. <http://www.nist.gov/director/prog-ofc/report02-1.pdf>
10. Gavrila S.I. and Barkley J.F. 1998. "Formal Specification for Role Based Access Control User/Role and Role/Role Relationship Management". ACM Workshop on Role-Based Access Control.
<http://citeseer.nj.nec.com/gavrila98formal.html>
11. IDEF1X. 2004.
<http://www.essentialstrategies.com/publications/modeling/idef1x.htm>
12. Jaeger T., Michailidis T. and Rada T. 1999. "Access control in a Virtual University". IEEE 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises . June 16-18.

13. Jia X. 2002. ZTC: A type checker for Z notation - user's guide.
<http://venus.cs.depaul.edu/fm/Papers/guide20.pdf>
14. Jordan C.S. 1987. "A guide to understanding discretionary access control in trusted systems". National Computer Security Center.
15. Loscocco P. and Smalley S. 2001. "Integrating Flexible Support for Security Policies into the Linux Operating System", Proceedings of the FREENIX Track: *2001 USENIX Annual Technical Conference*, The USENIX Association, June.
16. MySQL Manual. 2003. <http://dev.mysql.com/doc/mysql/en/What-is.html>
17. Posulns J., Shimonski R.J. and Faircloth J. 2003. "SSCP study guide and DVD training system". Syngress.
18. RBAC Case Study. 2004.
<http://infosecuritymag.techtarget.com/articles/april01/cover.shtml>
19. Role Based Access Control. National Institute of Standards and Technology. 2003. <http://csrc.nist.gov/rbac/>

20. Sandhu R. and Samarati P. 1994. "Access Control: Principles and Practice".

IEEE Computer, pp 40-48, September 1994.

<http://citeseer.ist.psu.edu/sandhu94access.html>

21. Saudhu Ravi. 2002. "Future directions in role based access control models".

http://www.list.gmu.edu/conf/rnc/misconf/pdf_ver/mms01-rbac-future.pdf

22. Spivey J.M. 1989. The Z Notation, A Reference Manual. Prentice Hall

International (UK) Ltd.

23. Sun Trusted Solaris 7 Operating Environment. 2004

http://www.sun.com/software/solaris/trustedsolaris/7/ts_feature_macdac.html

24. Technology Highlights: ACLs (Access Control Lists). 2004

<http://www.ind.alcatel.com/technologies/index.cfm?cnt=acl>

25. What is RBAC. 2003.

http://infosecuritymag.techtarget.com/articles/april01/cover.shtml#case_study

```

/*****
*
*                               Java Servlet codes
*
*****/

import java.io.*;
import java.net.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;

public class acceptUsers_servlet extends HttpServlet {
    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {res.setContentType("text/html");
        PrintWriter toClient = res.getWriter();
        boolean hasNewUser = false;
        try {Class.forName("com.mysql.jdbc.Driver").newInstance();
        } catch (Exception e) {toClient.println("Failed to load JDBC/ODBC driver.");}
        try {Connection con = DriverManager.getConnection(
            "jdbc:mysql://localhost/h_zhao", "h_zhao", "nRy2xN");
            Statement st = con.createStatement();
            st.executeUpdate("use h_zhao;");
            ResultSet result = st.executeQuery("select * from newUsers;");
            int fCol = result.findColumn ("lastname");
            int lCol = result.findColumn ("firstname");
            int sinCol = result.findColumn ("SIN");
            int rCol = result.findColumn ("rolename");
            int adCol = result.findColumn ("address");
            int phCol = result.findColumn ("phone");
            int emCol = result.findColumn ("email");
            int paCol = result.findColumn ("payment");

            while(result.next()) {
                hasNewUser = true;
                String fnm = result.getString(fCol);
                String lnm = result.getString(lCol);
                String sin = result.getString(sinCol);
                String rnm = result.getString(rCol);
                String ad = result.getString(adCol);
                String ph = result.getString(phCol);
                String em = result.getString(emCol);
                String pa = result.getString(paCol);

                String y = "1234567890"; //for username
                String x =
"abcdefghijklmnopqrstuvwxyABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890"; //for password
                String username = "e";
                String pw = "";

                for ( int j=0; j<7; j++ ) {

```

```

        username = username + String.valueOf( y.charAt((int)( Math.random() * y.length() )));
    }

    for ( int i=1; i<7; i++ ) {
        pw = pw + String.valueOf( x.charAt((int)( Math.random() * x.length() )) );
    }

    toClient.println("<html><body bgcolor=\"#065932\">"); // add background color
    toClient.println("Your username is: " + username + ", password is: " + pw);
    toClient.println("<br>");

    st.executeUpdate("insert into users values('"+username+"','"+pw+"','"+sin+"');");

    // assign this role to the user as a default role
    st.executeUpdate("insert into userRole values('"+username+"','"+rnm+"');");
    // create a session s1 for this user and put the role into the session
    st.executeUpdate("insert into sessions values('"+username+"','S1','"+rnm+"');");
    st.executeUpdate("insert into allUsers
values('"+lnm+"','"+fnm+"','"+sin+"','"+rnm+"','"+ad+"','"+ph+"','"+em+"','"+pa+"');");
    }

    st.executeUpdate("delete from newUsers;");

    if (!hasNewUser) {
        toClient.println("<body bgcolor=\"#065932\">"); // add background color
        toClient.println("<br><br>");
        toClient.println("<h3 align=\"center\" style=\"color:white\">Hello administrators: no new
user registered so far! check back later please!</h3>");
    }

    toClient.println("</body></html>");
    toClient.close();
    st.close();
    con.close();
    } catch (Exception e) {e.printStackTrace();}
    }
}

```

```

import java.io.*;
import java.net.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;
import java.util.*;

```

```

public class activeroles_servlet extends HttpServlet {

```

```
static int forPrivacyPurpose = 0; //ensure the current role's private permissions are displayed
static Vector rpv = new Vector(); // prevent the basic role permissions from displaying more than once
```

```
public void doPost(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException{
    res.setContentType("text/html");
    PrintWriter toClient = res.getWriter();
    toClient.println("<html>");
    toClient.println("<body bgcolor=\"8470FF\">");
```

```
String roleBtnName = (req.getParameter("rolebtn")).trim();
```

```
diaplayRolePermissions(toClient, roleBtnName);
forPrivacyPurpose = 0;
rpv.removeAllElements();
```

```
toClient.println("</body></html>");
toClient.close();
```

```
} // end of doPost()
```

```
public void diaplayRolePermissions (PrintWriter out, String roleClicked) { // a recursive function
    try {Class.forName("com.mysql.jdbc.Driver").newInstance();
    } catch (Exception e) {System.out.println("Failed to load JDBC/ODBC driver.");}
    try {Connection con = DriverManager.getConnection
        ("jdbc:mysql://localhost/h_zhao", "h_zhao", "nRy2xN");
        Statement st = con.createStatement();
        st.executeUpdate("use h_zhao;");
```

```
// display this role's permissions
ResultSet rst2 = st.executeQuery("select * from permissionRole;");
int pcol2 = rst2.findColumn ("permissionname");
int ccol2 = rst2.findColumn ("rolename");
int pvc2 = rst2.findColumn ("privacy");
```

```
String pname, mname, pv;
String link = "";
```

```
if (forPrivacyPurpose == 0) {
    while(rst2.next()) {
        link = "http://cs.stmarys.ca:8000/";
        pname = rst2.getString(pcol2);
        pname = getPermName(pname); // trip of the operation and -->
        mname = rst2.getString(ccol2);
        pv = rst2.getString(pvc2);
```

```
if (roleClicked.compareTo(mname)==0 && !(rpv.contains(pname)) ) {
    if (pname.compareTo("chiefAdministratorDuties")!=0){
        link = link+pname+".html";
        out.println("<a href=\""+link+"\"> "+pname+" </a>");
        out.println(" <br> ");
        rpv.addElement(pname);
```

```

    }
    else {
        link = link+"chiefAdministrator_login.html";
        out.println("<a href=\""+link+"\"> "+pname+"</a>");
        out.println(" <br> ");
    }
}
}
forPrivacyPurpose++;
}
else {
    while(rst2.next()) {
        link = "http://cs.stmarys.ca:8000/";
        pname = rst2.getString(pcol2);
        pname = getPermName(pname);
        rname = rst2.getString(ccol2);
        pv = rst2.getString(pvcol2);

        if((roleClicked.compareTo(rname)==0 && pv.compareTo("yes")!=0 &&
!(rpv.contains(pname)))){
            link = link+pname+".html";
            out.println("<a href=\""+link+"\"> "+pname+"</a>");
            out.println(" <br> ");
            rpv.addElement(pname);
        }
    }
}

ResultSet rst = st.executeQuery("select * from hierarchyRoles;");
int pcol = rst.findColumn("parent");
int ccol = rst.findColumn("child");
String childrenRole;
Vector children_vector = new Vector();

while(rst.next()) {
    if (roleClicked.compareTo(rst.getString(pcol))==0) {
        childrenRole = rst.getString(ccol);
        diplayRolePermissions (out, childrenRole);
    }
} //end of rst
st.close();
con.close();
} //end of try con
catch (Exception e) {e.printStackTrace();}
// end of method diplayRolePermissions

public String getPermName (String p) {
    int x = p.indexOf(">");
    return (p.substring(x+1, p.length())).trim();
}
}

```

```

import java.io.*;
import java.net.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;

public class assignmentQuizExam_servlet extends HttpServlet {

    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {res.setContentType("text/html");
        PrintWriter toc = res.getWriter();
        String ans = null;
        String stnm = null;
        String innm = null;
        String conm = null;
        String aqenm = null;
        String te = null;
        String ye = null;

        ans = (req.getParameter("as")).trim();
        stnm = (req.getParameter("username")).trim();
        innm = (req.getParameter("instname")).trim();
        conm = (req.getParameter("coursename")).trim();
        aqenm = (req.getParameter("aqen")).trim();
        te = (req.getParameter("term")).trim();
        ye = (req.getParameter("year")).trim();

        toc.println("<html>");
        toc.println("<body bgcolor=8470FF>");
        toc.println("<br><br><h3>Thank you, "+stnm+" "+, for submitting the work to "+ " "+innm+"!</h3>");
        toc.println("<br><h3>Here is a copy of the answers that you just submitted!</h3><br><br>");
        toc.println(" "+ans+" <br>");
        toc.println("</html>");
        toc.close();

        // store the answers into database
        try {Class.forName("com.mysql.jdbc.Driver").newInstance();
        } catch (Exception e) {toc.println("Failed to load JDBC/ODBC driver.");}
        try {Connection con = DriverManager.getConnection(
            "jdbc:mysql://localhost/h_zhao", "h_zhao", "nRy2xN");
            Statement st = con.createStatement();
            st.executeUpdate("use h_zhao;");
            st.executeUpdate("insert into academicRecords
values('"+stnm+"','"+innm+"','"+conm+"','"+te+"','"+ye+"','"+aqenm+"','"+ans+"',",",");");

```

```

        st.close();
        con.close();

    } catch (Exception e) {e.printStackTrace();}
}

```

```

import java.io.*;
import java.net.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;
import java.util.*;

```

```

public class assignmgt_servlet extends HttpServlet {

    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {

        res.setContentType("text/html");
        PrintWriter toClient = res.getWriter();
        Vector role_vector = new Vector();
        Vector user_vector = new Vector();
        Vector permission_vector = new Vector();
        Vector userRole_vector = new Vector();
        Vector roleUser_vector = new Vector();
        Vector sessionRole_vector = new Vector(); // hold the title
        Vector sessionRole_vector1 = new Vector();
        Vector sessionRole_vector2 = new Vector();
        Vector sessionRole_vector3 = new Vector();

        Vector prmsRole_vector = new Vector();
        Vector rolePrms_vector = new Vector();

        Vector session_vector = new Vector();
        session_vector.addElement("");
        session_vector.addElement("S1");
        session_vector.addElement("S2");
        session_vector.addElement("S3");

        toClient.println("<html>");
        toClient.println("<body bgcolor=\"\#065932\">");
        toClient.println("<h2 style=\"color:white\">ROLE/USER/PERMISSION ASSIGNMENT  
MANAGEMENT</h2>");
    }
}

```

```

toClient.println("<form action=\"http://cs.stmarys.ca:8000/servlet/assignmgt_servlet\"
method=post>");

/***** connect to database and show again *****/
try {Class.forName("com.mysql.jdbc.Driver").newInstance();
} catch (Exception e) {System.out.println("Failed to load JDBC/ODBC driver.");}
try {Connection con = DriverManager.getConnection("jdbc:mysql://localhost/h_zhao", "h_zhao",
"nRy2xN");

Statement st = con.createStatement();
st.executeUpdate("use h_zhao;");

if (req.getParameter( "addbtnUserRole" ) != null){
String u = (req.getParameter("userlist")).trim();
String s = (req.getParameter("sessionlist")).trim();
String r = (req.getParameter("rolelist")).trim();

if(s.compareTo("")==0) { // this is an simple user and role assignment
try {st.executeUpdate("insert into userRole values (" +u+""," +r+"");");}
catch (SQLException sqle){
toClient.println("<font color = \"red\">");
toClient.println("duplicate entry, user-role assignment already exists");
toClient.println("</font>");
}
}
else { // this is a session creation
// first have to chech if the user has such a role, if not, can not assign the role to the session
boolean theUserHasTheRole = false;
ResultSet sresult = st.executeQuery("select * from userRole;");
int Colu = sresult.findColumn( "username");
int Colr = sresult.findColumn( "rolename");
String unm="";
String rnm="";
while(sresult.next()) {
unm = sresult.getString(Colu);
rnm = sresult.getString(Colr);
if (unm.compareTo(u) == 0 && rnm.compareTo(r) == 0) {
theUserHasTheRole = true;
break;
}
}
}

if (theUserHasTheRole == true) {
try {st.executeUpdate("insert into sessions values (" +u+""," +s+""," +r+"");");
} catch (SQLException sqle){
toClient.println("<font color = \"red\">");
toClient.println("duplicate entry, this session-role already exists");
toClient.println("</font>");
}
}
else

```



```

        toClient.println("The user does not have such a role! Please assign the role to
the user, then you can add this role to a session!");
    }
}
else if (req.getParameter( "debtnUserRole" ) != null){
    String u = (req.getParameter("userlist")).trim();
    String s = (req.getParameter("sessionlist")).trim();
    String r = (req.getParameter("rolelist")).trim();

    if (s.compareTo("")==0) { // this is to remove the user from the role
        st.executeUpdate("delete from userRole where username='"+u+"' and
rolename='"+r+"';");
        st.executeUpdate("delete from sessions where username='"+u+"' and
activeroles='"+r+"';");
    }
    else // just remove the role from the user's this session
        st.executeUpdate("delete from sessions where username='"+u+"' and sname='"+s+"'
and activeroles='"+r+"';");
}

else if (req.getParameter( "addbtnPermRole" ) != null){
    String p = (req.getParameter("permlist")).trim();
    String r = (req.getParameter("prolelist")).trim();
    String ck = req.getParameter("yesnocheck"); // important! don't trim THIS ONE
    String privacy = "no";

    if ( ck != null)
        privacy = "yes";

    try { st.executeUpdate("insert into permissionRole values ('"+p+"','"+r+"','"+privacy+"');"); }
    catch (SQLException sqle){
        toClient.println("<font color = \"red\">");
        toClient.println("duplicate entry, this permission-role assignment already exists");
        toClient.println("</font>");
    }
}

else if (req.getParameter( "debtnPermRole" ) != null){
    String p = (req.getParameter("permlist")).trim();
    String r = (req.getParameter("prolelist")).trim();
    st.executeUpdate("delete from permissionRole where permissionname='"+p+"' and
rolename='"+r+"';");
}

else if (req.getParameter( "showbtnUserRole" ) != null){
    String ur = (req.getParameter("userlist")).trim();
    // add the user name into the first spot of the vector
    userRole_vector.addElement(ur);
    userRole_vector.addElement("roles");
    ResultSet uresult = st.executeQuery("select * from userRole;");
    int Colu = uresult.findColumn( "username");

```

```

int Colr = uresult.findColumn ("rolename");
String nmu="";
String nmr="";
while(uresult.next()) {
    nmu = uresult.getString(Colu);
    nmr = uresult.getString(Colr);
    if (nmu.compareTo(ur) == 0) {
        userRole_vector.addElement(nmr);
    }
}

// show the user's sessions and roles
sessionRole_vector.addElement(ur);
sessionRole_vector.addElement("sessions/roles");
sessionRole_vector1.addElement(ur);
sessionRole_vector1.addElement("session 1");
sessionRole_vector2.addElement(ur);
sessionRole_vector2.addElement("session 2");
sessionRole_vector3.addElement(ur);
sessionRole_vector3.addElement("session 3");

ResultSet rt = st.executeQuery("select * from sessions;");
int Cu = rt.findColumn ("username");
int Cs = rt.findColumn ("sname");
int Cr = rt.findColumn ("activeroles");
String usernm="";
String sesnm="";
String actnm="";
while(rt.next()) {
    usernm = rt.getString(Cu);
    sesnm = rt.getString(Cs);
    actnm = rt.getString(Cr);
    if (usernm.compareTo(ur) == 0) {
        if (sesnm.compareTo("S1")==0)
            sessionRole_vector1.addElement(actnm);
        else if (sesnm.compareTo("S2")==0)
            sessionRole_vector2.addElement(actnm);
        else if (sesnm.compareTo("S3")==0)
            sessionRole_vector3.addElement(actnm);
    }
}

}

else if (req.getParameter( "showbtnRoleUser" ) != null){
    String rl = (req.getParameter("rolelist")).trim();
    // add the role name into the first spot of the vector
    roleUser_vector.addElement(rl);
    roleUser_vector.addElement("roles");
    ResultSet rresult = st.executeQuery("select * from userRole;");
    int Colu = rresult.findColumn ("username");
    int Colr = rresult.findColumn ("rolename");

```

```

String nmu="";
String nmr="";
while(rresult.next()) {
    nmu = rresult.getString(Colu);
    nmr = rresult.getString(Colr);
    if (nmr.compareTo(rl) == 0) {
        roleUser_vector.addElement(nmu);
    }
}
}

else if (req.getParameter( "showbtnRolePrms" ) != null){
    String rp = (req.getParameter("prolelist")).trim();
    // add the role name into the first spot of the vector
    rolePrms_vector.addElement(rp);
    rolePrms_vector.addElement("permissions");
    ResultSet rresult = st.executeQuery("select * from permissionRole;");
    int Colp = rresult.findColumn( "permissionname");
    int Colr = rresult.findColumn( "rolename");
    String nmp="";
    String nmr="";
    while(rresult.next()) {
        nmp = rresult.getString(Colp);
        nmr = rresult.getString(Colr);
        if (nmr.compareTo(rp) == 0) {
            rolePrms_vector.addElement(nmp);
        }
    }
}

else if (req.getParameter( "showbtnPrmsRole" ) != null){
    String pr = (req.getParameter("permilist")).trim();
    prmsRole_vector.addElement(pr);
    prmsRole_vector.addElement("roles");
    ResultSet prresult = st.executeQuery("select * from permissionRole;");
    int Colp = prresult.findColumn( "permissionname");
    int Colr = prresult.findColumn( "rolename");
    String nmp="";
    String nmr="";
    while(prresult.next()) {
        nmp = prresult.getString(Colp);
        nmr = prresult.getString(Colr);
        if (nmp.compareTo(pr) == 0) {
            prmsRole_vector.addElement(nmr);
        }
    }
}

ResultSet rolresult = st.executeQuery("select * from roles;");
int rCol = rolresult.findColumn( "rolename");
String rnm="";

```

```

while(roleresult.next()) {
    rnm = roleresult.getString(rCol);
    role_vector.addElement(rnm);
}

ResultSet userresult = st.executeQuery("select * from users;");
int uCol = userresult.findColumn ("username");
String unnm="";
while(userresult.next()) {
    unnm = userresult.getString(uCol);
    user_vector.addElement(unnm);
}

ResultSet permresult = st.executeQuery("select * from permissions;");
int opeCol = permresult.findColumn ("operation");
int objCol = permresult.findColumn ("object");
String oper, obje;
while(permresult.next()) {
    oper = permresult.getString(opeCol);
    obje = permresult.getString(objCol);
    permission_vector.addElement(oper + " --> " + obje);
}

st.close();
con.close();
} catch (Exception e) {e.printStackTrace();}

/***** update and display assignments *****/
toClient.println("<h3 style=\"color:white\">User/Role Assignment</h3>");
toClient.println("<table border =\"1\">");
toClient.println("<tr>");
toClient.println("<td><h4 style=\"color:white\">USERS</h4></td>");
toClient.println("<td><h4 style=\"color:white\">SESSIONS</h4></td>");
toClient.println("<td><h4 style=\"color:white\">ROLES</h4></td>");
toClient.println("</tr>");

toClient.println("<tr>");
toClient.println("<td>");
toClient.println("<select name=\"userlist\">");
String ustr = "";
int s = user_vector.size();
for (int i = 0; i<s; i++) {
    ustr = (String)user_vector.elementAt(i);
    toClient.println("<option>"+ustr+"");
}
toClient.println("</select>");
toClient.println("</td>");

toClient.println("<td>");
toClient.println("<select name=\"sessionlist\">");
String snstr = "";

```

```

int sn = session_vector.size();
for (int i = 0; i < sn; i++) {
    snstr = (String)session_vector.elementAt(i);
    toClient.println("<option>" + snstr + "");
}
toClient.println("</select>");
toClient.println("</td>");

toClient.println("<td>");
toClient.println("<select name=\"rolelist\">");
String rstr = "";
int t = role_vector.size();
for (int i = 0; i < t; i++) {
    rstr = (String)role_vector.elementAt(i);
    toClient.println("<option>" + rstr + "");
}
toClient.println("</select>");
toClient.println("</td>");
toClient.println("</tr>");
toClient.println("</table>");

toClient.println("<input name=\"addbtnUserRole\" type=\"submit\" value=\"Assign selected user to
selected role/session\">");
toClient.println("<input name=\"debbtnUserRole\" type=\"submit\" value=\"Remove selected user from
selected role\">");
toClient.println(" <br><br><br>");

toClient.println("<input name=\"showbtnUserRole\" type=\"submit\" value=\"Display selected user's
roles/sessions\">");
diaplayAttributes(toClient, userRole_vector); // some of the roles may not be in the user's sessions
diaplayAttributes(toClient, sessionRole_vector);
diaplayAttributes(toClient, sessionRole_vector1);
diaplayAttributes(toClient, sessionRole_vector2);
diaplayAttributes(toClient, sessionRole_vector3);
toClient.println("<br><br><br>");
toClient.println("<input name=\"showbtnRoleUser\" type=\"submit\" value=\"Display selected role's
users\">");
diaplayAttributes(toClient, roleUser_vector);
toClient.println("<br><br><br><br><br>");

/***** permission role assignment *****/
toClient.println("<h3 style=\"color:white\">Permission/Role Assignment</h3>");
toClient.println("<table border =\"1\">");
toClient.println("<tr>");
toClient.println("<td><h4 style=\"color:white\">PERMISSIONS</h4></td>");
toClient.println("<td><h4 style=\"color:white\">ROLES</h4></td>");
toClient.println("</tr>");
toClient.println("<tr>");
toClient.println("<td>");
toClient.println("<select name=\"permlist\">");
String pstr = "";

```

```

        int ps = permission_vector.size();
        for (int i = 0; i < ps; i++) {
            pstr = (String)permission_vector.elementAt(i);
            toClient.println("<option>" + pstr + "");
        }
        toClient.println("</select>");
        toClient.println("</td>");
        toClient.println("<td>");
        toClient.println("<select name=\"proleiist\">");
        String prstr = "";
        int pt = roie_vector.size();
        for (int i = 0; i < pt; i++) {
            prstr = (String)role_vector.elementAt(i);
            toClient.println("<option>" + prstr + "");
        }
        toClient.println("</select>");
        toClient.println("</td>");
        toClient.println("<td>");
        toClient.println("<input name=\"yesnocheck\" type=\"CHECKBOX\"><font color=\"white\"> Private
permission!</font>");
        toClient.println("</td>");
        toClient.println("</tr>");
        toClient.println("</table>");
        toClient.println("<input name=\"addbtnPermRole\" type=\"submit\" value=\"Assign selected
permission to selected role\">");
        toClient.println("<input name=\"debbtnPermRole\" type=\"submit\" value=\"Remove selected
permission from selected role\">");
        toClient.println(" <br><br><br>");
        toClient.println("<input name=\"showbtnRolePrms\" type=\"submit\" value=\"Display selected role's
permissions\">");
        diplayAttributes(toClient, rolePrms_vector);
        toClient.println("<br><br><br>");
        toClient.println("<input name=\"showbtnPrmsRole\" type=\"submit\" value=\"Display selected
permission's roies\">");
        diplayAttributes(toClient, prmsRole_vector);
        toClient.println("<br><br><br><br><br>");
        toClient.println("<br><br><br><br><br>");
        toClient.println("</form></body></html>");
        toClient.close();
    }

    public void diplayAttributes (PrintWriter out, Vector v) {
        int s = v.size();
        if (s > 2) { // has to be greater than 2, to prevent the case - no session
            String str0 = " " + (String)v.elementAt(0) + " ";
            String fl = " " + (String)v.elementAt(1);

            out.println("<p><font color=\"white\">" + str0 + " has the following " + fl + ": </font>");
            out.println("<font color=\"white\">");

            for (int i = 2; i < s; i++) {

```

```

        String str = (String)v.elementAt(i);
        out.println("<li>"+str+"");
    }
    out.println("</font>");
}
}
}

```

```

import java.io.*;
import java.net.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;

```

```

public class chiefAdministrator_login_servlet extends HttpServlet {

    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter toClient = res.getWriter();
        toClient.println("<form action=\"http://cs.stmarys.ca:8000/servlet/session_servlet\" method=post>");
        String in_userm1 = null;
        String in_passwd1 = null;
        String in_userm2 = null;
        String in_passwd2 = null;

        Vector ussero_vector1 = new Vector();
        Vector ussero_vector2 = new Vector();
        Vector ussero_vector3 = new Vector();
        int numsessions = 0;

        in_userm1 = (req.getParameter("username1")).trim();
        in_passwd1 = (req.getParameter("password1")).trim();
        in_userm2 = (req.getParameter("username2")).trim();
        in_passwd2 = (req.getParameter("password2")).trim();

        try { Class.forName("com.mysql.jdbc.Driver").newInstance(); }
        catch (Exception e) { System.out.println("Failed to load JDBC/ODBC driver."); }

        try {Connection con = DriverManager.getConnection("jdbc:mysql://localhost/h_zhao", "h_zhao",
"nRy2xN");
            Statement st = con.createStatement();
            st.executeUpdate("use h_zhao;");

            ResultSet result = st.executeQuery("select * from users;");

```

```

int usernmCol = result.findColumn ("username");
int passwdCol = result.findColumn ("password");

String usernm = "", passwd = "";
String returnPage = "";
int user1ok = 0, user2ok =0;

while(result.next()) {
    usernm = result.getString(usernmCol);
    passwd = result.getString(passwdCol);

    if (usernm.equals (in_usernm1) && passwd.equals (in_passwd1) ) {
        user1ok=1;
    }
    else if (usernm.equals (in_usernm2) && passwd.equals (in_passwd2) ) {
        user2ok=1;
    }
}

// ----- get a user's sessions from the table sessions
ResultSet seresult = st.executeQuery("select * from sessions;");
int usCol = seresult.findColumn ("username");
int seCol = seresult.findColumn ("sname");
int roCol = seresult.findColumn ("activeroles");

String usnm, senm, ronm;

while(seresult.next()) {
    usnm = seresult.getString(usCol);
    senm = seresult.getString(seCol);
    ronm = seresult.getString(roCol);

    if (usnm.equals (in_usernm1) ) {
        numsessions++;
        if (senm.equals("S1")) {
            ussero_vector1.addElement(ronm);
        }
        else if (senm.equals("S2")) {
            ussero_vector2.addElement(ronm);
        }
        else if (senm.equals("S3")) {
            ussero_vector3.addElement(ronm);
        }
    }
}

// ----- check how many sessions this user has -----
if (ussero_vector1.size() != 0)
    numsessions++;
if (ussero_vector2.size() != 0)
    numsessions++;
if (ussero_vector3.size() != 0)

```



```

        numsessions++;
// -----

        // check login username and password
        if((user1ok==1)&&(user2ok==1)) {
            // login succeed, Respond to client

            Runtime rt = Runtime.getRuntime();
            try{ rt.exec("chmod 755 abc.html"); }
            catch(Exception e) { System.out.println(e.getMessage()); }

            String toPage="http://cs.stmarys.ca:8000/chiefAdministratorDuties.html";
            res.sendRedirect(toPage);
        }
        else {
            // login not succeed, re-login
            String relogin="http://cs.stmarys.ca:8000/chiefAdministrator_login.html";
            res.sendRedirect(relogin);
        }
        toClient.println("</form>");
        toClient.close();
        st.close();
        con.close();
    } catch (Exception e) { e.printStackTrace(); }
}
}

```

```

import java.io.*;
import java.net.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;

```

```

public class collectUserData_servlet extends HttpServlet {

    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter toClient = res.getWriter();
        String lastnm = (req.getParameter("lastname")).trim();
        String firstnm = (req.getParameter("firstname")).trim();
        String sin = (req.getParameter("sin")).trim();
        String rolenm = (req.getParameter("rolename")).trim();
        String addr = (req.getParameter("address")).trim();
        String ph = (req.getParameter("phone")).trim();
    }
}

```

```

String email = (req.getParameter("email")).trim();
String pay = (req.getParameter("pay")).trim();

try {Class.forName("com.mysql.jdbc.Driver").newInstance();
} catch (Exception e) {System.out.println("Failed to load JDBC/ODBC driver.");}
try {
    Connection con = DriverManager.getConnection(
        "jdbc:mysql://localhost/h_zhao", "h_zhao", "nRy2xN");

    Statement st = con.createStatement();
    st.executeUpdate("use h_zhao;");
    st.executeUpdate("insert into newUsers values('"+lastnm+"','"+firstnm+"','"+sin+"','"+rolenm+"',
"+addr+"','"+ph+"','"+email+"','"+pay+"');");

    String relogin="http://cs.stmarys.ca:8000/finishRegistration.html";
    res.sendRedirect(relogin);
    toClient.close();
    st.close();
    con.close();
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

```

import java.io.*;
import java.net.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;

```

```

public class login_servlet extends HttpServlet {
    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter toClient = res.getWriter();
        toClient.println("<form action=\"http://cs.stmarys.ca:8000/servlet/session_servlet\" method=post>");
        String in_userrnm = null;
        String in_passwd = null;

        Vector usserrnm_vector1 = new Vector();
        Vector usserrnm_vector2 = new Vector();
        Vector usserrnm_vector3 = new Vector();
        int s1ActiveRoles = 0;
        int s2ActiveRoles = 0;
        int s3ActiveRoles = 0;
    }
}

```

```

int numsessions = 0;

in_usernm = (req.getParameter("username")).trim();
in_passwd = (req.getParameter("password")).trim();

try { Class.forName("com.mysql.jdbc.Driver").newInstance(); }
catch (Exception e) { System.out.println("Failed to load JDBC/ODBC driver."); }

try {
    Connection con = DriverManager.getConnection("jdbc:mysql://localhost/h_zhao", "h_zhao",
"nRy2xN");

    Statement st = con.createStatement();
    st.executeUpdate("use h_zhao;");
    ResultSet result = st.executeQuery("select * from users;");
    int usernmCol = result.findColumn ("username");
    int passwdCol = result.findColumn ("password");
    String usernm = "", passwd = "";
    String returnPage = "";
    int usernmok = 0, passwdok =0;

    while(result.next()) {
        usernm = result.getString(usernmCol);
        passwd = result.getString(passwdCol);

        if (usernm.equals (in_usernm)) {
            usernmok=1;
            if (passwd.equals (in_passwd)) {
                passwdok=1;
                break;
            }
        }
    }

    // ----- get a user's sessions from the table sessions
    ResultSet seresult = st.executeQuery("select * from sessions;");
    int usCol = seresult.findColumn ("username");
    int seCol = seresult.findColumn ("sname");
    int roCol = seresult.findColumn ("activeroles");
    String usnm, senm, ronm;
    while(seresult.next()) {
        usnm = seresult.getString(usCol);
        senm = seresult.getString(seCol);
        ronm = seresult.getString(roCol);
        if (usnm.equals (usernm)) {
            numsessions++;
            if (senm.equals("S1")) {
                ussero_vector1.addElement(ronm);
            }
            else if (senm.equals("S2")) {
                ussero_vector2.addElement(ronm);
            }
        }
    }
}

```

```

        else if (senm.equals("S3")) {
            ussero_vector3.addElement(ronm);
        }
    }
}
// ----- check how many sessions this user has -----
if (ussero_vector1.size() != 0)
    numsessions++;
if (ussero_vector2.size() != 0)
    numsessions++;
if (ussero_vector3.size() != 0)
    numsessions++;

// check login username and password
if((usernmok==1)&&(passwdok==1)) {
    // login succeed, Respond to client
    displayResultPage(toClient, ussero_vector1, ussero_vector2, ussero_vector3);
}
else {
    // login not succeed, re-login
    String relogin="http://cs.stmarys.ca:8000/relogin.html";
    res.sendRedirect(relogin);
}
toClient.println("</form>");
toClient.close();
st.close();
con.close();
} catch (Exception e) { e.printStackTrace(); }
}

//-----
//          displayResultPage
//-----
public void displayResultPage( PrintWriter out, Vector v1, Vector v2, Vector v3 ) {
    out.println( "<html>" );
    out.println( "<body bgcolor=8470FF>" );
    out.println( "<br><br><center>" );
    out.println( "<h2><font face=\"Verdana\">Welcome to RBAC system.</font></h2>" );
    out.println( "</center>" );
    out.println( "<center>" );
    out.println( "<h2><font face=\"Verdana\">You have the following sessions and roles.</font></h2>" );
    out.println( "</center>" );
    out.println( "<br><hr>" );

    // determine how many sessions the user have
    int numOFsessions = 1; // a user has one session by default
    if (v2.size() > 0 || v3.size() > 0) {
        numOFsessions++;
    }
    if (numOFsessions > 1) { // once going to next page, will not be able to back to this page
        out.println( "<script language=\"JavaScript\">" );

```

```

        out.println(" var nHist = window.history.length; ");
        out.println(" if(window.history[nHist] != window.location) ");
        out.println(" window.history.forward(); ");
        out.println(" </script> ");
    }
    out.println( "<ul>" );

    if (v1.size() > 0) {
        String sessionRoles = "Session 1: ";
        for (int i = 0; i < v1.size(); i++) {
            if (i != v1.size()-1)
                sessionRoles = sessionRoles + (String)(v1.elementAt(i)) + ", ";
            else sessionRoles = sessionRoles + (String)(v1.elementAt(i));
        }
        out.println("<input type = \"submit\" name=\"session\" value=\"" + sessionRoles + "\">");
        out.println("<br><br><br>");
    }

    if (v2.size() > 0) {
        String sessionRoles = "Session 2: ";
        for (int i = 0; i < v2.size(); i++) {
            if (i != v2.size()-1)
                sessionRoles = sessionRoles + (String)(v2.elementAt(i)) + ", ";
            else sessionRoles = sessionRoles + (String)(v2.elementAt(i));
        }
        out.println("<input type = \"submit\" name=\"session\" value=\"" + sessionRoles + "\">");
        out.println("<br><br><br>");
    }

    if (v3.size() > 0) {
        String sessionRoles = "Session 3: ";
        for (int i = 0; i < v3.size(); i++) {
            if (i != v3.size()-1)
                sessionRoles = sessionRoles + (String)(v3.elementAt(i)) + ", ";
            else sessionRoles = sessionRoles + (String)(v3.elementAt(i));
        }
        out.println("<input type = \"submit\" name=\"session\" value=\"" + sessionRoles + "\">");
        out.println("<br><br><br>");
    }
    out.println( "</ul>" );
}
}

```

```

import java.io.*;
import java.net.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;

```

```

import javax.servlet.http.*;
import java.util.Date;
import java.util.*;

public class permissionmgt_servlet extends HttpServlet {
    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter toClient = res.getWriter();
        Vector object_vector = new Vector();
        Vector operation_vector = new Vector();
        Vector permission_vector = new Vector();

        toClient.println("<html>");
        toClient.println("<body bgcolor=\"\#065932\">");
        toClient.println("<h2 style=\"color:white\">PERMISSION MANAGEMENT</h2>");
        toClient.println("<form action=\"http://cs.stmarys.ca:8000/servlet/permissionmgt_servlet\"
method=post>");
        /***** connect to database and show again *****/
        try {Class.forName("com.mysql.jdbc.Driver").newInstance();
        } catch (Exception e) {System.out.println("Failed to load JDBC/ODBC driver.");}
        try {Connection con = DriverManager.getConnection("jdbc:mysql://localhost/h_zhao", "h_zhao",
"nRy2xN");
            Statement st = con.createStatement();
            st.executeUpdate("use h_zhao;");

            if (req.getParameter( "addbtnOP" ) != null){
                try {st.executeUpdate("insert into operations values
("+(req.getParameter("newoperation")).trim()+",");}
                catch (SQLException sqle){
                    toClient.println("<font color = \"red\">");
                    toClient.println("duplicate entry, the operation already exists");
                    toClient.println("</font>");
                }
            }

            else if (req.getParameter( "debtnOP" ) != null){
                st.executeUpdate("delete from operations where
operationname="+ (req.getParameter("operationlist")).trim()+",");
                st.executeUpdate("delete from permissions where
operation="+ (req.getParameter("operationlist")).trim()+",");
            }

            else if (req.getParameter( "addbtnOBJ" ) != null){
                try {st.executeUpdate("insert into objects values ("+(req.getParameter("newobject")).trim()+",");}
                catch (SQLException sqle){
                    toClient.println("<font color = \"red\">");
                    toClient.println("duplicate entry, the objects already exists");
                    toClient.println("</font>");
                }
            }
        }
    }
}

```

```

        else if (req.getParameter( "debtnOBJ" ) != null){
st.executeUpdate("delete from objects where objectname='"+(req.getParameter("objectlist")).trim()+"';");
st.executeUpdate("delete from permissions where object='"+(req.getParameter("objectlist")).trim()+"';");
        }
        else if (req.getParameter( "addbtnPERM" ) != null){
            String op = (req.getParameter("operationlist1")).trim();
            String obj = (req.getParameter("objectlist1")).trim();
            try {st.executeUpdate("insert into permissions values ('"+op+"','"+obj+"');"); }
            catch (SQLException sqle){
                toClient.println("<font color = \"red\">");
                toClient.println("duplicate entry, the permission already exists");
                toClient.println("</font>");
            }
        }
    }
    else if (req.getParameter( "debtnPERM" ) != null){
        String thePair = (req.getParameter("permissionlist")).trim();
        int sp = thePair.indexOf("-");
        String op = thePair.substring(0,sp-1);
        String ob = thePair.substring(sp+3,thePair.length());
st.executeUpdate("delete from permissions where operation='"+op+"' and object = '"+ob+"';");
    }

    ResultSet opresult = st.executeQuery("select * from operations;");
    int opCol = opresult.findColumn( "operationname");
    String opnm="";
    while(opresult.next()) {
        opnm = opresult.getString(opCol);
        operation_vector.addElement(opnm);
    }

    ResultSet result = st.executeQuery("select * from objects;");
    int oCol = result.findColumn( "objectname");
    String objnm="";
    while(result.next()) {
        objnm = result.getString(oCol);
        object_vector.addElement(objnm);
    }

    ResultSet permresult = st.executeQuery("select * from permissions;");
    int operCol = permresult.findColumn( "operation");
    int objeCol = permresult.findColumn( "object");
    String oper="";
    String obje="";
    while(permresult.next()) {
        oper = permresult.getString(operCol);
        obje = permresult.getString(objeCol);
        permission_vector.addElement(oper + " -- " + obje);
    }
    st.close();
    con.close();

```

```

    } catch (Exception e) {e.printStackTrace();}

    /***** update and display operations *****/
    toClient.println("<h4 style=\"color:white\">Enter a new operation name</h4>");
    toClient.println("<input type=\"text\" name=\"newoperation\" value=\"\">");
    toClient.println("<input name=\"addbtnOP\" type=\"submit\" value=\"Add an Operation\">");
    toClient.println(" <br>");
    toClient.println("<h4 style=\"color:white\">Operations</h4>");
    toClient.println("<select name=\"operationlist\">");
    String opstr = "";
    int ops = operation_vector.size();
    for (int i = 0; i<ops; i++) {
        opstr = (String)operation_vector.elementAt(i);
        toClient.println("<option>"+opstr+"");
    }
    toClient.println("</select>");
    toClient.println("<input name=\"debtnOP\" type=\"submit\" value=\"Remove the selected operation\">");
    toClient.println("<br><br><br><br><br>");

    /***** update and display objects *****/
    toClient.println("<h4 style=\"color:white\">Enter a new object name</h4>");
    toClient.println("<input type=\"text\" name=\"newobject\" value=\"\">");
    toClient.println("<input name=\"addbtnOBJ\" type=\"submit\" value=\"Add an Object\">");
    toClient.println(" <br>");
    toClient.println("<h4 style=\"color:white\">Objects</h4>");
    toClient.println("<select name=\"objectlist\">");
    String str = "";
    int s = object_vector.size();
    for (int i = 0; i<s; i++) {
        str = (String)object_vector.elementAt(i);
        toClient.println("<option>"+str+"");
    }
    toClient.println("</select>");
    toClient.println("<input name=\"debtnOBJ\" type=\"submit\" value=\"Remove the selected object\">");
    toClient.println("<br><br><br><br><br>");

    /***** update and display permissions *****/
    toClient.println("<h4 style=\"color:white\">Create a new permission</h4>");
    toClient.println("<table border =\"1\">");
    toClient.println("<tr>");
    toClient.println("<td><h5 style=\"color:white\">OPERATIONS</h5></td>");
    toClient.println("<td><h5 style=\"color:white\">OBJECTS</h5></td>");
    toClient.println("</tr>");
    toClient.println("<tr>");
    toClient.println("<td>");
    toClient.println("<select name=\"operationlist1\">");
    String opstr1 = "";
    int ops1 = operation_vector.size();
    for (int i = 0; i<ops1; i++) {
        opstr1 = (String)operation_vector.elementAt(i);

```



```

        toClient.println("<option>" + opstr1 + "");
    }
    toClient.println("</select>");
    toClient.println("</td>");

    toClient.println("<td>");
    toClient.println("<select name=\"objectlist1\">");
    String str1 = "";
    int s1 = object_vector.size();
    for (int i = 0; i < s1; i++) {
        str1 = (String) object_vector.elementAt(i);
        toClient.println("<option>" + str1 + "");
    }
    toClient.println("</select>");
    toClient.println("</td>");
    toClient.println("</tr>");
    toClient.println("</table>");

    toClient.println("<input name=\"addbtnPERM\" type=\"submit\" value=\"Make the selected operation -
object pair a permission\">");
    toClient.println("<br>");
    toClient.println("<h4 style=\"color:white\"Permissions</h4>");
    toClient.println("<select name=\"permissionlist\">");
    String permstr = "";
    int perms = permission_vector.size();
    for (int i = 0; i < perms; i++) {
        permstr = (String) permission_vector.elementAt(i);
        toClient.println("<option>" + permstr + "");
    }
    toClient.println("</select>");
    toClient.println("<input name=\"debbtnPERM\" type=\"submit\" value=\"Remove the selected
permission\">");
    toClient.println("<br><br><br><br><br>");
    toClient.println("</form></body></html>");
    toClient.close();
}
}

```

```

import java.io.*;
import java.net.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;
public class registration_servlet extends HttpServlet {
    public void doPost(HttpServletRequest req, HttpServletResponse res)

```

```

throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter toClient = res.getWriter();
    String firstnm = (req.getParameter("firstname")).trim();
    String lastnm = (req.getParameter("lastname")).trim();
    String rolenm = (req.getParameter("rolename")).trim();
    String firstnm = (req.getParameter("firstname")).trim();
    String lastnm = (req.getParameter("lastname")).trim();
    String rolenm = (req.getParameter("rolename")).trim();
    String rolenm = (req.getParameter("rolename")).trim();

    String usernm = firstnm.substring(0,1) + lastnm;
    String pw = "123456";
    toClient.println("Your user name is: "+usernm);
    toClient.println("Your password is: "+pw);
    try {Class.forName("com.mysql.jdbc.Driver").newInstance();}
    catch (Exception e) {System.out.println("Failed to load JDBC/ODBC driver.");}
    try {
        Connection con = DriverManager.getConnection(
            "jdbc:mysql://localhost/test", "h_zhao", "nRy2xN");
        Statement st = con.createStatement();
        st.executeUpdate("use test;");
        st.executeUpdate("insert into users values('"+usernm+"','"+pw+"','"+rolenm+"');");
        toClient.println("hello");
        toClient.close();
        st.close();
        con.close();
    } catch (Exception e) {e.printStackTrace();}
}
}

```

```

import java.io.*;
import java.net.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;
import java.util.*;
public class rolemgt_servlet extends HttpServlet {
    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter toClient = res.getWriter();
        Vector role_vector = new Vector();
        Vector conflictRole_vector = new Vector();
        Vector hierarchyRole_vector = new Vector();
        toClient.println("<html>");
    }
}

```

```

toClient.println("<body bgcolor=#065932>");
toClient.println("<h2 style='color:white'>ROLE MANAGEMENT</h2>");
toClient.println("<form action='http://cs.stmarys.ca:8000/servlet/rolemgt_servlet' method=post>");

/***** connect to database and show again *****/
try {Class.forName("com.mysql.jdbc.Driver").newInstance();
} catch (Exception e) {System.out.println("Failed to load JDBC/ODBC driver.");}
try {Connection con = DriverManager.getConnection("jdbc:mysql://localhost/h_zhao", "h_zhao",
"nRy2xN");

    Statement st = con.createStatement();
    st.executeUpdate("use h_zhao;");

    if (req.getParameter( "addbtn" ) != null){
        try {
            st.executeUpdate("insert into roles values ("+(req.getParameter("newrole")).trim()+");");
        }
        catch (SQLException sqle){
            toClient.println("<font color = 'red'>");
            toClient.println("duplicate entry, the role already exists");
            toClient.println("</font>");
        }
    }

    // delete the selected role
    else if (req.getParameter( "debtn" ) != null){
        st.executeUpdate("delete from roles where rolename="+(req.getParameter("roielist")).trim()+");");
        st.executeUpdate("delete from userRole where roiename="+(req.getParameter("rolelist")).trim()+");");
        st.executeUpdate("delete from permissionRole where
rolename="+(req.getParameter("rolelist")).trim()+");");
        st.executeUpdate("delete from conflictRoles where
rolename="+(req.getParameter("rolelist")).trim()+ " or itsconflictrole="+(req.getParameter("rolelist")).trim()+");");
        st.executeUpdate("delete from hierarchyRoles where
parent="+(req.getParameter("rolelist")).trim()+ " or child="+(req.getParameter("roielist")).trim()+");");
        st.executeUpdate("delete from sessions where
activeroles="+(req.getParameter("rolelist")).trim()+");");
    }

    // select and add the selected CONFLICT role pair
    else if (req.getParameter( "addbtnC" ) != null){
        st.executeUpdate("insert into conflictRoles values
("+(req.getParameter("rolelist1")).trim()+", " +(req.getParameter("rolelist2")).trim()+");");
    }
    // delete the selected CONFLICT role pair
    else if (req.getParameter( "debtnC" ) != null){
        String thePair = (req.getParameter("crolelist")).trim();
        int sp = thePair.indexOf("<");
        String r1 = thePair.substring(0,sp-1);
        String r2 = thePair.substring(sp+6,thePair.length());
        st.executeUpdate("delete from conflictRoles where rolename="+r1+" and itsconflictrole = "+r2+");");
    }
}

```

```

// select and add HIERARCHY role pair
else if (req.getParameter( "addbtnH" ) != null){
    st.executeUpdate("insert into hierarchyRoles values
    ("+(req.getParameter("rolelist1")).trim()+", "+(req.getParameter("rolelist2")).trim()+");");
}

// delete the selected HIERARCHY role pair
else if (req.getParameter( "debtnH" ) != null){
    String thePair = (req.getParameter("hrolelist")).trim();
    int sp = thePair.indexOf("-");
    String r1 = thePair.substring(0,sp-1);
    String r2 = thePair.substring(sp+5,thePair.length());
    st.executeUpdate("delete from hierarchyRoles where parent="+r1+" and child = "+r2+";");
}

ResultSet result = st.executeQuery("select * from roles;");
int rCol = result.findColumn( "rolename");
String rolenm="";
while(result.next()) {
    rolenm = result.getString(rCol);
    role_vector.addElement(rolenm);
}

ResultSet cresult = st.executeQuery("select * from conflictRoles;");
int crCol = cresult.findColumn( "rolename");
int ccrCol = cresult.findColumn( "itsconflictrole");
String crolenm="";
String ccrolenm="";
while(cresult.next()) {
    crolenm = cresult.getString(crCol);
    ccrolenm = cresult.getString(ccrCol);
    conflictRole_vector.addElement(crolenm+" <----> "+ccrolenm);
}

/***** store hierarchy roles data *****/
ResultSet hresult = st.executeQuery("select * from hierarchyRoles;");
int hpCol = hresult.findColumn( "parent");
int hcCol = hresult.findColumn( "child");
String hprolenm="";
String hcrolenm="";
while(hresult.next()) {
    hprolenm = hresult.getString(hpCol);
    hcrolenm = hresult.getString(hcCol);
    hierarchyRole_vector.addElement(hprolenm+" ----> "+hcrolenm);
}

st.close();
con.close();
} catch (Exception e) {e.printStackTrace();}

```

```

/***** update and display roles *****/
toClient.println("<table width=250 border=\\2\\>");
toClient.println("<tr><td align=center>");
toClient.println("<h3 style=\\\"color:white\\\">Enter a new role</h3>");
toClient.println("<input type=\\\"text\\\" name=\\\"newrole\\\" value=\\\"\\\">");
toClient.println("<input name=\\\"addbtn\\\" type=\\\"submit\\\" value=\\\"Add a Role\\\">");
toClient.println(" <br></td><td align=center>");

toClient.println("<h3 style=\\\"color:white\\\">Current Roles</h3>");
toClient.println("<select name=\\\"rolelist\\\">");
String str = "";
int s = role_vector.size();
for (int i = 0; i<s; i++) {
    str = (String)role_vector.elementAt(i);
    toClient.println("<option>"+str+""");
}
toClient.println("</select>");
toClient.println("<input name=\\\"debtn\\\" type=\\\"submit\\\" value=\\\"Remove selected role\\\">");
toClient.println("</td></tr></table><br><br><br>");

/***** display and update conflict roles *****/
toClient.println("<table width=250 border=\\2\\>");
toClient.println("<tr><td align=right>");
toClient.println("<h3 style=\\\"color:white\\\">Available Roles</h3>");
toClient.println("<select name=\\\"rolelist1\\\">");
String str1 = "";
int s1 = role_vector.size();
for (int i = 0; i<s1; i++) {
    str1 = (String)role_vector.elementAt(i);
    toClient.println("<option>"+str1+""");
}
toClient.println("</select>");

toClient.println("<br><br><br><br></td><td align=left>");

toClient.println("<h3 style=\\\"color:white\\\">Available Roles</h3>");
toClient.println("<select name=\\\"rolelist2\\\">");
String str2 = "";
int s2 = role_vector.size();
for (int i = 0; i<s2; i++) {
    str2 = (String)role_vector.elementAt(i);
    toClient.println("<option>"+str2+""");
}
toClient.println("</select>");
toClient.println("<br><br><br><br></td></tr>");

toClient.println("<tr align=center><td>");
toClient.println("<h3 style=\\\"color:white\\\">Conflict Role Pairs</h3>");
toClient.println("<select name=\\\"crolelist\\\">");
String strC = "";
int sc = conflictRole_vector.size();

```

```

        for (int j = 0; j < sc; j++) {
            strC = (String) conflictRole_vector.elementAt(j);
            toClient.println("<option>" + strC + "");
        }
        toClient.println("</select>");
        toClient.println("<input name=\"addbtnC\" type=\"submit\" value=\"Add Conflict Role pair\">");
        toClient.println("<input name=\"debbtnC\" type=\"submit\" value=\"Remove conflict role pair\">");
        toClient.println("</td>");

        /***** display hierarchy roles *****/
        toClient.println("<td>");
        toClient.println("<h3 style=\"color:white\">Hierarchical role relations</h3>");
        toClient.println("<select name=\"hrolelist\">");
        String strH = "";
        int sh = hierarchyRole_vector.size();
        for (int k = 0; k < sh; k++) {
            strH = (String) hierarchyRole_vector.elementAt(k);
            toClient.println("<option>" + strH + "");
        }
        toClient.println("</select>");
        toClient.println("<input name=\"addbtnH\" type=\"submit\" value=\"Add hierarchy relation\">");
        toClient.println("<input name=\"debbtnH\" type=\"submit\" value=\"Remove hierarchy relation\">");
        toClient.println("</td></tr>");
        toClient.println("</table>");
        toClient.println("</form></body></html>");
        toClient.close();
    }
}

```

```

import java.io.*;
import java.net.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;

public class session_servlet extends HttpServlet {
    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter toClient = res.getWriter();
        toClient.println("<html>");
        toClient.println("<body bgcolor=\"#065932\">");
        toClient.println("<form action=\"http://cs.stmarys.ca:8000/servlet/activeroles_servlet\"");
        method=post>");

        String sessionName = "";

```

```

String roles = "";
String temp = (req.getParameter("session")).trim();
int sIndex = temp.indexOf(":")+1;
sessionName = temp.substring(0, sIndex).trim();
roles = temp.substring(sIndex).trim();
toClient.println("You are in " + sessionName + " (with the following active roles)!");
toClient.println("<br><br>");
StringTokenizer str = new StringTokenizer(roles);

while (str.hasMoreTokens()) {
    String r = str.nextToken(",").trim();
    toClient.println("<li>");
    toClient.println("<input name=\"rolebtn\" type=\"submit\" value=\""+r+"\">");
    toClient.println("</li>");
}
toClient.println("</form></body></html>");
toClient.close();
}
}

```

```

import java.io.*;
import java.net.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;

```

```

public class submitMarks_servlet extends HttpServlet {
    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {res.setContentType("text/html");
        PrintWriter toc = res.getWriter();
        String stuid = null;
        String inst = null;
        String course = null;
        String work = null;
        String te = null;
        String ye = null;
        String mark = null;
        String comments = null;
        stuid = (req.getParameter("studentID")).trim();
        inst = (req.getParameter("instname")).trim();
        course = (req.getParameter("coursename")).trim();
        work = (req.getParameter("aqename")).trim();
        te = (req.getParameter("term")).trim();
        ye = (req.getParameter("year")).trim();
        mark = (req.getParameter("mark")).trim();
        comments = (req.getParameter("comment")).trim();
    }
}

```

```

        toc.println("<html>");
        toc.println("<body bgcolor=8470FF>");
        toc.println("</html>");

        // put the marks and comments into database
        try {Class.forName("com.mysql.jdbc.Driver").newInstance();
        } catch (Exception e) {toc.println("Failed to load JDBC/ODBC driver.");}
        try {
            Connection con = DriverManager.getConnection(
                "jdbc:mysql://localhost/h_zhao", "h_zhao", "nRy2xN");
            Statement st = con.createStatement();
            st.executeUpdate("use h_zhao;");

            int temp = st.executeUpdate("update academicRecords set marks="+mark+",
            comments="+comments+" where studentID="+stuid+" and instructorname="+inst+" and
            coursenam="+course+" and term="+te+" and year="+ye+" and aqe="+work+";");
            if (temp !=0)
                toc.println("<br><br><h3>Thank you for submitting the student mark!</h3>");
            else
                toc.println("<br><br><h3>The mark can not be submitted, please check the submit
            information!</h3>");
            toc.close();
            st.close();
            con.close();
        } catch (Exception e) {e.printStackTrace();}
    }
}

```

```

import java.io.*;
import java.net.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;

```

```

public class usermgt_servlet extends HttpServlet {
    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {

        res.setContentType("text/html");
        PrintWriter toClient = res.getWriter();

        toClient.println("<html>");
        toClient.println("<body bgcolor=#065932>");
        toClient.println("<head><title>USER MANAGEMENT</title></head>");
        toClient.println("<h2 style='color:white'>USER MANAGEMENT</h2>");
    }
}

```



```

toClient.println("<form action=\"http://cs.stmarys.ca:8000/servlet/usermgt_servlet\" method=post>");

toClient.println("<h3 style=\"color:white\">Current users</h3>");
toClient.println("<select name=\"userlist\">");

// connect to database and show data
try {Class.forName("com.mysql.jdbc.Driver").newInstance();
} catch (Exception e) {System.out.println("Failed to load JDBC/ODBC driver.");}
try {Connection con = DriverManager.getConnection("jdbc:mysql://localhost/h_zhao", "h_zhao",
"nRy2xN");

    Statement st = con.createStatement();
    st.executeUpdate("use h_zhao;");

    if (req.getParameter( "addbtn" ) != null){ //first screen, no button is pressed, so req is ok here
        try {st.executeUpdate("insert into users values ("+(req.getParameter( "newuser"
)).trim()+";";");}

        catch (SQLException sqle){
            toClient.println("<font color = \"red\">");
            toClient.println("duplicate entry, the user already exists");
            toClient.println("</font>");
        }
    }

    else if (req.getParameter( "debtn" ) != null){
st.executeUpdate("delete from users where username="+req.getParameter( "userlist" ).trim()+";");
st.executeUpdate("delete from userRole where username="+req.getParameter( "userlist" ).trim()+";");
st.executeUpdate("delete from sessions where username="+req.getParameter( "userlist" ).trim()+";");
st.executeUpdate("delete from academicRecords where studentID="+req.getParameter( "userlist"
)).trim()+";");
    }

    ResultSet result = st.executeQuery("select * from users;");
    int rCol = result.findColumn( "username");
    String usernm="";

    while(result.next()) {
        usernm = result.getString(rCol);
        toClient.println("<option>"+usernm+"");
    }
    st.close();
    con.close();
} catch (Exception e) {e.printStackTrace();}
toClient.println("</select>");
toClient.println("<input name=\"debtn\" type=\"submit\" value=\"Remove the selected user\">");
toClient.println("</form></body></html>");
toClient.close();
}
}

```

```

import java.io.*;
import java.net.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;

public class viewMarks_servlet extends HttpServlet {
    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        String temp = "hasNoRecord";

        // information from studentWorks page
        String psnm = null;
        String pinm = null;
        String pcnm = null;
        String paqnm = null;
        String pte = null;
        String pye = null;

        psnm = (req.getParameter("stname")).trim();
        pinm = (req.getParameter("instname")).trim();
        pcnm = (req.getParameter("coursename")).trim();
        paqnm = (req.getParameter("aqen")).trim();
        pte = (req.getParameter("term")).trim();
        pye = (req.getParameter("year")).trim();

        // information in database
        String snm = "";
        String inm = "";
        String cnm = "";
        String aqnm = "";
        String te = "";
        String ye = "";
        String m = "";
        String c = "";

        try { Class.forName("com.mysql.jdbc.Driver").newInstance(); }
        catch (Exception e) { System.out.println("Failed to load JDBC/ODBC driver."); }
        try {
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost/h_zhao", "h_zhao",
                "nRy2xN");
            Statement st = con.createStatement();
            st.executeUpdate("use h_zhao;");
            ResultSet result = st.executeQuery("select * from academicRecords;");
            int snmCol = result.findColumn("studentID");
            int inmCol = result.findColumn("instructorname");
            int cnmCol = result.findColumn("coursename");

```

```

int teCol = result.findColumn ("term");
int yeCol = result.findColumn ("year");
int aqeCol = result.findColumn ("aqe");
int mCol = result.findColumn ("marks");
int cCol = result.findColumn ("comments");

while(result.next()) {
    snm = result.getString(snmCol);
    inm = result.getString(inmCol);
    cnm = result.getString(cnmCol);
    te = result.getString(teCol);
    ye = result.getString(yeCol);
    aqenm = result.getString(aqeCol);
    m = result.getString(mCol);
    c = result.getString(cCol);

    if (psnm.compareTo(snm)==0 && pinm.compareTo(inm)==0 &&
pcnm.compareTo(cnm)==0 && paqenm.compareTo(aqenm)==0 && pte.compareTo(te)==0 &&
pye.compareTo(ye)==0 ){
        temp = "hasRecord";
        break;
    }
}
st.close();
con.close();
} catch (Exception e) { e.printStackTrace(); }

//-----
//          display student works
//-----
ot.println( "<html>" );
ot.println( "<body bgcolor=8470FF>" );
ot.println("<form action=\"http://cs.stmarys.ca:8000/servlet/submitMarks_servlet\" method=post>");
ot.println( "<br><br>" );
ot.println( "<center>" );
ot.println( "<h2><font face=\"Verdana\">Following are the available answers to the
assignments.</font></h2>");
ot.println( "</center>" );
ot.println( "<br><hr>" );

ot.println( " Student ID: "+psnm+"<br>" );
ot.println( " Instructor: "+pinm+"<br>" );
ot.println( " Course: "+pcnm+"<br><br>" );

ot.println( " Work name: "+paqenm+"<br>" );
ot.println( " Term: "+pte+"<br>" );
ot.println( " Year: "+pye+"<br><br><br>" );

if (temp.compareTo("hasRecord")==0) {
    ot.println( " Your mark is: "+m+" and comments are: "+c+"<br>" );
}

```

```

        else {ot.println( "<h2><font face=\"Verdana\">You has no records for this course, or the information
you entered is not correct, please reenter them.</font></h2>");}

```

```

        ot.println("</body>");
        ot.println("</html>");
        ot.close();
    }
}

```

```

/*****
*
*
*****/
                                HTML codes
                                *
/*****/

chiefAdministrator_login

<html>
<body bgcolor="#065932">
<head>
<title>RBAC TEST</title>
</head>

<br><br>
<h1 align="center" style="color:white">RBAC Chief Administrators </h1>
<form action=http://cs.stmarys.ca:8000/servlet/chiefAdministrator_login_servlet method=post>
<br>
<table>
<tr>
<td width="180"></td>
<td>
<table>
<tr>
<td width="100"><h4 style="color:white"> User1 Name: </td>
<td> <input name="username1" type="text" size="20" value=""><br></td>
</tr>
<tr>
<td width="100"> <h4 style="color:white">Password:</td>
<td> <input name="password1" type="password" size="20" value=""><br></td>
</tr>
</table>
</td>
</tr>
</table>

<br><br>

<table>
<tr>

```

```

<td width="180"> </td>
<td>
<table>
<tr>
<td width="100"> <h4 style="color:white">User2 Name: </td>
<td> <input name="username2" type="text" size="20" value="" color="white"><br>
</td>
</tr>
<tr>
<td width="100"> <h4 style="color:white">Password:</td>
<td> <input name="password2" type="password" size="20" value=""><br>
</td>
</tr>
</table>
</td>
</tr>
<tr>
<td width="180"> </td>
<td><input type=SUBMIT name="LOG IN" value="LOG IN"><br><br><br>
</td>
</tr>
<tr>
<td width="180"></td>
<td>
</td>
</tr>
</table>
</form>

<!--
<li> <a href="login.html">For normal users login</a></li>
-->

</body>
</html>

```

confirmUser

```

<html>
<body bgcolor="white">
<head>
<title>RBAC TEST</title>
</head>
<body>

<h3>RBAC TEST - For Registration </h3>

<form action="http://cs.stmarys.ca:8000/servlet/acceptUsers_servlet" method =post>
  <input name="search" type="submit" value="Get user information from Database">

```

```

</form>
</body>
</html>

```

assignment

```

<html>
<body bgcolor=#8470FF text="#000000" link="#044d84" vlink="#044d84" alink="#a2937a">
<br>
<form action=http://cs.stmarys.ca:8000/servlet/assignmentQuizExam_servlet method=post>
<SCRIPT LANGUAGE="JavaScript">

<!-- This script and many more are available free online at -->
<!-- The JavaScript Source!! http://javascript.internet.com -->

<!-- Begin
// Set the dates below
var goLiveDate = "20030618";
//var expireDate = "20040831";
var expireDate = "20040929";

var expireYear = expireDate.substr(0,4)
var expireMonth = expireDate.slice(4,-2)
var expireDay = expireDate.slice(6)
var liveYear = goLiveDate.substr(0,4)
var liveMonth = goLiveDate.slice(4,-2)
var liveDay = goLiveDate.slice(6)
var nowDate = new Date();
var day = nowDate.getUTCDate();
var month = nowDate.getUTCMonth();
var month1 = month + 1;

if (month1 < 10){ month1 = "0" + month1; }
if (day < 10){ day = "0" + day; }

var year = nowDate.getYear();
var GMTdate = year + "" + month1 + "" + day
if ((GMTdate < expireDate) && (GMTdate >= goLiveDate)) {
    document.write(" <div align='center'><b>This assignment posted on "
    +liveMonth+ "/" +liveDay+ "/" +liveYear+ "<br>and will be due on <font color=red>"
    +expireMonth+ "/" +expireDay+ "/" +expireYear+. </b> </font><br><br> "
    + "<table>"
    + "<tr><td>"
    + "Your Student ID: </td><td><input type='text' name='username'> </td></tr> "
    + "<tr><td>"
    + "Instructor name: </td><td><input type='text' name='instname'> </td></tr> "
    + "<tr><td>"

```

```

+ "Course name: </td><td><input type='text' name='coursename'> </td></tr> "

+ "<tr><td>"
+ "Assignment/Quiz/Exam title:</td><td> <input type='text' name='aqen'> </td></tr> "

+ "<tr><td>"
+ "Term: </td><td> <input type='text' name='term'> </td></tr> "

+ "<tr><td>"
+ "Year: </td><td> <input type='text' name='year'> </td></tr> "

+ "<tr><td></td>"
+ "<td><input name='search' type='submit' value='Submit'> </td></tr>"
+ "</table></div>")
}
</script>

<br><br>
<table align=center>
<tr>
<td width="150"></td>
<td> 1. What does RBAC stand for?
</td>
</tr>
<tr>
<td width="150"></td>
<td> 2. What is the advantage of RBAC?
</td>
</tr>
<tr>
<td width="150"></td>
<td> 3. What are the basic elements for RBAC?
</td>
</tr>
<tr>
<td width="150"></td>
<td> Please put your answers in the following area!
</td>
</tr>

<tr>
<td width="150"></td>
<td> <textarea name = "as" rows="7" cols="55"></textarea>
</td>
</tr>
</table>
</form>
</body>
</html>

```

currentCourses

```

<html>
<body bgcolor=8470FF>
<br><br><br>
<h1 align="center" style="color:white">For CSC 226:</h1>
<table>
<tr>
<td width="180"></td>
<td>
<ul>
<li><a href="assignment1.html"><h3>Assignment1</a></li>
<li><a href="quiz1.html"><h3>quiz1</a></li>
</td>
</tr>
</table>
<br><br><br>
<table>
<tr>
<td width="200"></td>
<td>
</td>
</tr>
</table>

</body>
</html>

```

facultyPage

```

<html>

<body bgcolor="8470FF" text="#000000" link="#044d84" vlink="#044d84"
alink="#a2937a">
<br><br><br>
<table>
<tr>
<td width="300"></td>
<td>FACULTY PAGE</td>
</tr>
<tr>
<td width="300"></td>
<td><br><br><br>
</td>
</tr>
<tr>
<td width="300"></td>
<td>

```



```

<ul>
<li> <a href="addDrop.html">Available Courses</a></li>
<li> <a href="csci1101.shtml">Program Info</a></li>
<li> <a href="csci2110.shtml">Online Payment</a></li>
<li> <a href="csci2112.shtml">Marks Online</a></li>
<li> <a href="csci2121.shtml">Account Statement</a></li>
<li> <a href="csci2132.shtml">Course Timetable</a></li>
<li> <a href="csci3110.shtml">Current Courses</a></li>
<li> <a href="csci3110.shtml">Programs, Faculty & Research</a></li>
<li> <a href="studentWorks.html">Evaluate students works</a></li>
</ul>
</td>
</tr>
</table>
</body>
</html>

```

finishRegistration

```

<html>
<body bgcolor=8470FF>
<head>
<title>RBAC TEST</title>
</head>
<br><br><br><br><br>
<h1 align="center" style="color:white">You have done your registration. You will get email with your username
and password within four working days.</h1>

<table>
<tr>
<td width="180"></td>
<td>
<li> <a href="registration.html"><h3>Back to homepage</a></h3></li>
</td>
</tr>
<tr>
<td width="200"></td>
<td>
</td>
</tr>
</table>
</body>
</html>

```

leveloneAdministrator

```

<html>

```

```

<body bgcolor="#065932">
<head>
<title>RBAC TEST</title>
</head>
<br><br>
<h1 align="center" style="color:white">RBAC Administration </h1>
<br>
<table>
<tr>
<td width="200"></td>
<td><form action="http://cs.stmarys.ca:8000/servlet/acceptUsers_servlet" method =post>
  <input name="search" type="submit" value="Confirm users' registration"></form>
</td>
</tr>
<tr>
<td width="200"></td>
<td><form action="http://cs.stmarys.ca:8000/servlet/usermgt_servlet" method =post>
  <input name="search" type="submit" value="User Management"></form>
</td>
</tr>
</table>
<br><br>
<table>
<tr>
<td width="200"></td>
<td>
</td>
</tr>
</table>
</body>
</html>

```

levelthreeAdministrator

```

<html>
<body bgcolor="#065932">
<head>
<title>RBAC TEST</title>
</head>
<h1 align="center" style="color:white">RBAC Administration </h1>
<table>
<tr>
<td width="180"></td>
<td>
  <form action="http://cs.stmarys.ca:8000/servlet/acceptUsers_servlet" method =post>
    <input name="search" type="submit" value="Confirm users' registration">
  </form><br>

  <form action="http://cs.stmarys.ca:8000/servlet/rolemtg_servlet" method =post>

```

```

<input name="search" type="submit" value="Role Management">
</form><br>

<form action="http://cs.stmarys.ca:8000/servlet/usermgt_servlet" method =post>
<input name="search" type="submit" value="User Management">
</form><br>

<form action="http://cs.stmarys.ca:8000/servlet/permissionmgt_servlet" method =post>
<input name="search" type="submit" value="Permission Management">
</form><br>

<form action="http://cs.stmarys.ca:8000/servlet/assignmgt_servlet" method =post>
<input name="search" type="submit" value="Role Assignment Management">
</form><br>
</td>
</tr>
<tr>
<td width="180"></td>
<td>
</td>
</tr>
</table>
</body>
</html>

```

leveltwoAdministrator

```

<html>
<body bgcolor="#065932">
<head>
<title>RBAC TEST</title>
</head>

<br><br>
<h1 align="center" style="color:white">RBAC Administration </h1>
<br>
<table>
<tr>
<td width="200"></td>
<td><form action="http://cs.stmarys.ca:8000/servlet/acceptUsers_servlet" method =post>
  <input name="search" type="submit" value="Confirm users' registration"></form>
</td>
</tr>
<tr>
<td width="200"></td>
<td><form action="http://cs.stmarys.ca:8000/servlet/usermgt_servlet" method =post>
  <input name="search" type="submit" value="User Management">
</form>
</td>

```

```

</tr>
<tr>
<td width="200"></td>
<td><form action="http://cs.stmarys.ca:8000/servlet/permissionmgt_servlet" method =post>
  <input name="search" type="submit" value="Permission Management"></form>
</td>
</tr>
<tr>
<td width="200"></td>
<td><form action="http://cs.stmarys.ca:8000/servlet/assignmgt_servlet_admll" method =post>
  <input name="search" type="submit" value="Role Assignment Management"></form><br><br>
</td>
</tr>
<tr>
<td width="200"></td>
<td>
</td>
</tr>
</table>
</body>
</html>

```

login

```

<html>
<body bgcolor=8470FF>
<head>
<title>RBAC TEST</title>
</head>
<p>
<h1 align="center" style="color:white">Welcome to E-education </h1>

<form action=http://cs.stmarys.ca:8000/servlet/login_servlet method=post>
<!--create a table with on borders -->
<table>
<tr>
<td width="200"></td>
<td>
<table>
<tr>
<td width="100"> User Name: </td>
<td> <input name="username" type="text" length="20" value=""></td>
</tr>
<tr>
<td width="100"> Password:</td>
<td> <input name="password" type="password" length="20" value=""></td>
</tr>
</table>
</td>

```

```

</tr>
<tr>
  <td width="200"></td>
  <td> <input type=SUBMIT name="LOG IN" value="LOG IN">
</form>
</td>
</tr>

<!--
<br><br><br>
<li> <a href="chiefAdministrator_login.html">For Chief Administrators login only</a></li>
-->
<tr>
  <td width="200"></td>
  <td>
    <li><a href="introduction.html"><h4>The introduction about the e-ecucation</h4></a></li>
  </td>
</tr>
<tr>
  <td width="200"></td>
  <td>
    <li><a href="program.html"><h4>Program information</h4></a></li>
  </td>
</tr>
<tr>
  <td width="200"></td>
  <td>
    <li><a href="registration.html"><h4>For a New User</h4></a></li>
  </td>
</tr>
<tr>
  <td width="200"></td>
  <td>
    <p><h3 style="color:white">
    If you have any concern, please contact us at :
    <a href="mailto:hong.zhao@stmarys.ca?subject=Hello%20again">Send Mail</a></h3>
    </p>
  </td>
</tr>
<tr>
  <td width="200"></td>
  <td>
    
  </td>
</tr>
</table>
</body>
</html>

```

quiz

```

<html>
<body bgcolor=8470FF text="#000000" link="#044d84" vlink="#044d84" alink="#a2937a"><br><br>
<meta http-equiv="Refresh" content="10; URL=http://cs.stmarys.ca:8000/login.html">
<form action=http://cs.stmarys.ca:8000/servlet/assignmentQuizExam_servlet method=post>
<div align=center><b>This quiz must be completed in <font color=red> 30 minutes! <br>WATCH OUT YOUR
TIME! </font></b>
<br><br>
<table>
<tr>
<td width="100"></td>
<td>Your username: </td>
<td><input type="text" name="username"></td>
</tr>
<tr>
<td></td>
<td>Instructor name:</td>
<td><input type="text" name="instname"></td>
</tr>
<tr>
<td></td>
<td>Course name: </td>
<td><input type="text" name="coursename"></td>
</tr>
<tr>
<td></td>
<td>Assignment/Quiz/Exam and number: </td>
<td><input type="text" name="aqen"></td>
</tr>
<tr>
<td></td>
<td>Term:</td>
<td><input type="text" name="term"></td>
</tr>
<tr>
<td></td>
<td>Year:</td>
<td><input type="text" name="year"></td>
</tr>
<tr>
<td></td><td></td>
<td><input name="search" type="submit" value="Submit"></td>
</tr>
</table>
</div>

<br>
<table align=center>
<tr>
<td width="200"></td>

```

```

<td> 1. List three access control technologies? </td>
</tr>
<tr>
<td width="200"></td>
<td> 2. List the advantage and disadvantage for each of them?
</td>
</tr>
<tr>
<td width="200"></td>
<td> 3. What is the major difference between RBAC and other access control?
</td>
</tr>
<br><br>
<tr>
<td width="200"></td>
<td> Please enter your answers in the following areal
</td>
</tr>

<tr>
<td width="200"></td>
<td> <textarea name = "as" rows="7" cols="55"></textarea>
</td>
</tr>
</table>
</form>
</body>
</html>

```

registration

```

<html>
<body bgcolor=8470FF>
<head>
<title>RBAC TEST</title>
</head>
<h1 align="center" style="color:white">For Registration for a New User</h1>
<form action="http://cs.stmarys.ca:8000/servlet/collectUserData_servlet" method =post>
<table>
<tr>
<td width = "180"></td>
<td>
<table>
<tr>
<td width = "100">Last Name:</td>
<td><input name="lastname" type="text" length="20" value="">
</td>
</tr>
</table>

```

```

<tr>
  <td width = "100">First Name:</td>
  <td><input name="firstname" type="text" length="20" value=""></td>
</tr>
<tr>
  <td width = "100">SIN:</td>
  <td><input name="sin" type="text" length="20" value=""></td>
</tr>
<tr>
  <td width = "100">Role Name:</td>
  <td><input name="rolename" type="text" length="20" value=""></td>
</tr>
<tr>
  <td width = "100">Address:</td>
  <td><input name="address" type="text" length="20" value=""></td>
</tr>
<tr>
  <td width = "100">PhoneNumber:</td>
  <td><input name="phone" type="text" length="20" value=""></td>
</tr>
<tr>
  <td width = "100">E-Mail:</td>
  <td><input name="email" type="text" length="20" value=""></td>
</tr>
<tr>
  <td width = "100">Online Pay:</td>
  <td><input name="pay" type="text" length="20" value=""></td>
</tr>
<tr>
  <td width = "100"><input name="search" type="submit" value="Submit">
  </form></td>
</tr>
</table>
</tr>
<tr>
  <td width="180"></td>
  <td>
  </td>
</tr>
</table>
</body>
</html>

```

relogin

```

<html>
<body bgcolor=8470FF>
<head>
<title>RBAC TEST</title>

```



```

</head>
<br><br>
<h1 align="center" style="color:white">Welcome to E-education </h1>
<form action=http://cs.stmarys.ca:8000/servlet/login_servlet method=post>
<h3 align="center" style="color:pink">Please put the correct user
name and password, and try again!</h3>
<table>
<tr>
<td width="180"></td>
<td>
<table>
<tr>
<td width="80">User Name:</td>
<td><input name="username" type="text" length="20" value="">
</td>
</tr>
<tr>
<td width="80">Password:</td>
<td><input name="password" type="password" length="20" value="">
</td>
</tr>
</table>
</td>
</tr>
<tr>
<td width="180"></td>
<td><input type=SUBMIT name="LOG IN" value="LOG IN"></form>
</td>
</tr>
</table>
</td>
</tr>
<table>
<tr>
<td width="180"></td>
<td><li> <a href="login.html"><h3>Back to e-Education Homepage</a></h3></li>
</tr>
<tr>
<td width="180"></td>
<td>
</td>
</tr>
</table>
</body>
</html>

```

studentPage

<html>

```

<body bgcolor="#8470FF" text="#000000" link="#044d84" vlink="#044d84"
alink="#a2937a">
<br><br><br>
<table>
<tr>
<td width="300"></td>
<td>STUDENT PAGE</td>
</tr>
<tr>
<td width="300"></td>
<td><br><br><br>
</td>
</tr>
<tr>
<td width="300"></td>
<td>
<ul>
<li><a href="addDrop.html">Add/Drop Courses</a></li>
<li><a href="program.html">Program Info</a></li>
<li><a href="onlinePayment.html">Online Payment</a></li>
<li><a href="studentsMarks.html">Marks Online</a></li>
<li><a href="accountStatement.html">Account Statement</a></li>
<li><a href="courseTimetable.html">Course Timetable</a></li>
<li><a href="course.html">Current Courses</a></li>
<li><a href="adm"></a></li>
<li><a href="stu.html">stu</a></li>
<li><a href="fac"></a></li>
<li><a href="sta"></a></li>
</ul>
</td></tr></table>
</body>
</html>

```

studentsMarks

```

<html>
<body bgcolor=8470FF>
<br><br><br>
<form action=http://cs.stmarys.ca:8000/servlet/ViewMarks_servlet method=post>
<div align='center' style="color:white">
Please fill the following information to get your marks.<br><br>
<table>
<tr>
<td width="80"></td>
<td>Instructor name:</td>
<td><input type="text" name="instname"></td>
</tr>
<tr>
<td width="80"></td>

```

```

<td>Your student ID:</td>
<td><input type="text" name="sname"></td>
</tr>
<tr>
<td>
</td>
<td>Course name:</td>
<td><input type="text" name="coursename"></td>
</tr>
<tr>
<td></td>
<td>Assignment/Quiz/Exam and number: </td>
<td><input type="text" name="aqen"></td>
</tr>
<tr>
<td></td>
<td>Term:</td>
<td><input type="text" name="term"></td>
</tr>
<tr>
<td></td>
<td>Year:</td>
<td><input type="text" name="year"></td>
</tr>
<tr>
<td></td>
<td><input name='search' type='submit' value='View Marks'></td>
</tr>
</table>
</div></form>

<table>
<tr>
<td width="100"></td>
<td>
</td>
</tr>
</table>
</body>
</html>

```

studentWorks

```

<html>
<body bgcolor=8470FF text="#000000" link="#044d84" vlink="#044d84" alink="#a2937a"><br><br>
<form action=http://cs.stmarys.ca:8000/servlet/evaluateWorks_servlet method=post>
<div align='center'><b>Please fill the following information to get the student work! </b> <br><br>
<table>
<tr>

```

```

<td width="100"></td>
<td>Instructor username:</td>
<td><input type="text" name="instname"> </td>
</tr>
<tr>
<td></td>
<td>Student ID:<td>
<td><input type="text" name="sname"></td>
</tr>
<tr>
<td></td>
<td>Course name:</td>
<td><input type="text" name="coursename"></td>
</tr>
<tr>
<td></td>
<td>Assignment/Quiz/Exam and number: </td>
<td><input type="text" name="aqen"></td>
</tr>
<tr>
<td></td>
<td>Term: <input type="text" name="term"></td>
<td>Year: <input type="text" name="year"></td>
</tr>
<tr>
<td></td>
<td><input name="search" type="submit" value="Get Student work and Evaluate"> </td>
</tr>
</table><br><br>
<table>
<tr>
<td width="180"></td>
<td>

</td>
</tr>
</table>
</div>
</form>
</body>
</html>

```